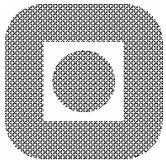
NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK



# HOVEDOPPGAVE

| | |
|---|---|
| Kandidatens navn: | John Arne Skjervold Pedersen |
| Fag: | Datateknikk |
| Oppgavens tittel (norsk): | Bruk av standard kontorapplikasjoner for lagring i et proprietært Content Management System |
| Oppgavens tittel (engelsk): | **Use of standard office applications for storing in a proprietary Content Management System** |

Oppgavens tekst:

Sentrale innholdsarkiver eller Content Management Systemer (CMS) er forretningskritiske IT-systemer som lever av å formidle kunnskap og fakta. Slike systemer skal sikre lengst mulig levetid på innholdet og maksimal spredning på ulike media som nett og trykksaker.

En utfordring knyttet til slike systemer er at de ofte stiller krav til innføring av nye arbeidsmetoder og kompetanse på bruk av nye systemer. Dette hindrer adopsjon og gevinster ved implementasjon. Denne oppgaven skal se nærmere på bruk av CMS som et usynlig bindingsverk og innholdsarkiv mellom de applikasjoner som benyttes i organisasjonen, slik at sluttbrukerne ikke får noen direkte innbefatning med nye grensesnitt, samtidig som alt innhold arkiveres og merkes i et sentralt innholdsarkiv

Oppgaven går ut på å:

1. Redegjøre for problemstillinger knyttet til sømløs integrasjon mellom tredjepartsapplikasjoner og et sentralt Content Management System.
2. Lage en prototype på sømløs integrasjon basert på WebDAV-protokollen mellom CMS og Microsoft Word og Adobe FrameMaker.
3. Lage en prototype på ekstrahering av metadata fra henholdsvis Word-dokumenter og Adobe-filer, eksemplifisert ved en anvendelse.
4. På bakgrunn av erfaringer fra bruk av prototypen vurdere egnethet for typer av Applikasjoner

Implementasjon og databasemodellering (Oracle) gjøres i Inspera Platform rammeverket (J2EE).

| | |
|---|---|
| Oppgaven gitt: | 20. januar 2003 |
| Besvarelsen leveres innen: | 16. juni 2003 |
| Besvarelsen levert: | 16. juni 2003 |
| Utført ved: | Institutt for Datateknikk og Informasjonvitenskap |
| Veileder: | Ingeborg Sølvberg og Trond Aalberg |

Trondheim, 16. juni 2003

Ingeborg Sølvberg

# Sammendrag

Content Management Systemer (CMS) tas i bruk i stadig flere bedrifter. Bred funksjonalitet støtter opp under stadig flere prosesser. Dette fører til spesialiserte og ofte kompliserte grensesnitt som brukeren må forholde seg til, og utgjør en stor kostnad ved opplæring i forbindelse med bytte av CMS.

Denne rapporten ser på konseptet sømløs integrasjon mellom arbeidsverktøy og CMS. Med sømløs integrasjon menes at innholdsarkivet i CMS'et integreres i brukerens arbeidsmiljø. På denne måten usynliggjøres CMS'et og brukeren forholder seg til innholdsarkivet som om det var en lokal filstruktur. Denne typen integrasjon muliggjøres ved hjelp av WebDAV. WebDAV er en protokoll for kommunikasjon mellom et CMS og en klient. Ved å benytte Microsofts WebDAV-klient WebFolders kan et WebDAV-kompatibelt CMS navigeres og anvendes som en vanlig Windows-mappe, og CMS'et oppleves som integrert med arbeidsmiljøet.

For å evaluere muligheten for slik integrasjon ble WebDAV-støtte implementert i et proprietært CMS. Her har det vist seg at CMS'et tilbød mesteparten av funksjonaliteten som benyttes i WebDAV, så en implementasjon innebar i hovedsak mapping av WebDAV-funksjoner til eksisterende funksjonalitet. I tillegg til generell støtte for WebDAV-protokollen ble det implementert funksjonalitet for ekstrahering av metadata og import av filer fra to WebDAV-kompatible applikasjoner, Microsoft Word og Adobe Framemaker. Ved å ha mulighet til å ekstrahere metadata fra dokumenter kan merking av metadata flyttes vekk fra innholdsarkivet og ut i sluttbrukerprogrammene, enda et steg på veien mot sømløs integrasjon. Importfunksjonalitet gjør at filene som importeres blir innholdsdokumenter i CMS'et i motsetning til kun filer. Dermed kan tredjepartsapplikasjoner brukes til å generere og revidere innhold til CMS'et.

En brukertest av prototypeimplementasjonen ga ikke noen definitive svar, men tydet på at sømløs integrasjon kan ha postensiale, om ikke i situasjonen som ble testet. Her var brukeren fornøyd med eksisterende arbeidssituasjon og så liten grunn til å endre, selv om hun mente den nye metoden kunne være nyttig. Selv om WebFolder sannsynligvis er den mest brukte WebDAV-klienten har testing vist at den er svært begrenset i metadatahåndtering og støtter heller ikke låsing av filer. Dette begrenser de situasjoner der sømløs integrasjon ved WebDAV kan benyttes.

WebDAV implementeres i stadig flere produkter, og et CMS som støtter WebDAV vil kunne integreres med et stadig økende antall WebDAV-kompatible produksjonsverktøy. Sømløs integrasjon med WebDAV er ikke aktuelt i alle situasjoner, men ettersom produktene som støtter protokollen utvikles vil også integrasjonen og bruksnytte forbedres.

# Forord

Denne oppgaven er skrevet i forbindelse med hovedoppgave for Sivilingeniørutdanningen ved NTNU. Oppgaven er gitt av professor Ingeborg Sølvberg ved Institutt for Datateknikk og Informasjonsvitenskap, NTNU, etter ønske fra Bjørn Rustberggard hos Inspera AS.

Oppgaven er utført av John Arne S. Pedersen, diplomstudent ved Institutt for Datateknikk og Informasjonsvitenskap.

Undertegnede vil rette en stor takk til Ingeborg Sølvberg og Trond Aalberg for nyttig veiledning gjennom hele perioden. Takker også Bjørn Rustberggard for innspill, idéer og entusiasme, og Turi Lundeby hos Aschehoug som testet systemet.

Til slutt må en takk rettes til Ingvild Kanck som fant seg i å bli nedprioritert gjennom nok en prosjektinnspurt.

Trondheim, 16. juni 2003

John Arne S. Pedersen

# Innholdsfortegnelse

# Figurliste



# Tabelliste

# 1 Innledning

Content Management Systemer er innholdsarkiver som i tillegg til selve innholdslagringen støtter arbeidsflyt og annen funksjonalitet for å støtte opp under produksjon og distribusjon av innhold. Ettersom Content Management Systemer omfavner flere aspekter ved innholdsforvaltning og prosessene rundt, kan grensesnittene og funksjonaliteten bli for omfattende for mange brukere. Dersom man kun ønsker å benytte seg av innholdsarkivsfunksjonaliteten kan grensesnittet gjøre oppgaven vanskeligere enn nødvendig. I tillegg til å innføre kompliserende grensesnitt og oppgaver heves bruksterskelen, og den jevne bruker vil trenge opplæring for å bruke systemet selv om hun bare skal utføre enkle oppgaver.

Ved å knytte arbeidsverktøyet direkte til innholdsarkivet abstraherer man bort dette mellomleddet, og oppnår en sømløs integrasjon mellom arbeidsverktøy og innholdsarkiv. Dermed unngår man ekstraarbeid og kan fokusere på innholdet. Innhold kan åpnes, revideres og lagres tilbake i innholdsarkivet direkte fra arbeidsverktøyet. Dersom i tillegg brukeren er kjent med arbeidsverktøyet sparer man den opplæring som ville vært nødvendig om hun skulle benytte et avansert CMS-grensesnitt.

Opplæring er også en kostnad ved bytte av arbeidsverktøy. I mange organisasjoner må brukerne bytte arbeidsverktøy dersom bedriftens sentrale CMS endres, siden disse er spesialisert til hverandre. Dette vil enten føre til ekstrakostnader ved opplæring av brukere til nye arbeidsverktøyer, eller at bedriften nøler med å ta i bruk nye CMS-systemer selv om disse er bedre egnet for bedriften. Dersom ulike arbeidsverktøy og CMS'er kunne samarbeide, ville disse problemene forsvunnet. Da kunne brukerne beholdt arbeidsverktøy selv om CMS'et ble byttet ut, eller velge å bytte arbeidsverktøy og fortsette å jobbe mot samme CMS.

## *1.1 Problemstilling*

Oppgaven hadde som mål å konstruere en protoype for sømløs integrasjon, gjennom å implementere støtte for WebDAV-protokollen i et eksisterende CMS, Inspera Content Server. Implementasjonen ble avgrenset til å kun støtte det som var nødvendig for testing gjennom valgt WebDAV-klient, Microsoft WebFolder. Denne klienten ble valgt fordi den gir et Utforsker-lignende grensesnitt, og illustrerer dermed den sømløse integrasjonen med innholdsarkivet på Windows-platformen.

For å kunne teste brukbarheten av sømløs integrasjon i en praktisk anvendelse måtte en brukertest avholdes. Til dette formål ble Aschehoug involvert siden de allerede benyttet Inspera Content Server til digital publisering. Testcase ble dermed en redaksjonssekretærs arbeidsoppgaver når dokumenter ankommer fra forfattere og skal lagres i innholdsarkivet.

I oppgaveteksten er to applikasjoner nevnt, Microsoft Word og Adobe FrameMaker. Under arbeide med oppgaven ble det klart at fokus ble lagt mer over på Word enn FrameMaker. Siden FrameMaker ikke ble anvendt av brukeren, ble oppgaven avgrenset til kun å vise proof-of-concept import av FrameMaker.

## 1.2 Leseveiledning

Rapporten består av 3 hoveddeler:

- Teoridel: Kapittel 2 og 3.
- Implementasjon: Kapittel 4.
- Evaluering/Konklusjon: Kapittel 5 og 6.

Teoridelen gjennomgår først hva som legges i begrepene Content Management (System), dokumentproduksjon og sømløs integrasjon. Deretter følger en beskrivelse av WebDAV, en protokoll som anvendes for å knytte produksjonsverktøyer direkte til innholdsarkiver, og på den måten oppnå sømløs integrasjon. Microsofts WebDAV-klient WebFolder beskrives, og det nevnes også andre alternative WebDAV-klienter.

Implementasjonen består av tre deler. Først beskrives implementasjonen av selve WebDAV-støtten, en modul som kommuniserer med WebDAV-klienter og gir de tilgang til kjernefunksjonaliteten i innholdsarkivet. Deretter beskrives konstruksjonen av ekstraktorer for metadata fra Word- og Adobe-filer. Den siste delen omhandler design og implementasjon av en ny struktur for import av innhold til arkivet.

Evalueringen gjør først en teknisk evaluering av systemet, før brukertesten beskrives. Denne gir en innføring i testcaset og omtaler brukerens arbeidsmåte før og ved bruk av sømløs integrasjon. Avslutningsvis konkluderes det på basis av det som er nevnt i brukertesten og den tekniske evalueringen, og også av diskusjonen som er gjort i implementasjonsdelen.

# 2 Content Management

Begrepet Content Management (innholdsforvaltning) er blitt et markant buzzword de senere år, men det er vanskelig å enes om en definisjon på hva innholdsforvaltning er. Banalt kan innholdsforvaltning sies å være forvaltning av innhold [1], men hva er i så fall innhold? [2] gir skiller mellom *informasjon* og *innhold*. Informasjon er all lagret kommunikasjon, som for eksempel tekst, lyd, bilder, film og lignende. Innhold er informasjon tatt i bruk, altså pakket og presentert for en grunn. [1] definerer innholdforvaltning som samling, forvaltning og presentasjon.

- *Samling* betyr å opprette eller samle inn informasjon som så konverteres til et internt format (for eksempel XML). Disse bitene kalles innholdselementer, og inneholder metadata for å lettere kunne organisere, lagre og gjenfinne informasjonen.
- Innholdet blir *forvaltet* i et innholdsarkiv, for eksempel en database eller et filsystem. I tillegg til innholdselementene kan innholdsarkivet inneholde administrativ informasjon.
- For å gjøre innholdet tilgjengelig *presenteres* det i publikasjoner som websider, trykte dokumenter eller annet.

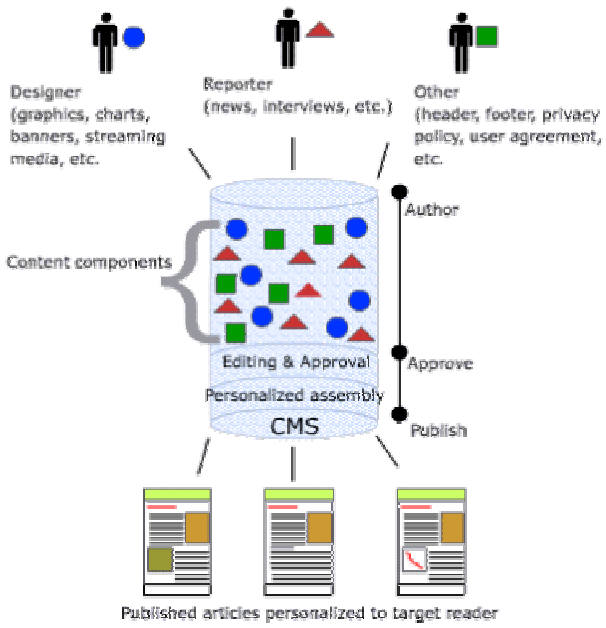
Fokus i denne rapporten ligger på de to første punktene; oppretting og forvaltning.

## 2.1 Content Management System

Begrepet Content Management System (CMS) benyttes ofte innen digital publisering om systemer som håndterer innholdet. [3] definerer et CMS ut fra dets funksjon: Å forenkle skapelse og administrering av digitalt innholdt. To hovedaspekter ved et CMS er nevnt:

- Skille mellom innhold og presentasjon. Innholdet blir enklere vedlikeholdt og gjenbrukbart ved å separere det fra presentasjonen. Samme innhold kan benyttes på forskjellige medier og i forskjellige visninger for å personalisere innholdet til brukeren.
- Arbeidsflyt. Ved å benytte arbeidsflyt kan de forskjellige fasene i innholdselementets levetid understøttes av systemet. Eksempelvis kan et dokument etter at det er levert av en forfatter automatisk sendes til en redaktør for godkjenning, og når denne er gitt publiseres dokumentet automatisk. Etter en bestemt tid kan så dokumentet fjernes og arkiveres.

I denne sammenheng vil det fokuseres på det første aspektet. Skillet mellom innhold og presentasjon oppnås ved å lagre innholdet som innholdselementer. Et innholdselement er en bit informasjon, innkapslet med metadata, lagret i innholdsarkivet. Disse innholdselementene kan så kombineres og stiles for å presenteres til brukeren. Figur 1 (hentet fra [3]) illustrerer dette. Her leverer flere forfattere innhold til CMS'et, og innhold fra forskjellige forfattere kan kombineres i forskjellige stiler for å skape personaliserte visninger til ulike målgrupper (gjerne på ulike medie). På figuren er også arbeidsflyten illustrert, fra innholdet ankommer (*author*-fase) til det godkjennes (*approve*) og publiseres (*publish*).




**Figur 1: Content Management System**

## 2.2 Dokumentproduksjon

For å forstå hvordan produksjonsverktøyer og Content Management Systemer kan benyttes for å støtte innholdsproduksjon, er det nyttig å se på hvordan innhold opprettes. Dette kapitlet vil beskrive ulike aspekter ved dokumentproduksjon[1]. Kapittel 2.2.1 beskriver dokumentproduksjonsprosessen og hvordan den kan deles opp i faser og modelleres. I mange tilfeller vil flere forfattere samarbeide på samme dokument. Dette innebærer visse problemstillinger, og disse er beskrevet i kapittel 2.2.2. Deretter gis en kort beskrivelse av metadata i kapittel 2.2.3.

### 2.2.1 Modeller for dokumentproduksjon

For å få oversikt over hvordan et dokument blir til er det nyttig å modellere prosessene som fører dokumentet fra en idé i en (eller flere) forfatters hode til et ferdig produkt. Det finnes åpenbart ikke en definitiv og allmennakseptert modell for dokumentproduksjon. Hvordan prosessen foregår avhenger av organisasjonsrutiner, personene involvert, og oppgaven. En organisasjon kan ha tradisjonsrike rutiner som de holder på, tross at disse ikke nødvendigvis er best egnet for oppgaven. Samtidig kan personene som utfører oppgaven ha sine egne varianter av utførelse. Mest av alt vil prosessen variere i forbindelse med hva slags oppgave som utføres.

Generelt kan et dokument levetid deles inn i seks faser [4]:

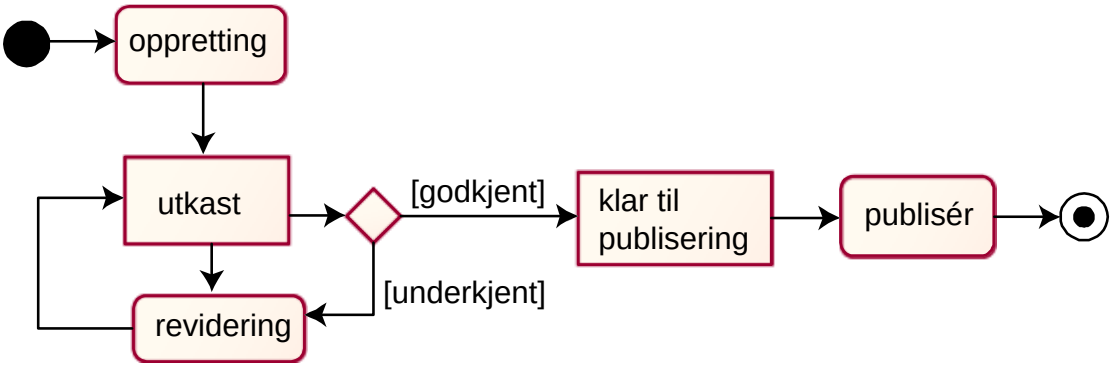
- **Skapelse.** Forfatteren oppretter dokumentet, vanligvis et utkast.
- **Gjennomgang.** Dokumentet blir korrekturlest og revidert.

[1] Det er verd å nevne noe om bruken av ordet dokument. Ordet dokument er ofte forbundet med tekst, men det er ikke nødvendigvis tilfelle. Innenfor CM-kretser kan digitale dokumenter være alt fra tekst til bilder, film og lyd, eller en kombinasjon. I denne rapporten brukes CM-definisjonen, og det vil spesifiseres nærmere dersom det er nødvendig å skille mellom tekstlig eller annen dokumenttype.

- **Lagring.** Dokumentet lagres i et innholdsarkiv.
- **Publisering.** Publisering av dokumentet, enten som trykksak eller på nett.
- **Arkivering.** Dokumentet bevares for ettertiden.
- **Henlegging.** Dokumentet har utspilt sin rolle og fjernes.

Dette er overordnede faser, og en modell som forsøker å vise dokumentproduksjon i et case vil ha mye mindre granularitet på fasene. Hvilken granularitet som velges, og hva som tas med av prosesselementer vil være avhengig av hvilken vinkling man har på modellen, det vil si hva den er ment benyttet til. I tillegg illustreres disse fasene vanligvis som en sekvensiell gang, mens det i praksis vil være itereringer mellom enkelte av fasene. De tre første fasene i dokumentets levetid kan sammen kalles fremstillingen av dokumentet, og vil fokuseres på videre. I denne kontekst er ikke publisering eller dokumentets endelikt gjenstand for videre studie.

Et eksempel på en modell for dokumentproduksjon er gitt i Figur 2. Her gjennomgår dokumentet iterasjon med revidering til det blir godkjent for publisering. Forfatter oppretter et utkast som så må godkjennes av en redaktør eller moderator. Hvis dokumentet blir underkjent må det revideres før det på ny kan sendes til godkjenning.




**Figur 2: Eksempel på produksjonsmodell**

Tilsvarende som Figur 2 kan det settes opp modeller for andre case. I kap. 5.2.1 gis en modell over bokproduksjonsprosessen hos forlagshuset Aschehoug, som er testcase for prototypen.

### 2.2.2 Samforfatterskap

Samforfatterskap (*collaborative authoring*) betegner samarbeid om å skape et innholdselement. Tanken er at en gruppe i samarbeid skal yte noe større eller bedre enn en enkelt person ville være i stand til. Begrepet samforfatterskap innebærer også noe mer en man legger i et begrep som medforfattere. Ved samforfatterskap samarbeider gruppen ofte om de samme delene av dokumentet, muligens til samme tid.

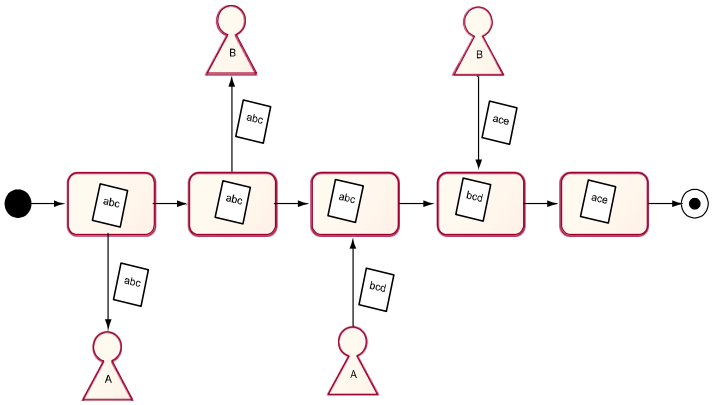
Samforfatterskap vil utøves på ulike måter, avhengende av kontekst. Det vil variere hvordan dokumentet opprettes, hvordan forfatterne samarbeider på dokumentet, hvordan det avgjøres at dokumentet er ferdig og lignende. En *arbeidsflyt* er en sekvensiell modell for hvordan en gitt oppgave gjøres, i praksis en realisering av en prosessmodell. Arbeidsflyt kontrolleres ved meldinger (for eksempel gjennom e-post, telefon, et

automatisert arbeidsflytverktøy eller annet) mellom gruppens medlemmer. En studie beskrevet i [5] viser 7 forskjellige mønstre for arbeidsflyt ved samforfatterskap:

1. Gruppen planlegger og skisserer oppgaven, så gjør hver av forfatterne sin del. Gruppen samler så de individuelle delene, og reviderer etter behov.
2. Gruppen planlegger og skisserer oppgaven og dokumentet, deretter utarbeider en av forfatterne et utkast som gruppen så endrer og reviderer.
3. Et medlem av gruppen planlegger og forfatter et utkast, gruppen reviderer deretter utkastet.
4. En person planlegger og forfatter et utkast, deretter reviderer ett eller flere medlemmer av gruppen dokumentet uten å konsultere den opprinnelige forfatteren.
5. Gruppen planlegger og forfatter utkastet, deretter reviderer en ett eller flere medlemmer av gruppen dokumentet uten å konsultere den opprinnelige forfatteren.
6. En person fordeler oppgaver, hvert medlem i gruppen utfører sin oppgave, en person samler og reviderer dokumentet.
7. En dikterer, en annen skriver og reviderer.

Studiet viste at den vanligste av disse var 3. (31 %), den sjeldneste 5. (3 %). Samforfatterskap kan deles i to kategorier [6]: *Shared mind* og *division of labour*. Den første kategorien betegner et samarbeid der forfatterne jobber på samme dokument på sin egen lokalitet. En koordinator er ansvarlig for å integrere endringene. I den andre kategorien deles jobben opp i uavhengige deloppgaver som så fordeles på forfatterne. Disse settes sammen til det ferdige dokumentet. Disse to kategoriene er parallelle med *asynkront* og *synkront* samarbeid. Av de nevnte samforfatterskapsmønstrene er 5. et eksempel på shared mind og 1. et eksempel på division of labour.

Et vanlig problem med samforfatterskap er samtidighetskontroll. Dersom flere forfattere jobber på samme dokument er det en risiko for at en forfatter overskriver en annens endringer. Dette er kjent i databaseterminologien som ”lost update”-problemet, og er vist i Figur 3. I figuren henter først forfatter A ut dokumentet og deretter forfatter B. Når de begge lagrer sine endringer etter tur vil forfatter A’s endringer forsvinne siden de ikke var med i dokumentet som B hentet ut.

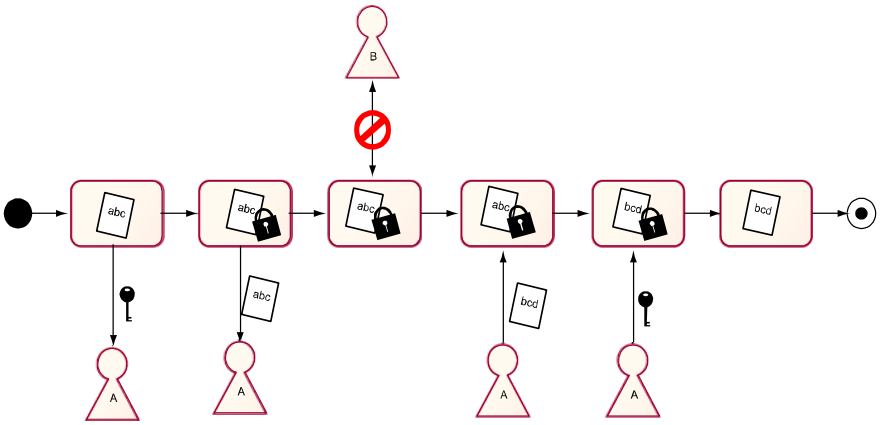



**Figur 3: "Lost update"-problematikk**

De to hovedparadigmene for samtidighetskontroll er låsing og sammenføyning (eng. *merging*). En tredje mulighet, og den utvilsomt enkleste, er den som er kjent som POTS

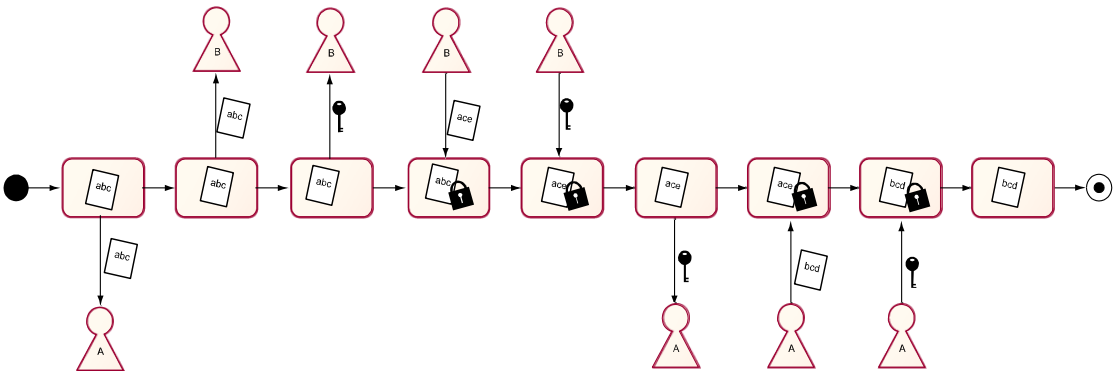
(Plain Old Telephone System), der brukere selv kommuniserer (for eksempel over telefon eller epost) og sier ifra når de er ferdige med sine endringer. En slik metodikk fungerer bra for lokalt samforfatterskap, men er både tungvindt og tidkrevende for en bredt distribuert forfatterskare.

Ved låsing gjør en forfatter kjent for systemet hans intensjoner om å endre dokumentet. Systemet låser da dokumentet for andre, og gir ingen andre muligheter til å gjøre endringer på det. Når denne forfatteren har gjort sine endringer lagrer han de, og låser opp dokumentet. Denne metodikken er vist i Figur 4. Her låser forfatter A dokumentet før han henter det ut for redigering, og når forfatter B forsøker å hente ut dokumentet oppdager han at det er låst[2] og venter til det er låst opp med å gjøre sine endringer.




**Figur 4: Bruk av låser for samtidighetskontroll**

Dog er det selv med låsefunksjonalitet mulig med ubevisst overskriving, dersom det ikke er påkrevd å ta ut en lås på et dokument i forkant av uthenting av dokumentet for endringer. Figur 5 viser et eksempel på dette. Her henter både forfatter A først ut dokumentet (muligens ikke med planer om endringer), og låser det senere før han gjør sine endringer. Men da har forfatter B hentet, låst, endret og låst opp dokumentet i mellomtiden. Disse endringene har ikke A fått med seg, og blir overskrevet.




**Figur 5: Manglende låskontroll gir tapte oppdateringer**

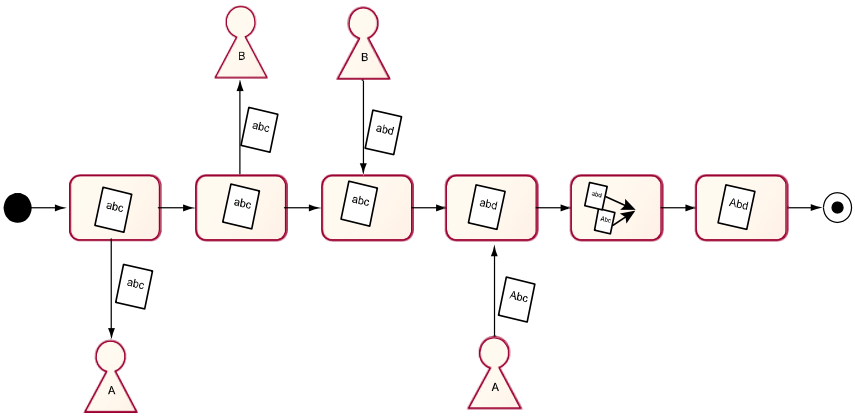
Slike tilfeller kan hindres med leselåser eller strenge krav for låsing. Fra databasedomenet er dette og andre problemer velkjente, og metoder for å unngå dem er velpubliserte.

[2] Det finnes forskjellige strategier for dette. Leselåser hindrer andre å hente ut et dokument som er låst, låsoppdaging gjør brukere oppmerksom på låsen, og i noen tilfelle merker brukerne ikke låsen før de forsøker å lagre. Metodikken for å hindre samtidighetsproblematikk avhenger av låsestrategi.

Ulike grader låsing (delte/eksklusive låser, leselåser) kan benyttes for å oppnå samtidighetskontroll, i tillegg varierer granularieten på låsene (om låser gjelder hele dokumentet eller kun deler). Ulike strategier benyttes i ulike systemer for samforfatterskap (se kapittel 2.3.1.2).

Hovedkritikken mot låsestrategien er utilgjengeligheten av et låst dokument. Et dokument som er låst kan ikke endres av enn den som eier låsen, og i tilfellet med leselåser heller ikke leses av andre enn eieren. Denne problematikken er åpenbart proporsjonal med granulariteten på låsene. Dersom kun et avsnitt av et dokument er låst, er problemet mindre enn om hele dokumentet (eller hele mappen) er låst. Et annet spørsmål som må adresseres er hva som gjøres med "gjenglemte" låser, det vil si låser som ikke låses opp (for eksempel på grunn av feil hos brukeren). En vanlig løsning på dette er å automatisk låse opp låser etter et gitt tidsintervall, men dette har åpenbare problemer. Dette og andre vanskeligheter må være gjennomtenkte for at en låseprotokoll skal kunne fungere.

En annen løsning på samtidighetskontrollproblematikken er sammenføyning, kjent fra CVS [7]. Her lar man flere forfattere gjøre endringer på dokumentet samtidig, og håndterer tilfellet der en to forfattere har gjort endringer på samme dokument i det nummer to forsøker å lagre. Systemet oppdager da at dokumentet har endret seg i arkivet siden forfatteren åpnet det, og en lagring vil overskrive disse endringene. De nye endringene må da sammenføyes med de lagrede endringene (Figur 6).




**Figur 6: Sammenføyning unngår tapte oppdateringer**

Hvordan dette gjøres avhenger fra system til system. Kombinering av begges endringer er den ideelle løsningen. I CVS kombinerer systemet automatisk endringer på ulike steder i samme tekstlige dokument, og skaper et dokument som innbefatter begge forfatternes endringer. Ulike endringer på samme tekst gir konflikter. I slike tilfeller overlates ansvaret om å kombinere endringene til den som forsøker å lagre. Han kan da velge å beholde sine eller den forriges endringer, eller manuelt kombinere disse. Samme framgangsmåte kan i teorien gjøres for bilder og andre innholdstyper, dog mer komplisert. En granularitet mindre enn hele innholdselementet innebærer kjennskap til innholdstypen. Dette er den framtredende kritikken mot sammenføyningsparadigmet, idet det innebærer en begrensning av innholdstyper for å ha noen praktisk nytte. Uten slik kjennskap kan systemet ikke gjøre mer enn å gjøre brukeren oppmerksom på at noen andre har endret dokumentet, og så la brukeren vurdere disse endringene.

I [8] observeres det at samforfatterskap ikke kun avhenger av en klart definert arbeidsflyt og samtidighetskontroll. Konvensjoner for metadata må også defineres, slik at disse blir konsistente, ikke bare for selve dokumentet men på tvers av dokumenter. På samme måte må det også defineres konvensjoner for filmanipulering, altså mappeorganisering, navnekonvensjoner og versjoner.

### 2.2.3 Metadata

Metadata, data om data, er informasjon om et informasjonsobjekt. I denne konteksten kan vi se på metadata som forskjellige typer beskrivende informasjon om en ressurs. Metadata kan kategoriseres i 5 hovedkategorier [9]:

- Administrativ, for eksempel rettighetsinformasjon.
- Deskriptiv, for eksempel tittel, beskrivelse, forfatter.
- Preserverende, for eksempel tilstand
- Teknisk, for eksempel format, kryptering
- Bruk, for eksempel brukslogg.

Alle disse er relevante for digitale dokumenter i et innholdsarkiv, selv om det kan være lett å fokusere på kun deskriptive metadata.

Det finnes ingen allmenn aksept over hvilken metadata som bør benyttes, like lite er dette hensiktsfullt eller mulig. Forskjellige anvendelser setter ulike krav til hvilken metadata som er nødvendig. Likevel er det mye av den samme metadata som benyttes fra system til system. Naturlige elementer som tittel, beskrivelse, opphavsmann og lignende benyttes i de fleste sammenhenger. Forskjellige standarder for metadata eksisterer eksempelvis kan MARC [10], RFC 1807 [11] og Dublin Core [12] nevnes. Det vil ikke gås nærmere inn på disse her, interesserte henvises til referansene.

I systemer der innhold importeres og eksporteres bør alt eller så mye som mulig av innholdselementenes tilhørende metadata medfølge til og fra systemet. Dette er ingen selvfølge, siden representasjonen og utvalget av metadata vil variere fra system til system. Felles taksonomier kan hjelpe på dette. Dersom metadatastrukturene følger velformulerte skjema kan det opprettes mappinger mellom dem.

## *2.3 CMS som arbeidsmiljø*

En utfordring i innholdsproduksjon er integrasjonen mellom produksjonsverktøyet og innholdsarkivet. Når kreasjonsperioden for dokumentet er over, må dokumentet lagres i innholdsarkivet. Innholdsarkivet er i så måte arbeidsmiljøet for brukerne, og et grensesnitt man må være kjent med for å arbeide effektivt. Dermed innebærer endringer/bytte av dette grensesnittet en stor kostnad idet brukerne må sette seg inn i et nytt arbeidsmiljø. Dette kan også gjelde ved bytte av arbeidsverktøy, dersom disse ikke kan knyttes til innholdsarkivet på samme måte som tidligere.

### 2.3.1 Verktøy for innholdsproduksjon

Produkter som benyttes under innholdsproduksjon kan deles i to grupper: Produkter som benyttes under utarbeiding og produksjon av selve innholdselementet, beskrevet i

2.3.1.1, og produkter som benyttes for å lagre dette innholdselementet, beskrevet i 2.3.1.2. Til sistnevnte hører vanligvis også metadatamerking, dog kan enkelte produksjonsprodukter også benyttes til noe metadatamerking.

#### 2.3.1.1 Produksjonsprodukter

Produksjonsprodukter er verktøy som benyttes til å skape og revidere innholdselementer.

Forskjellige verktøy skiller seg åpenbart i hva slags innholdstyper de produserer (dokumenter, bilder, osv), men også på hvilken støtte de har for for eksempel metadatamerking og eksport til standardformater. Proprietære filformater er tungvinte da de ikke kan leses av andre enn programmet selv. For å være nyttige må dokumentet kunne tolkes i innholdsarkivet slik at innholdet kan lagres strukturert. Denne oversettingen kan gjøres av programmet selv (ved eksport til standardformater som XML for dokumenter og jpeg/gif for bilder) eller ved hjelp av mellomvaren som knytter programmet til innholdsarkivet. For innholdstyper som lyd og bilder er det uvanlig med proprietære filformater, nesten alle billed- og lydredigeringsprogrammer støtter de standard filformatene. For dokumenter er situasjonen annerledes. Mange tekstbehandlere opererer med egne, inkompatible filformater. For tekstlige dokumenter er det ønskelig at innholdet er atskilt fra presentasjon, slik at dokumentet kun inneholder en logisk struktur og ingen visningsmal. Dette kan for eksempel gjennomføres ved at innholdet lagres som XML.

I denne sammenhengen vil det fokuseres på produksjonsverktøy for dokumenter. De to produksjonsverktøyene som vil diskuteres er Microsoft Word (versjon 2000, nyeste versjon (XP) skiller seg ikke markant fra denne) og Adobe Framemaker (versjon 7.0).

- *Microsoft Word*
  - *Generelt*: Svært utbredt tekstbehandler, anerkjent for brukervennlighet.
  - *Filformat*: Proprietært filformat, og få eksterne konverteringsprogrammer finnes siden strukturen i filformatet er hemmeligholdt av Microsoft. Word har mulighet til å lagre i andre formater inkludert RTF (Rich Text Format, [13]), et åpent format laget av Microsoft for å eksportere dokumenter på tvers av programmer. RTF er langt bredere støttet, og det finnes flere gratis konverteringsprogrammer for dette formatet. Det er ingen støtte for lagring av XML fra Word.
  - *Metadata*: Word støtter lagring av metadata (med en samling definerte og i tillegg mulighet for å legge til egendefinerte) både i .doc- og .rtf-dokumenter.
- *Adobe Framemaker*
  - *Generelt*: Avansert verktøy for produksjon av dokumenter og bøker. Inneholder større fokus på struktur enn Word. Langt fra så utbredt som Word, men bedre egnet til profesjonelle oppgaver.
  - *Filformat*: Proprietært filformat, men mulighet til å lagre innholdet som XML.
  - *Metadata*: Lagrer en definert mengde metadata i Adobes XMP-format. Elementene som kan lagres er Author, Title, Subject, Keywords, Copyright, Web Statement, Job reference og Marked. I tillegg lagres automatisk metadata som opprettings- og sist-endret-dato.

Felles for de to produksjonsproduktene er at de begge kan lagre metadata (utnyttes i kap. 4.2) og eksportere til et åpent filformat (utnyttes i kap. 4.3)

#### 2.3.1.2 Innholdsarkiv

Når dokumentet er opprettet må det lagres. Systemer for lagring av innhold kalles innholdsarkiver, og kan variere veldig i omfang og utførelse. Trivielle typer innholdsarkiver er filsystemer og databaser, men de fleste innholdsarkiver benytter disse (eller andre) som lager i bakrgrunnen og presenterer brukeren for et eget brukergrensesnitt.

Et Content Management System (CMS) er en type innholdsarkiv som i tillegg til lagring av innhold støtter prosessene rundt innholdet, for eksempel gjennom publisering og støtte for samforfatterskap ved hjelp av låser, kommentering og lignende. I slike systemer kan det legges opp faste arbeidsflyter og mekanismer som støtter disse.

Under beskrives en rekke innholdsarkiver/Content Management Systemer, med fokus på hvordan disse knyttes opp mot produksjonsverktøy, og hvilken støtte de har for omliggende prosesser.

- *BSCW – Basic Support for Cooperative Work*. [14]
  - *Generelt:* BSCW er et webbasert samarbeidsverktøy med støtte for dokumentopplasting, automatisk hendelsesunderretting, gruppehåndtering mer mer, utviklet på det tyske forskningssentret GMB.
  - *Arbeidsmiljø:* Grunnideen i BSCW er *delt arbeidsrom*. Arbeidsrommet er tilgjengelig gjennom weblesere, og bruker dermed HTTP-protokollen for å kommunisere med sitt interne innholdsarkiv. Innhold kan organiseres i mappestrukturer, og opplasting foregår gjennom web-skjema.
  - *Innholdsarkiv:* Internt. BSCW benytter et integrert innholdsarkiv på tjeneren, og har støtte for alle innholdstyper. BSCW gjør ingen konvertering av opplastet innhold og lagrer kun filen binært.
  - *Samarbeid:* Samtidighetskontroll håndteres ved hjelp av låser og versjonslagring, men systemet har fokus på asynkront samarbeid. Til dette kan kommentering og diskusjonsgrupper benyttes. Tilgangskontroll.
  - *Arbeidsflyt:* BSCW kan støtte arbeidsflyt gjennom å sende meldinger hver gang et innholdselements tilstand endres.
  - *Metadata:* Opererer med en definert mengde metadata, inkludert filnavn, størrelse, forfatter og lignende. Enkelte av disse kan endres av brukeren, men de fleste opprettes og oppdateres automatisk.
- *Microsoft SharePoint Portal Server* [15]
  - *Generelt:* MS SharePoint Portal Server er Microsofts samarbeids- og portaltjener. Brukere kan opprette egne samarbeidsportaler med integrerte innholdsarkiver og søkefunksjonalitet.
  - *Arbeidsmiljø:* Webbasert arbeidsmiljø integrert i tjeneren
  - *Samarbeid:* Tilbyr både låsing og versjonskontroll.
  - *Arbeidsflyt:* Støtter arbeidsflyt, både med ferdigdefinerte og egendefinerte flytmodeller.
  - *Metadata:* Mapper kan tilordnes ulike profiler, som sier hvilke egenskaper som er definert på dokumentene i denne mappen. Profiler kan også gis direkte på dokumenter.
- *Inspera Content Server* [16]

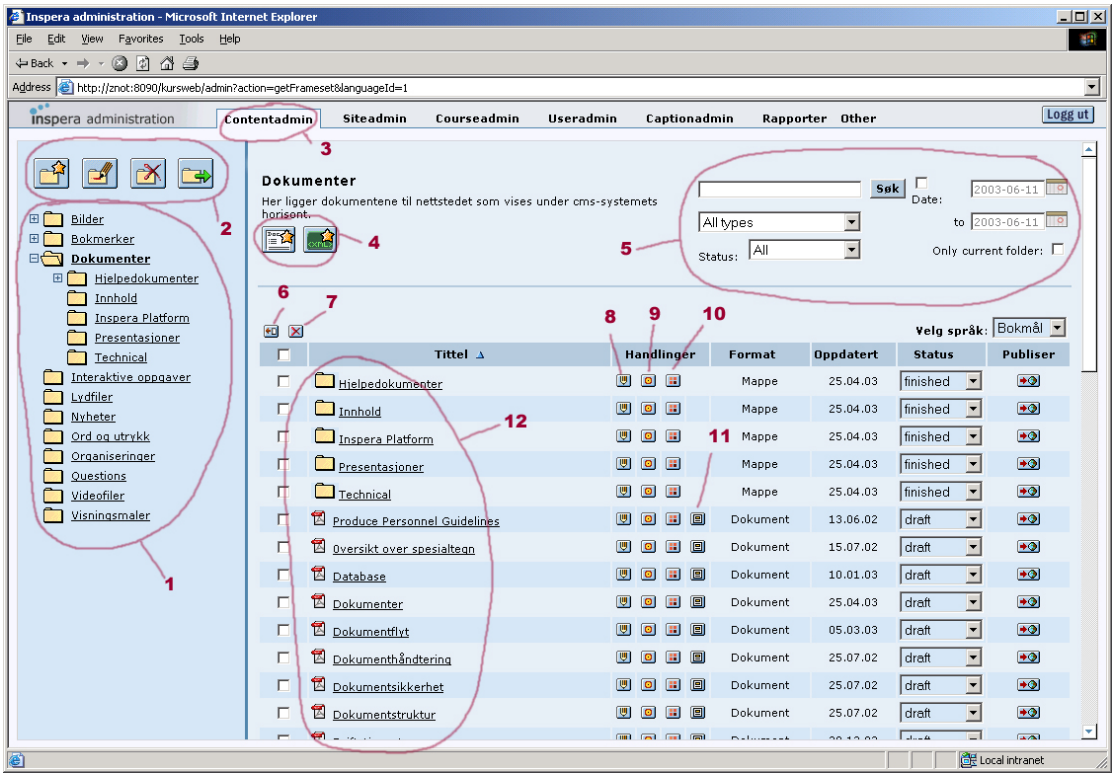
- *Generelt:* Inspera Content Server er et CMS utviklet av Inspera AS, inkludert web-publisering, arbeidsflyt, støtte for ulike innholdstyper med mer. Se kapittel 2.3.2.
- *Arbeidsmiljø:* Webbasert. Innhold kan struktureres i mapper, og filmanipulering som flytting, kopiering og sletting støttes i grensesnittet.
- *Innholdsarkiv:* Integrert innholdsarkiv som benytter Oracle 9i database. Innholdsarkivet støtter en definert mengde innholdstyper direkte (inkl. forskjellige XML-baserte dokumenttyper, lyd-/bilde-/video-formater) som håndteres spesielt og med egen semantikk. Ukjente innholdstyper håndteres som eksterne filer, og lagres uten videre prosessering.
- *Samarbeid:* Ingen låsing av innhold, men innhold lagres i versjoner slik at tapte oppdateringer ikke er tapte, bare ikke med i neste versjon. Dog overlates det til brukerne å oppdage at slik har hendt, og kombinere endringene.
- *Arbeidsflyt:* Støtter arbeidsflyt. Det kan defineres flytmodeller slik at brukerne får opp lister over aktuelle arbeidsflytsoppgaver. Disse oppdateres automatisk ettersom arbeidsflyten følges.
- *Metadata:* Innholdsarkivet inneholder en rekke ferdigdefinerte metadata for de forskjellige innholdstypene, i tillegg kan det defineres egne metadata. Egendefinert metadata kan være av forskjellige typer, for eksempel tekststrenger eller fra valg.

### 2.3.2 Inspera Content Server

Inspera Content Server er et web-basert Content Management System for flermedial publisering. Systemet har et webbasert grensesnitt og har støtte for blant annet arbeidsflyt, versjonskontroll, tilgangskontroll, tilordning av maler og brukerdefinerte innholdstyper. Av tjenester knyttet til innhold som CMS'et tilbyr kan nevnes:

- Oppretting og redigering av innhold: Innhold kan importeres eller opprettes direkte i systemet gjennom XML-koding eller bruk av en WYSIWYG-editor.
- Sentral lagring av alt innhold: Alt innhold lagres adskilt fra presentasjon i XML-strukturer. CMS'et bruker Oracle database til lagring av innhold. Innholdet kan organiseres i mappestrukturer.
- Assosiasjon: Innholdselementer kan knyttes til hverandre.
- Versjonskontroll: Systemet lagrer revisjoner og holder full dokumenthistorikk. Brukere kan se på eldre revisjoner og sette de som aktive.
- Bidragskontroll: Bidragsytere kan knyttes med ulike roller opp mot innhold, som for eksempel forfatter, fotograf, redaktør og lignende.
- Flerspråklighet: Et innholdselement kan opprettes i flere målformer og presenteres etter brukerens preferanse.
- Fleksibel metadatastruktur. Det kan defineres ulike metadatafelter for de forskjellige innholdstypene på ulike portaler.
- Kategorisering: Innhold kan kategoriseres etter ulike strukturer som emne og lignende.
- Søk i innhold og metadata: Systemet har innebygd søkefunksjon som søker både i innhold og i metadata.

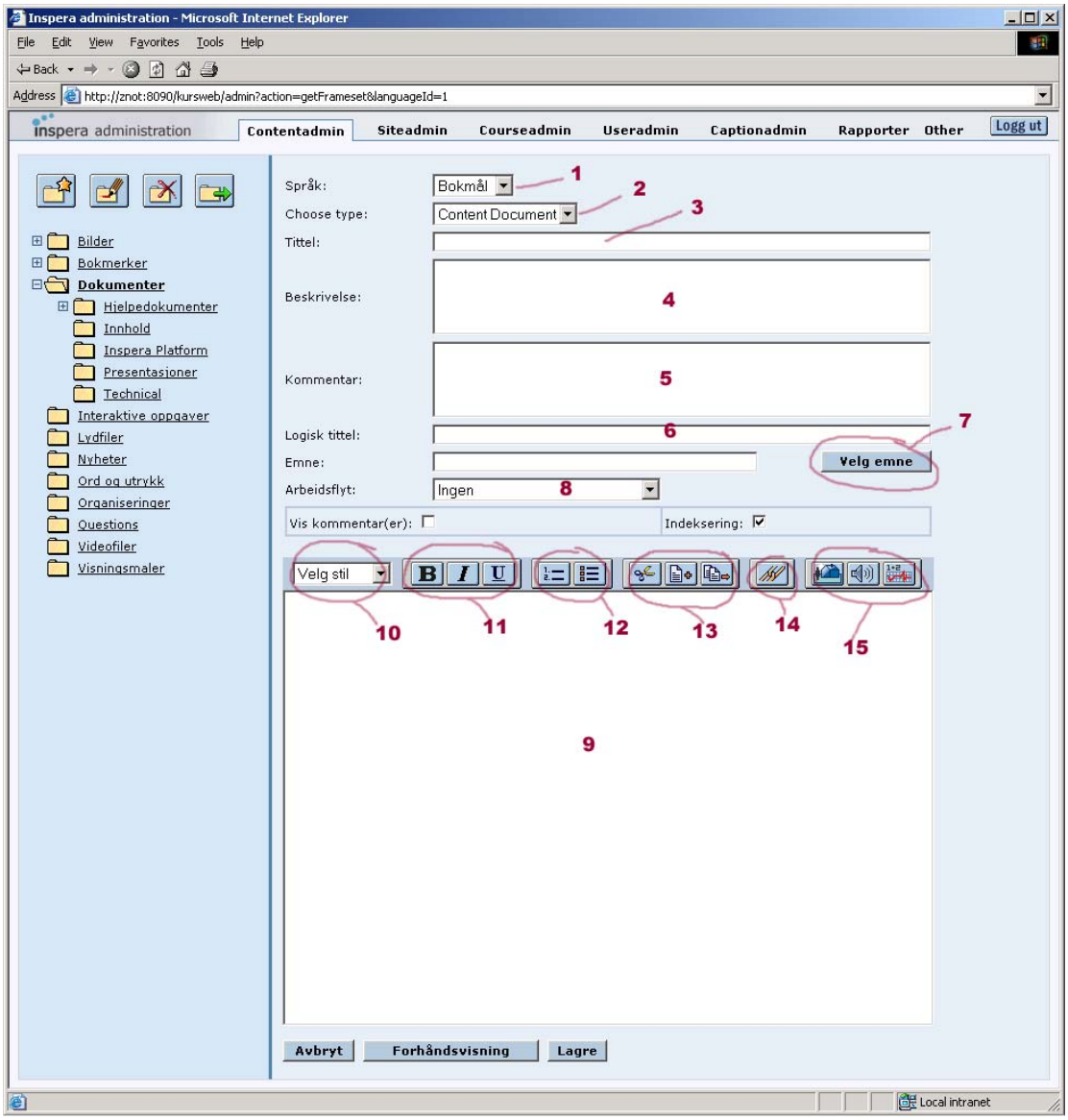
Figur 7 viser grensesnittet for Inspera Content Server, og forklares nærmere under.




**Figur 7: Inspera Content Server grensesnitt**

1. Mappestruktur. Viser mappene i innholdsarkivet, med åpen mappe uthevet.
2. Knapper for å opprette, redigere, slette eller flytte mappe.
3. Klaffene på toppen skifter mellom innholdsadministrering og de andre installerte systemene på Inspera Content Server, som nettpublisering, e-læring og lignende.
4. Opprett nytt dokument, ved WYSIWYG-editor eller opplasting av XML. Knappene her vil variere etter hva slags innhold mappen er registrert for å kunne inneholde. Mappen på figuren kan bare inneholde dokumenter.
5. Søk i innholdsarkivet.
6. Flytt merkede elementer til en annen mappe.
7. Slett merkede elementer.
8. Rediger innholdselement.
9. Se/endre egenskaper (metadata).
10. Dokumenthistorikk/revisjoner.
11. Tilordne visningsmal til innholdselement.
12. Innholdselementene i valgt mappe.

Oppretting av nye innholdselementer foregår enten ved import (for innholdstyper som lyd, bilde og video) eller ved oppretting ved hjelp av CMS'et (for dokumenter, men disse kan også importeres fra XML-filer). CMS'et tilbyr en WYSIWYG-editor ("What You See Is What You Get) for oppretting av nye dokumenter. Editoren støtter klipp-og-lim fra Word hvor stilene beholdes. Figur 8 viser oppretting av nye dokumenter med WYSIWYG-editoren, og forklares under.



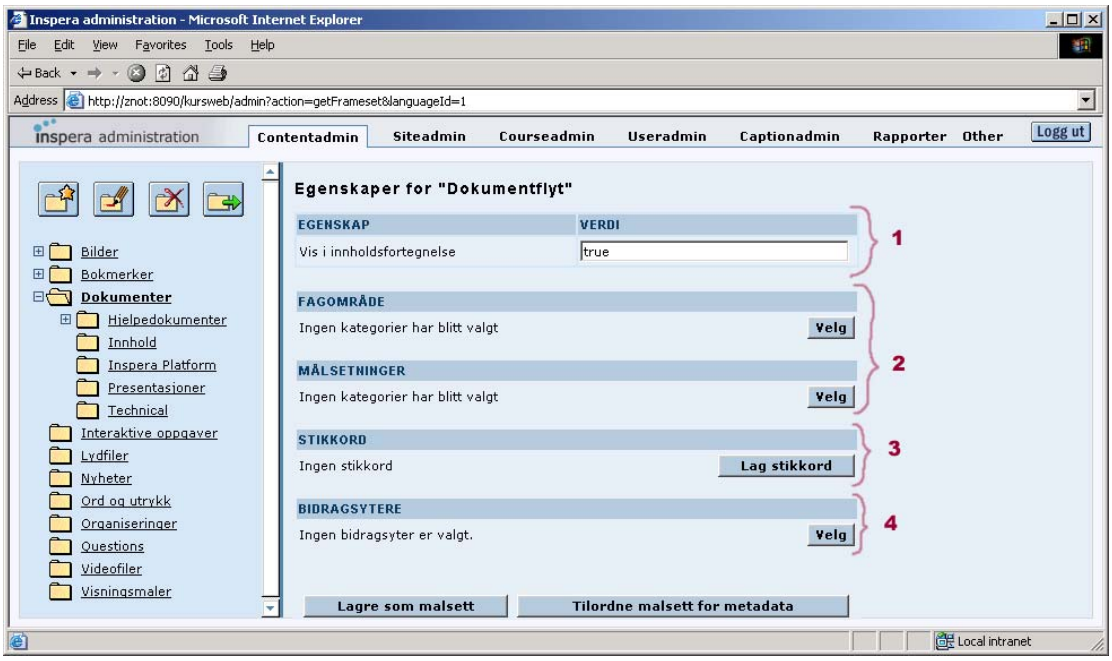

**Figur 8: Inspera WYSIWYG-editor**

1. Velg språk. Her kan man velge mellom alle språk som er definert på portalen.
2. Velg type. Det finnes undertyper av dokumenter, som for eksempel nyhetsartikler og lignende.
3. Tittel på dokumentet.
4. Beskrivelse.
5. En kommentar til denne revisjonen.
6. Logisk tittel benyttes til å entydig identifisere dokumentet.
7. Knappen ”Velg emne” bringer opp en trestruktur av Insperas emneklassifisering der emne kan velges.
8. I boksen ”Arbeidsflyt” kan man velge mellom ulike arbeidsflyter som er definert på portalen.
9. WYSIWYG-editor.
10. Stiler som kan benyttes er Normal og Overskrift 1-3.
11. Fet, kursiv og understrekning.
12. Punktliste og nummerert liste.
13. Klipp ut, lim inn og kopier.

14. Sett inn hyperlenke.
15. Sett inn bilde, lyd eller matematisk formel.

Ved redigering av dokumenter (punkt 8 i Figur 7) kommer samme grensesnitt opp, men alle felter og WYSIWYG-editoren er fylt ut. I tillegg er det en knapp for å se på/redigere XML-koden.

Metadata settes i et eget grensesnitt, vist i Figur 9. Innholdet i dette grensesnittet vil variere etter hva som er definert av metadata på portalen. Figuren viser at én egenskap er definert for innholdstypen og det finnes to ulike klassifiseringstrær på portalen. Stikkord og bidragsyter er felles for alle portaler.




**Figur 9: Inspera Content Server – egenskaper**

1. Egenskaper. Denne vil variere ettersom hvilke egenskaper som er definerte. Egenskaper kan være av ulike typer, på figuren er en enkel tekst-egenskap definert. I tillegg finnes binære (ja/nei) egenskaper og option/multiple choice.
2. En portal kan ha definert ulike klassifiseringstrær. Ved å trykke på knappen "Velg" kommer en visning av treet frem og man kan klassifisere dette innholdselementet etter ønske.
3. Det er en felles stikkordsliste som benyttes. Denne er tilgjengelig på alle definerte språk.
4. Det kan registreres bidragsytere på innholdselementet. Bidragsytere registreres med rolle, som for eksempel forfatter, redaktør og lignende.

### 2.3.3 Sømløs integrasjon

Grensesnittet som innholdsarkivet tilbyr brukeren kan kalles arbeidsmiljøet, siden det er her brukeren håndterer innholdet. I de fleste tilfeller vil arbeidsmiljøet som brukes for å lagre til innholdsarkivet være en integrert del av innholdsarkivet, i andre tilfeller benyttes en eller annen type mellomvare. Mellomvare er produkter som knytter et system til et annet. Eksempelvis kan en FTP-klient (og FTP-protokollen) være en mellomvare for et

produksjonsverktøy og en FTP-tjener som fungerer som innholdsarkiv. Utfra denne definisjonen kan også grensesnitt innebygd i innholdsarkivet tenkes på som mellomvare, siden de skjermer brukeren fra innholdsarkivets interne struktur.

I begrepet *sømløs integrasjon* ligger en tanke om å unngå mellomledd mellom produksjonsverktøy og innholdsarkiv, og gi brukeren mulighet til å aksessere innholdsarkivet direkte fra produksjonsverktøyet. Det er i denne sammenheng satt opp tre forutsetninger for sømløs integrasjon:

- **Sømløst integrert arbeidsmiljø.** Brukeren må ha mulighet til å navigere og manipulerere innholdsarkivet direkte.
- **Metadatamerking.** Metadatamerking foregår vanligvis under lagring til innholdsarkivet. Ved sømløs integrasjon må brukeren ha mulighet til å metadatamerke dokumentet i produksjonsverktøyet, og disse metadata må følge dokumentet inn i innholdsarkivet.
- **XML-integrasjon.** Som nevnt i introduksjonen til dette kapitlet må dokumenter konverteres til et internt format (i denne sammenheng XML) for at de skal kunne benyttes som innhold.

Disse tre punktene vil diskuteres for seg i påfølgende delkapitler. I tillegg til en beskrivelse vil det også nevnes hvordan prototypen beskrevet i rapporten løser dette punktet, og hva som vil være tekniske suksesskriterier.

#### 2.3.3.1 Sømløst integrert arbeidsmiljø

Arbeidsmiljøet består av produksjonsverktøyet, innholdsarkivet (CMS) og knytningen mellom disse. Ved sømløst integrert arbeidsmiljø menes et arbeidsmiljø der brukeren ikke trenger å forholde seg til ekstra steg eller mellomvare når dokumentet skal over fra produksjonsverktøyet til innholdsarkivet. I tillegg er det vanligvis ønskelig å kunne manipulere dokumentene i innholdsarkivet, for eksempel ved å organisere de i mapper, gi de nye navn og lignende. Dette anses også som en del av arbeidsmiljøet. Et fullstendig sømløst integrert arbeidsmiljø innebærer at brukeren skal kunne lagre dokumentet i innholdsarkivet og i tillegg gjøre overnevnte manipuleringer, direkte fra arbeidsverktøyet. En slik distansering gjør at brukere slipper å forholde seg til innholdsarkivet.

Prototypen implementerer WebDAV-protokollen (kapittel 3). WebDAV-protokollen er en måte for ulike innholdstjenere og klienter å samarbeide. Ved at Inspera Content Server støtter denne protokollen kan WebDAV-klienter aksessere innholdsarkivet direkte, og generere egne visninger av arkivet. Både Microsoft Windows, Microsoft Word og Adobe FrameMaker støtter WebDAV. Det betyr at disse får direkte tilgang til innholdsarkivet, og skaper en visning av dette på samme måte som lokale filstrukturer.

Det finnes ikke noe enkelt teknisk suksesskriterie for WebDAV-implementasjonen. Siden det kun er implementert prototypisk støtte er mulighetene begrenset i forhold til hva protokollen åpner for. Siden disse begrensninger ikke reflekteres i klienten kan det ikke forventes at alle muligheter klienten tilbyr vil håndteres eller utføres korrekt i tjeneren. Begrensninger og forenklinger som er gjort er beskrevet i kap. 4.1.2.5, og disse må tas til etterretning under evaluering. For brukertesten, som vil være den virkelige evalueringen av denne delen er det så langt som mulig gjort tilpasninger slik at brukerens prosedyrer kan støttes av systemet.

#### 2.3.3.2 Metadatamerking

I de fleste systemer utsettes metadatamerking til ved lagring. Dette på tross av at store mengder metadata er kjent tidligere i prosessen. Det er dermed ønskelig å distribuere metadatamerkingen utover prosessen, slik at hvert steg i prosessen kan merke opp metadata som fremkommer i denne prosessen. For eksempel vil forfattere merke opp tittel, forfatter og lignende felter, mens sekretær kan legge på rettighetsinformasjon og lignende. Når dokumentet til slutt skal lagres i innholdsarkivet har metadata akkumulert seg opp og det er kun metadata relatert til lagringen som trenger å legges til.

Prototypen importerer metadata fra Word- og FrameMaker-dokumenter. Metadata-ekstraheringen er begrenset til en kjent mengde metadata-elementer. Disse mappes til elementer i Inspera Content Server. Ekstrahering av metadata er beskrevet i kap. 4.2.

Det tekniske suksesskriteriet for dette punktet vil være at all kjent metadata ekstraheres fra Word- og FrameMaker-dokumenter, og mappes korrekt til metadatastrukturen benyttet i Inspera Content Server.

#### 2.3.3.3 XML-integrasjon

XML-integrasjon mellom arbeidsverktøy og Content Management System innebærer at dokumenter fra produksjonsverktøyet skal kunne importeres og eksporteres fra CMS'et uten tap av nødvendig informasjon. I praksis vil dette si at uavhengig av produksjonsverktøy vil dokumentet konverteres til XML (etter en gitt DTD) før lagring i innholdsarkivet, og tilsvarende kunne eksporteres til forskjellige filformater. Begrunnelsen for det er ønskelig med XML som lagringsformat i stedet for proprietære filformater er at XML lettere kan eksporteres til ulike formater og medier, for eksempel HTML for digital visning eller PDF for trykk. I denne sammenheng er vurderte arbeidsverktøy begrenset til Microsoft Word og Adobe FrameMaker.

Prototypen løser dette gjennom å implementere støtte for import og eksport av disse filformatene til Inspera Content Server, beskrevet i kap. 4.3.

Teknisk suksesskriterie for prototypen er at dokumenter av begrenset kompleksitet (basert på gitte stilmaler) skal kunne importeres til korrekt XML, og eksporteres til å være leselig for arbeidsverktøyet. Dette gjelder for Microsoft Word og Adobe Framemaker.

# 3 Sømløs integrasjon vha. WebDAV

Dette kapitlet vil beskrive en kommunikasjonsprotokoll som muliggjør direkte samarbeid mellom arbeidsverktøyet og innholdsarkivet. WebDAV er en protokoll laget for å åpne for skriving til WWW-tjenere over HTTP. Enhver web-tjener som implementerer protokollen kan snakke med enhver WebDAV-klient. Fordelen med dette er at ulike brukere kan benytte ulike WebDAV-klienter, etter smak og behov. I tillegg betyr dette at selv om en organisasjon bytter innholdsarkiv trenger ikke brukerne å bytte arbeidsmiljø, siden de samme WebDAV-klientene de tidligere har brukt også vil fungere med det nye innholdsarkivet, forutsatt at det støtter protokollen. I kapittel 3.1 beskrives WebDAV-prosjektet og protokollen.

Microsofts WebDAV-klient WebFolders er integrert i Internet Explorer og gir et Utforsker-aktig grensesnitt for å navigere innholdsarkivet, i tillegg støttes den direkte av applikasjoner som Microsoft Word. Dette kan utnyttes til å skape et sømløst integrert arbeidsmiljø. WebFolder er beskrevet i kap. 3.2. I kap. 3.3 beskrives alternative WebDAV-klienter, og disse sammenlignes med WebFolder.

## *3.1 WebDAV-protokollen*

WebDAV Distributed Authoring Protocol (heretter kalt WebDAV-protokollen) er en utvidelse av HTTP/1.1-protokollen som støtter distribuert, asynkron, samarbeidsorientert editering. Ved å fokusere på en protokoll kan WebDAV enklere integreres i eksisterende programvare, i stedet for å bli et nytt mellomledd mellom produsent og innholdsarkiv. Det finnes mange WebDAV-klienter, og flere store bedrifter har tatt imot WebDAV og integrert støtte for protokollen i deres produkter:

- *Microsoft* har integrert sin WebDAV-klient WebFolder i Windows 2000/XP, Internet Explorer 5 og Office 2000, og har innebygd støtte for WebDAV i sine tjenere Internet Information Server 5, SharePoint og Exchange Server.
- *Apache* har konstruert en modul (mod_dav, [17]) som gir WebDAV-støtte i Apache Server, og har i tillegg et eget prosjekt under Jakarta-paraplyen, Slide, som er en open-source javaimplementert WebDAV-tjener og -klient.
- *Adobe* har integrert WebDAV-støtte i nyere versjoner av alle sine produkter, inkludert Adobe FrameMaker 7.0, Adobe Illustrator 10.0 og Adobe Acrobat 5.0.
- *KDE* [18] er et grafisk brukergrensesnitt for UNIX arbeidsstasjoner som støtter WebDAV direkte.
- *Apple*s operativsystem MacOS X støtter WebDAV direkte, men støtten er rapportert å være noe ustabil. Goliath [19] er en egen WebDAV-klient for Mac som kan benyttes i stedet, og som støtter MacOS 8.1 og oppover.

Akkurat som en telefon er nær ubrukelig hvis kun noen få eier en, men uovertruffen når millioner eier en, vil WebDAVs nytte øke etter hvert som flere produkter blir kompatible med protokollen.

Hovedtanken bak WebDAV er samarbeid om dokumentproduksjon mot innholdsarkivet. Der samarbeid tidligere har bestått i å sende dokumenter frem og tilbake over epost kan dokumenter nå redigeres direkte over WWW. For å unngå konflikter og overskrivning har man vanligvis tydd til epost på for å ”sende stafettpinnen videre”, men

WebDAV tilbyr låsemekanismer for å hindre overskrivning og vise andre at dokumentet er under redigering. [20] beskriver noen av fordelene med WebDAV:

- Sømløs, interoperabel publisering av alle typer innhold til web ved bruk av HTTP.
- Distribuert samarbeid. Dokumenter kan fjernredigeres over nett, og låsemekanismer hindrer problemer som forbindes med samforfatterskap.
- Ingen begrensninger på innholdstype. WebDAV begrenses ikke til HTML eller XML, men støtter alle typer dokumenter, regneark, bilder, og alle andre formater.
- WebDAV og HTTP gir et felles grensesnitt til en variert mengde innholdsarkiver, for eksempel databaser, CMS'er, filsystemer og lignende. I praksis gjør WebDAV at man kan se på WWW som et nettverkstilgjengelig filsystem.

Det siste punktet er viktig for organisasjoner. Siden en felles protokoll gir interoperabilitet mellom ulike tjenere og klienter, vil organisasjonens innholdsarkiv (som støtter protokollen) øke i verdi uten nye investeringer siden den automatisk støtter nye WebDAV-klare arbeidsverktøy [21]. På samme måte kan brukere i organisasjonen, som har brukt verdifull tid til å bli komfortabel med sitt arbeidsverktøy, beholde arbeidsverktøyet selv om innholdsarkivet byttes ut.

### 3.1.1 Bakgrunn

WebDAV (World Wide Web Distributed Authoring and Versioning) oppsto som en ad hoc arbeidsgruppe i 1996, offisielt tatt inn i Internet Engineering Task Force (IETF) i mars 1997. Gruppens mål var å åpne Internet for fjernediting av dokumenter. Dette skulle gjøres gjennom en utvidelse av HTTP-protokollen, som brukes til kommunikasjon over web. De kom tidlig frem til nøkkelutvidelser som måtte støttes ([22]):

- **Metadata**. Mulighet for å opprette, fjerne og spørre etter informasjon om dokumenter (websider), som f.eks. forfatter, dato for oppretting, osv. I tillegg mulighet til å knytte sammen relaterte sider.
- **Håndtering av navnerom:** Mulighet til å kopiere og flytte dokumenter, og mulighet til å få en opplisting av sider på ett gitt nivå i hierarkiet (jmf. mappelistinger i et filsystem).
- **Overskrivingsbeskyttelse:** Mulighet til å hindre at mer enn én person jobber på et dokument om gangen, slik at ikke den andre skriver over den førstes endringer.
- **Versjonskontroll:** Mulighet til å lagre viktige revisjoner av et dokument for senere uthenting. Versjonskontroll kan også støtte samarbeid ved at to eller flere bidragsytere kan jobbe på samme dokument i parallelle baner.

Arbeidsgruppen har fokusert arbeidet på tre dokumenter: Et *scenarios document* [23] som beskriver typiske scenarier tilknyttet distribuert dokumenteditering, et *requirements document* (RFC 2291) [24] som beskriver og begrunner høy-nivå funksjonelle krav til et fjernediteringssystem, og en *protocol specification* (RFC 2518) [25] som beskriver en utvidelse av HTTP/1.1-protokollen med nye metoder og lignende for WebDAV. Sistnevnte beskrives i detalj i kap. 3.1.3, mens en oversikt over de funksjonelle kravene som ble funnet er gitt i kap. 3.1.2. WebDAV-gruppen har også vært opphav til to undergrupper som jobber med utvidelser av WebDAV, nemlig DAV Searching and

Locating (DASL), som jobber med søkemuligheter for DAV[3], Web Versioning and Configuration Management (Delta-V), som fullfører jobben WebDAV ikke fikk fullført, nemlig versjonskontroll, og WebDAV Access Control (ACL), en undergruppe av WebDAV som jobber med tilgangsstyring og autentisering.

### 3.1.2 Funksjonelle krav

WebDAV-gruppen ferdigstilte i februar 1998 sin kravspesifikasjon til en distribuert editerings- og versjonskontrollprotokoll, "Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web", påbegynt juni 1996. I dette dokumentet beskrives funksjonaliteten som vil muliggjøre distribuert åpning, editering og publisering av ulike medietyper over nettet. Det unnlates å gi noen forslag til implementering, men de funksjonelle kravene er gitt med begrunnelser. WebDAV-protokollen spesifisert i [25] og beskrevet i kap 3.1.3 er WebDAV-gruppens forsøk på å tilfredsstille disse kravene.

De viktigste punktene som blir beskrevet i dokumentet er:

- *Lik støtte for alle innholdstyper.* En protokoll som kun støtter gitte dokumenttyper er av begrenset nytteverdi. Å fokusere på enkelte typer (for eksempel HTML) er også problematisk fordi slike typer er i stadig utvikling. Ved å støtte alle innholdselementer likt mister man noe spesialisert funksjonalitet (for eksempel sammenkjøring (merging) av dokumenter), men man sørger for støtte for alle nåværende og kommende innholdstyper.
- *Støtte for samtidighetskontroll.* En kjent problematikk fra databasedomenet er det såkalte "lost update problem", der en brukers endringer blir overskrevet av en annens. En samarbeidsprotokoll må dermed inneholde mekanismer som hindrer slike problemer. Dette kan gjøres på flere måter, enten ved avansert sammenkjøring av endrede dokumenter, eller ved å låse dokumenter for å hindre samtidig endring.
- *Støtte for metadata.* Ofte ligger en stor del av informasjonen om dokumenter (metadata) ikke lagret direkte på dokumentene. Eksempler kan være forfatter, nøkkelord, rettighetsinformasjon og lignende. Dermed må protokollen inneholde funksjonalitet for å lage, endre, lese og slette vilkårlig metadata på alle dokumenttyper.
- *Støtte for type-uavhengige lenker.* Ressurser kan være relaterte på forskjellige måter. Det er nyttig å kunne modellere disse relasjonene.
- *Uthenting av uprosessert kilde.* For websider er det ofte en forskjell mellom det som hentes frem ved en forespørsel, og den lagrede kilden. Eksempler kan være HTML med *server-side includes*, CGI-skript, Java Server Pages, XSL-transformert XML og lignende. For editering er det dermed ikke nok å gjøre en vanlig forespørsel for å hente dokumentet, siden det er kilden man vil endre.
- *Manipulasjon av navnerom.* Med manipulasjon av navnerom forstås støtte for operasjoner som kopiering, flytting og lignende av ressurser. Dokumenter lever i et navnerom, og protokollen må ha muligheter til å manipulere dette.
- *Støtte for samlinger.* Samlinger er nyttige for å gruppere dokumenter med sammenfallende egenskaper, for dermed å gjøre korpuset mer oversiktlig for de som jobber med det, og for navigering gjennom store korpus.

[3] Denne gruppen ble opprettet i 1998 og opprettet en rekke protokollutkast, men ble etter hvert avsluttet på grunn av manglende fremgang. Dog finnes det implementasjoner av eksisterende utkast, og det er stadig interesse i å fullføre spesifikasjonen.

### 3.1.3 Protokoll

WebDAV-protokollen er en foreslått standard, og er ute for høring fra IETF. Protokollen slik den står har støtte for tre hovedpunkter:

- *Samtidighetskontroll*
- *Egenskaper (metadata)*
- *Navneromsmanipulering*

Disse blir gjennomgått i følgende kapitler. Ytterligere tre punkter er på trappene gjennom relaterte prosjekter:

- *Versjonshåndtering.* (Delta-V) Lagring av viktige revisjoner for senere gjenfinning.
- *Avanserte samlinger.* (WebDAV Advanced Collections) Støtte for sorterte samlinger, samt samlinger som inneholder referanser.
- *Tilgangsstyring.* (WebDAV ACL) Autentisering og rettighetskontroll på ressurser.

Protokollen er en utvidelse av HTTP/1.1-standarden [26]. En introduksjon til denne standarden er gitt i kap. 3.1.3.1.

#### 3.1.3.1 Grunnleggende HTTP

HTTP-protokollen er de facto standard for overføring av dokumenter over World Wide Web, vanligvis vha. TCP/IP. Modellen for kommunikasjon over HTTP er en klient-tjener-modell, det vil si at en HTTP-klient sender en forespørsel (*request*) til en HTTP-tjener som så returnerer et svar (*response*).

Formatet for både forespørsler og svar er som følger:

- Startlinje, ulik for forespørsel og svar
- Null eller flere linjer med meldingshoder (*headers*)
- En tom linje
- Muligens en meldingskropp

Startlinjen for forespørsler består av en metode, en URI-adresse til en ressurs, og HTTP-versjonen som brukes. Eksempel:

```
GET /~johnarne/index.html HTTP/1.1
```

Metoden GET betegner en forespørsel om å hente ressursen adressert med den gitte URI'en. Andre metoder spesifisert i HTTP/1.1 er HEAD, som fungerer som en GET bortsett fra at tjeneren ikke returnerer ressursen, kun meldingshodene. Dette er nyttig for å sjekke tilstanden til en tjener/ressurs. POST er en metode som brukes for å sende data til tjeneren for prosessering. Til forskjell fra GET inneholder POST-forespørsler ofte data i meldingskroppen, i tillegg er ofte den adresserte ressursen et program eller lignende som skal prosessere denne data. Vanligste bruk for denne metoden er utvilsomt å sende informasjon i HTML-skjemaer til CGI-, PHP- eller JSP-skript. POST-metoden er løst spesifisert i HTTP-spesifikasjonen, noe som gjør at den veldig ofte brukes til utvidninger av HTTP.

Startlinjen i et svar er statuslinjen, og inneholder en melding fra tjeneren om resultatet av forespørselen. Også i tre deler, består statuslinjen av HTTP-versjon, en tresifret statuskode og en engelsk beskrivelse av statuskoden. Eksempel:

```
HTTP/1.1 404 Not Found
```

Statuskodene deles i fem grupper: 1xx for informasjonsmeldinger, 2xx for suksess, 2xx for videresendinger, 4xx for klientfeil, 5xx for tjenerfeil. De vanligste statuskodene er 200 (alt ok, ressursen returneres i meldingskroppen), 404 (ressursen ble ikke funnet), 301 (ressursen er midlertidig flyttet), 302 (ressursen er permanent flyttet), 303 (se annen ressurs, indikerer ofte at ressursen er flyttet, og Location:-meldingshodet gir ny adresse), 500 (uventet tjenerfeil).

Meldingshoder inneholder informasjon om forespørselen eller svaret, eller om innholdet av meldingskroppen, og er på formatet *Navn: Verdi*. Eksempler:

```
Host: www.idi.ntnu.no
Content-Type: text/html
Last-Modified: Tue, 25 Feb 2003 17:24:59 CET
```

Navnet på meldingshodet er ikke følsomt for store/små bokstaver, men verdien kan være det. HTTP/1.0 spesifiserer 16 meldingshoder, hvorav ingen er obligatoriske, mens HTTP/1.1 spesifiserer 46 der ett er obligatorisk i forespørsler (Host:, gir navnet på tjeneren).

Meldingskroppen inneholder den returnerte ressursen i svarmeldinger, eller mer informasjon om forespørsler (for eksempel innholdet av et HTML-skjema). Vanligvis beskrives innholdet av meldingskroppen av ett eller flere meldingshoder, for eksempel Content-Type, som gir medietypen til innholdet (image/gif, text/html og lignende) og Content-Length, som gir lengden av innholdet i bytes.

Eksempel på HTTP-kommunikasjon:
Klient (sendes til www.idi.ntnu.no, på port 80):

```
GET / HTTP/1.1
Host: www.idi.ntnu.no
```

Klienten ber her om rot-filen på www.idi.ntnu.no, og spesifiserer at den benytter HTTP/1.1. Siden HTTP/1.1 sier at meldingshodet Host: er obligatorisk, sendes dette også med.

Svar fra tjener:

```
HTTP/1.1 200 OK
Date: Tue, 25 Feb 2003 16:39:20 GMT
Server: Apache/1.3.26 (Unix) PHP/4.0.6
X-Powered-By: PHP/4.0.6
Transfer-Encoding: chunked
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
```
… og ytterligere HTML-kode i meldingskroppen

Tjeneren svarer med statuskode 200, som betyr at ressursen returneres uten problemer. Videre meldingshoder opplyser om når ressursen er sist endret (da dette er en dynamisk ressurs er dette tidspunktet for HTML-generering), gir teknisk informasjon om tjeneren og sendingskoding, og at innholdet i meldingskroppen er av typen text/html.

HTTP/1.1 spesifiserer fire andre sjeldent brukte metoder i tillegg til GET, POST og HEAD. Disse er OPTIONS, PUT, DELETE og TRACE. OPTIONS er en forespørsel om informasjon om den adresserte ressursen (eller tjeneren). I tillegg skal eventuelle mellomliggende proxy-tjenere endre dette svaret til å reflektere begrensninger hos dem. PUT er ment brukt til å lagre ressursen inneholdt i meldingskroppen til adressen spesifisert, og overskrive det som eventuelt ligger der. Forskjellen mellom PUT og POST er at for POST spesifiserer adressen hvilken ressurs som skal håndtere dataen i meldingskroppen, mens for PUT er adressen der innholdet i meldingskroppen skal lagres. DELETE er ment brukt til å slette ressurser (eller flytte de til en utilgjengelig lokalitet). TRACE er ment brukt til diagnostisering, og returnerer forespørselen tilbake i meldingskroppen.

#### 3.1.3.2 WebDAV utvidelser av HTTP/1.1

WebDAV-protokollen utvider HTTP/1.1-protokollen med både nye metoder og ny funksjonalitet i de eksisterende metodene. De nye metodene kan deles i tre kategorier:

- Samtidighetskontroll: LOCK og UNLOCK, beskrevet i kapittel 3.1.3.3
- Egenskapshåndtering: PROPFIND og PROPPATCH, beskrevet i kapittel 3.1.3.4
- Navneromsmanipulering: COPY, MOVE og MKCOL, beskrevet i kapittel 3.1.3.5

Både disse metodene og de eksisterende har utvidet funksjonalitet ved hjelp av spesifiseringer i meldingskroppen på forespørslene. Disse spesialiseringene kunne vært gjort ved hjelp av meldingshoder, men WebDAV har valgt å benytte XML i meldingskroppen da denne er utvidbar og mer oversiktlig. Ved hjelp av XML-tagger spesifiseres funksjonaliteten til metodene utover metodenavnet.

Siden WebDAV-forespørsler kan gjelde mer enn én ressurs, en samling ressurser for eksempel, er ikke den enslige statuslinjen i svarmeldingen tilstrekkelig. WebDAV-protokollen innfører dermed en ny statuskode (207, Multi-Status) som betyr at dette svaret inneholder statusinformasjon i meldingskroppen. På denne måten kan hver ressurs som er affektert av forespørselen returnere sin egen statuskode.

#### 3.1.3.3 Samtidighetskontroll

Samtidighetskontroll er en viktig del av et editeringssystem med flere bidragsytere. Uten samtidighetskontroll ville brukere risikere å skrive over hverandres endringer. Hvis en annen bruker har lagret endringer i tiden mellom den første åpner filen og lagrer sine endringer, vil den andres endringer ikke være med i filen den første åpner og dermed ikke være med i filen hun lagrer. Dette er det som er kjent som ”lost update”-problematikken.

En løsning på dette er ved å låse filer[4]. Hvis en bruker som skal endre en fil låser den idet hun åpner den, sørger hun for at ingen andre kan gjøre endringer på den i mellomtiden og dermed miste endringene sine når hun lagrer. Dette kalles en *exclusive write lock*. Hvis en gruppe bidragsytere er i stand til å koordinere sine endringer slik at de unngår å overskrive hverandres endringer, kan de heller benytte en *shared write lock*.

| Eksisterende låser | Forespørsel om delt lås | Forespørsel om eksklusiv lås |
|---|---|---|
| Ingen | Innvilget | Innvilget |
| Delt lås | Innvilget | Nektet |
| Eksklusiv lås | Nektet | Nektet |

**Tabell 1: Innvilging av låser**

Siden HTTP-protokollen ikke beholder oppkoblinger lengre enn for enforespørsel-svar-runde, er WebDAVs låsemekanisme oppkoblingsuavhengig. Låsing foregår ved at en bruker ber om en lås på en ressurs (eller en samling), og hvis hun får den (se Tabell 1) returneres en såkalt *Lock Token*, en identifikator som sier at hun har en lås på objektet. På denne måten slipper hun å beholde en konstant oppkobling. Men denne identifikatoren er i seg selv ikke nok til å få tilgang til en låst ressurs. Tvert om er denne identifikatoren åpent tilgjengelig for alle. En property på ressurser viser hvem som har innvilget lås på denne ressursen, og tilhørende låsidentifikator. For å få tilgang på en låst ressurs må man dermed vise denne identifikator (dvs. sende den med som meldingshode) og i tillegg korrekt autentiseres som brukeren som fikk innvilget låsen. Hvordan autentisering foregår avhenger av tjeneren, men HTTP spesifiserer en metode (HTTP Digest Authentication [27]) basert på brukernavn og kryptert passord. Grunnen til at det ikke er nok å autentifiseres som korrekt bruker er for å sørge for at alle brukerens programmer er oppmerksomme på låsen, hvilket de gjør ved å sende med låsidentifikatoren.

Støtte for låsing gis gjennom to nye HTTP-metoder: LOCK og UNLOCK. For LOCK-metoden spesifiseres i meldingskroppen hvilken type lås som ønskes (delt/eksklusiv) og om låsen skal gjelde kun ressursen, eller eventuelle inneholdte ressurser (hvis ressursen er en mappe). For UNLOCK, i likhet med alle andre forespørsler på låste ressurser, må låsidentifikatoren være med som meldingshode.

I tillegg til disse to metodene har ressurser to egenskaper (properties, se kapittel 3.1.3.4 i forbindelse med låsing; *lockdiscovery*, som gir info om hvilke låser som er på ressursen for øyeblikket, og *supportedlocks*, som opplyser om hvilke låstyper ressursen støtter.

### 3.1.3.4 Egenskaper

Ressurser har mye informasjon tilknyttet seg som ofte ikke er lagret direkte på ressursen. Eksempler kan være forfatter, emne, rettighetsinformasjon og lignende. Slik informasjon kalles metadata, og er svært nyttig i informasjonsgjenfinning og søking.

WebDAV støtter metadata gjennom *properties*, egenskaper. Egenskaper er navn-verdi-par tilknyttet en ressurs, og er definert som XML-elementer. Navnet er en URI, og verdien er

[4] Et alternativ til dette er fingranulær sammenføyning av endringer, der man benytter seg av kunnskap om dokumentformatet til å kombinere flere brukeres endringer. Her gir man avkall på uavhengighet av ressurstyper. Det er en avveining av dette mot ubekvemmeligheten med låste dokumenter. WebDAV ønsker uavhengighet på ressurstyper på grunn av mengden ressurstyper som forekommer på WWW.

XML. Protokollen spesifiserer 11 egenskaper i DAV-navnerommet, men egenskapsmodellen er utvidbar med vilkårlige navn, og siden navnet er en URI sørges det for å unngå tvetydigheter. Av de 11 egenskapene som spesifiseres i WebDAV-protokollen er 5 (get*-egenskapene) tilsvarende meldingshoder som returneres ved en GET-forespørsel, for eksempel er verdien til getcontentlanguage lik verdien av meldingsholdet Content-Language ved en GET-forespørsel på ressursen. Egenskapene som er spesifiserte i DAV-navnerommet er:

- *creationdate* – tid og dato ressursen ble skapt.
- *displayname* – navn som er egnet for visning til sluttbruker
- *getcontentlanguage* – språket ressursen er på (for eksempel no_NO, no_NY, en_US)
- *getcontentlength* – størrelsen på ressursen i bytes.
- *getcontenttype* – MIME-typen til ressursen (for eksempel image/gif, text/html)
- *getetag* – en entitetsidentifikakor for ressursen
- *getlastmodified* – tid og dato for siste endring på ressursen
- *lockdiscovery* – oversikt over låser som er innvilget på ressursen
- *resourcetype* – hvilken type ressurs dette er
- *source* – en lenke til den uprosesserte kilden til denne ressursen
- *supportedlock* – informasjon om hvilke låstyper denne ressursen støtter

Protokollen deler egenskaper i to typer: *live* og *dead* egenskaper. *Live* egenskaper har en semantikk og syntaks som er kjent av tjeneren, og tjeneren sørger for at disse blir opprettholdt. Enkelte *live* properties oppdateres automatisk av tjeneren og kan ikke endres av klienter. Eksempler kan være DAV:getcontentlength og DAV:creationdate. Andre *live* egenskaper kan settes av klienter, men tjeneren kontrollerer syntaks på disse før de godtas, eksempelvis DAV:source der tjeneren kan sjekke at dette er en gyldig URI. *Dead* egenskaper settes av brukere, og beholder denne verdien til den blir satt til en annen. Det er ingen syntaks eller semantisk kontroll på disse fra tjenerens side.

For håndtering av egenskaper spesifiserer WebDAV-protokollen to metoder: PROPFIND og PROPPATCH. PROPFIND benyttes til å oppdage egenskaper ved en ressurs, og har tre bruksområder, som spesifiseres ved XML-tagger i meldingskroppen. Man kan be om å få alle egenskaper som er definert på ressursen (*allprop*, også standard ved tom meldingskropp), eller man kan få en liste av navnene på alle egenskapene som er definerte (*propnames*), eller spesifisere hvilke egenskaper som skal finnes (ved en eller flere *prop*-tagger). PROPPATCH benyttes for å sette eller fjerne egenskaper på en ressurs. En PROPPATCH-forespørsel er en atomisk operasjon, det vil si at enten gjennomføres alle endringene, eller ingen. Dette er for å hindre at en ressurs blir satt i en inkonsistent tilstand. Hvis en feil hindrer at egenskapene blir endret, blir klienten gjort oppmerksom på hvilken endring som medførte feilen gjennom et multistatus-svar.

#### 3.1.3.5 Navneromsmanipulering

Manipulering av navnerom kan ses på som innholdsarkiv-varianten av et filsystem. Selv om det ikke skulle være noen teknisk nødvendighet å gruppere dokumenter i mapper, gjør det samlingen meget mer oversiktlig og håndterlig for brukere. De vante oppgavene fra et filsystem er gjeldende også her; det er nyttig å kunne kopiere en ressurs, og flytte/gi ressurser nye navn. Standard samlinger i WebDAV kan inneholde ressurser og samlinger (som egentlig også er en ressurs). Et medlem i samlingen må ha en URI (adresse) som er lik samlingens pluss et tillegg (navnet på ressursen). Det betyr at hvis det finnes en

ressurs med adresse http://foo.com/bar/blah må http://foo.com/bar/ være en samling og inneholde http://foo.com/bar/blah som et medlem.

Innføringen av samlinger gjør at semantikken for standard HTTP-metoder må endres, for å ta høyde for at den adresserte ressursen er en samling. For GET og HEAD er semantikken uendret, siden disse er definert løst som å "retrieve whatever information (in the form of an entity) is identified by the Request-URI" [26], og dermed kan vurderes av tjeneren hva som skal returneres (et standarddokument, en liste over innholdet i samlingen, eller noe annet). Samme gjelder POST-metoden, som er så løst definert i HTTP/1.1 at det ikke gir mening å spesifisere den nærmere. DELETE for samlinger gjelder rekursivt for samlingen og alle dens medlemmer. PUT er ikke definert for samlinger, da det skiller seg fra semantikken definert for metoden. Det er heller definert en ny metode (MKCOL) for å opprette samlinger.

Nye metoder for navneromsmanipulering innført i WebDAV er MKCOL, COPY og MOVE. MKCOL benyttes for å opprette en samling, og oppretter en ny samling på adressen gitt i forespørselen, forutsatt at samlingen over eksisterer. COPY kopierer en ressurs til en ny adresse. Siden mulighetene på den nye adressen kan være annerledes enn på originalen (for eksempel hvis originalen er et cgi-skript) er det ikke sikkert at de to kopiene oppfører seg likt. Endringer på kopien vil ikke affektere originalen. I tillegg er det mulig at ikke alle *live* egenskaper lar seg duplisere på den nye lokaliteten som *live*. Disse må da kopieres som *dead* egenskaper. Det er mulig å spesifisere i forespørselen om metoden skal feile hvis ikke *live* egenskaper kan reproduseres på den nye adressen. Meldingshodet Overwrite brukes for å angi hvordan metoden skal håndtere en eventuell eksisterende ressurs på måladressen. Et annet meldingshode, Depth, brukes for å angi om innholdet i samlinger skal kopieres.

MOVE fungerer som en *logisk* ekvivalent av COPY etterfulgt av DELETE. Merk at den ikke i praksis gjør disse to operasjonene. For eksempel vil man ved en COPY forvente at en unik indikator endres på kopien, mens dette ikke vil være tilfelle ved MOVE. På samme måte som for COPY brukes meldingshodene Overwrite og Depth for å angi funksjonalitet. Hvis en feil oppstår under flytting av et tre (en hierarkisk mappestruktur) skal operasjonen ikke flytte noen ressurser under det subtreet som feilet (da dette vil føre til et inkonsistent navnerom), men likevel fortsette operasjonen for andre subtrær.

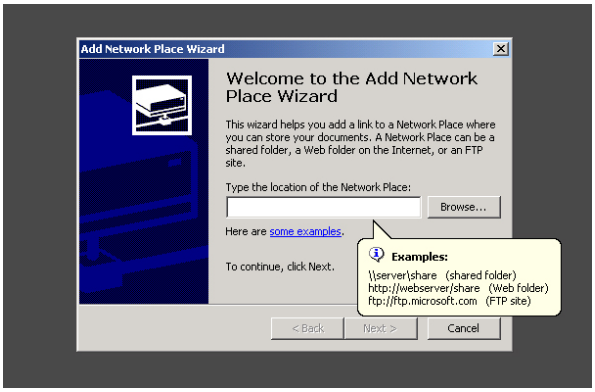
## 3.2 Microsoft WebFolders

Microsoft WebFolder er en utvidelse av Internet Explorer som støtter både WebDAV og Microsofts publiseringsprotokoll Web Extender Client (WEC), og følger med Internet Explorer 5 og oppover og tilsvarende Windows-versjoner (ME, 2000 og XP). Den integreres med Internet Explorer og gjør det mulig å navigere WebDAV-kompatible tjenere i et grensesnitt som tilsvarer Windows Utforsker, slik at mapper på en WebDAV-tjener for brukeren oppleves som om de var vanlige lokale mapper, eller en delt mappe på en Windows-tjener. WebFolders kan også integreres direkte med applikasjoner, som for eksempel i Office. Her brukes WebFolders-funksjonaliteten til å gi tilgang til WebDAV-tjenere direkte fra applikasjonen.

Dette kapitlet vil gi en introduksjon til bruk av Microsoft WebFolder (3.2.1) og hvordan den forholder seg til WebDAV-spesifikasjonen (3.2.2),

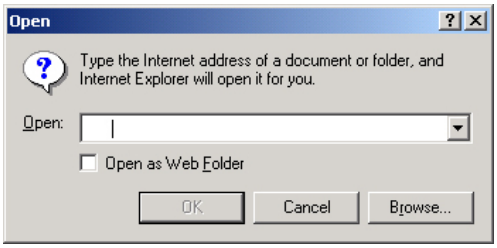
### 3.2.1 Bruk av WebFolder

Tilgang til en WebFolder kan gjøres på to forskjellige måter, enten gjennom å sette opp en WebFolder-snarvei eller ved å åpne WebFolderen i Internet Explorer. Oppsett av snarvei gjøres gjennom en veiviser i My Nework Places/Mine nettverkssteder (Figur 10). Denne resulterer i en snarvei som åpner den oppgitte WebDAV-adressen i Internet Explorer.




**Figur 10: Veiviser for å lage snarvei til WebFolder**

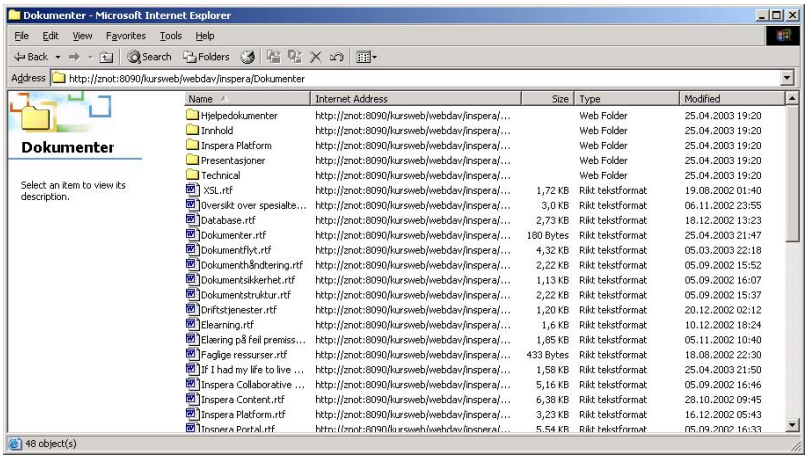
Man kan også åpne WebDAV-adresser direkte i Internet Explorer, ved å benytte seg av menyvalget Fil->Åpne. Her finnes en avhukningsboks (Figur 11) som åpner adressen som en WebFolder (dvs. ved bruk av WebDAV).




**Figur 11: Åpne WebFolder i Explorer**

Grunnen til at man ikke kan åpne WebFoldere direkte ved å skrive adressen inn i adressefeltet i Internet Explorer er at det ikke er mulig å på forhånd avgjøre om en gitt adresse er til en WebFolder, altså en WebDAV-ressurs, eller en vanlig web-ressurs. URI-en gir ingen indikasjoner på dette, ingenting skiller en WebDAV-adresse fra en web-adresse. I [27] beskrives alternativene for å løse dette utvikleren av Microsoft WebFolders utforsket, og hvorfor valget med Fil->Åpne ble gjort. Hovedtrekkene er at 99,9 % av forespørslene benytter ikke WebDAV, dermed må det være et alternativ. Og siden det så sjelden benyttes bør det ikke forvirre brukere. Dermed ble avgjørelsen tatt om å legge en avkrysningsboks på Fil->Åpne, siden ingen benytter dette menyvalget. Dersom denne er avkrysset vil Explorer forsøke å sende en OPTIONS-forespørsel til tjeneren, og hvis tjeneren returnerer et DAV-meldingshode er det bekreftet at tjeneren er WebDAV-kompatibel og adressen kan hentes som en WebFolder. Ved oppretting av WebFolder-snarveier slipper man denne ekstra forespørselen da sjekk etter WebDAV kun vil gjøres da snarveien opprettes.

Når en WebFolder åpnes i Internet Explorer vises et grensesnitt tilsvarende Windows Utforsker (Figur 12).




**Figur 12: Explorers WebFolder-grensesnitt**

På samme måte som i Windows Utforsker kan man her åpne, kopiere, flytte og opprette både filer og mapper. Samme hurtigtaster som fungerer i Utforsker fungerer også her, i likhet med dra-og-slipp-funksjonalitet for kopiering og flytting. Men enkelte ulikheter finnes. Muligheten til å opprette nye filer (Fil->Ny eller høyreklikk->Ny) inneholder kun mulighet til å opprette nye mapper. I tillegg kan ikke generelle filer åpnes direkte fra WebFolderen. Filer som kan vises i Explorer (JPEG, GIF) og filer som gjenkjennes av Office (RTF, DOC, XLS, PPT) lar seg åpne ved dobbelklikk eller høyreklikk->Åpne. Hvorvidt andre applikasjoner enn Office har denne støtten er ikke kjent. Andre filer må kopieres til en lokal mappe før de kan åpnes. I tillegg er det observert at Explorer cacher resultatet av WebDAV-forespørsler. Dette kan gi seg utslag i at en mappe som en bruker ser ikke er konsistent med innholdsarkivet. Det er antatt at det er satt en timeout for cachingen, men denne er ikke kjent. Mapper kan tvinges til oppdatering ved menyvalget Refresh/Oppdater (hurtigtast F5).

I Windows Utforsker kan man se en fils egenskaper. Eksempel er navn, sist endret, størrelse og lignende. Dette er også med i WebFolder. Her er egenskapene properties som er definert på ressursene. WebFolder viser ikke alle properties som er definert på en ressurs, kun en begrenset mengde. Tabell 4 viser hvilke properties som hentes, hva de svarer til i WebFolder-grensesnittet, beskrivelse og om de er definerte i protokollen. Forklaringen til mappe-egenskapene (ishidden, iscollection, isstructureddocument og defaultdocument) er hentet fra [29], et utkast til ekstra mappeegenskaper skrevet av Microsoft, resten er hentet fra Microsoft Developer Network Library [30]. Som vist hentes ikke alle standard DAV:-properties ut (mangler source, getetag, lockdiscovery og supportedlock), i tillegg har Microsoft definert egne properties under DAV:-navnerommet, noe som ikke skal gjøres. Egendefinerte properties skal lages i egendefinerte navnerom, men alle egenskaper nevnt i tabellen er i DAV:-navnerommet. Underlig nok ser det ikke ut til at klienten benytter seg av disse ekstra egenskapene, bortsett fra ishidden og lastaccessed. På MSDN er også flere nye DAV:-egenskaper beskrevet, men da de ikke hentes av klienten sees de bort fra her.

| Navn på egenskap | Element i WebFolder | Beskrivelse | WebDAV |
|---|---|---|---|
| name | | Ukjent. Ikke omtalt på MSDN eller i [29] | N |
| parentname | | Navn på mappen dokumentet ligger i. | N |
| href | Vises i listen som "Internett-adresse" | Adressen til ressursen. Det er ukjent hvorfor MS definerer denne egenskapen da adressen er gitt som et eget element (href) for ressursen. | N |
| ishidden | Filen skjules | Hvorvidt filen er skjult, tilsvarende vanlig Windows skjulte filer. | N |
| isreadonly | | Indikerer om ressursen kan slettes eller endres. | N |
| getcontenttype | | Se kap. 3.1.3.4. | J |
| contentclass | | Ressursens "content class", tilsynelatende Exchange-spesifik egenskap. | N |
| getcontentlanguage | | Se kap. 3.1.3.4. | J |
| creationdate | Vises under filegenskaper | Se kap. 3.1.3.4. | J |
| lastaccessed | Vises under filegenskaper | Siste aksess på filen. | N |
| getlastmodified | Vises under filegenskaper | Se kap. 3.1.3.4. | J |
| getcontentlength | Filstørrelse, vises i listen. | Se kap. 3.1.3.4. | J |
| iscollection | | Om ressursen er en mappe. For eksempel kan denne angi om et strukturert dokument skal vises som mappe eller enkeltdokument. Denne er omtalt som isfolder på MSDN. | N |
| isstructureddocument | | Strukturert dokument, i praksis en mappe som kan vises som et enkelt dokument. | N |
| defaultdocument | | Standard dokument for mappen. | N |
| displayname | Navn på filen | Se kap. 3.1.3.4. | J |
| isroot | | Indikerer om mappen er en rotmappe. | N |
| resourcetype | | Se kap. 3.1.3.4. | J |

**Tabell 2: Properties benyttet av WebFolder**

WebFolder har ingen funksjonalitet for å sette eller endre properties. Dette er en markant begrensning, for en viktig del av WebDAV-protokollen er anvendelse av metadata. Siden WebFolder ikke kan sette eller endre og kun viser en delmengde av definert metadata er den ubrukelig til dette formål. Ved bruk av WebFolder må altså eventuell metadatainformasjon integreres i filen som skal lastes opp (se kap. 4.2).

### 3.2.2 WebFolders avvik fra WebDAV

Microsoft WebFolders støtter WebDAV-protokollen, men med visse modifikasjoner. WebDAV er en utvidbar protokoll, og det er lagt opp til at klienter og tjenere skal kunne legge på ekstra informasjon og lignende som gjenkjennes i enkelte klienter men ignoreres i andre. Likeså med Microsoft, der WebFolder anvender tagger og properties som ikke er en del av WebDAV men som gjenkjennes av Microsofts egne WebDAV-kompatible tjenere (SharePoint og Exchange). Microsoft er allment kjent for sin "Embrace and Extend"-taktikk[5]. Og så lenge utvidelsene ikke bryter med protokollen er det problemfritt. Men WebFolder følger ikke protokollen på flere punkter. En liste over uregelmessigheter som er kjent ved ulike versjoner av WebFolder finnes i [31]. De viktigste av disse er nevnt under, og i tillegg noen momenter som ikke er i strid med protokollen men likevel verd og merke seg. Hvilke feil WebFolder inneholder varierer med versjonen, de nevnte er fra versjonen som følger med Windows 2000 (og alle Service Packs) og Office 2000. For full oversikt over versjoner refereres til [31].

- **Ukjente DAV:-properties.** Som omtalt i forrige kapittel ber WebFolders tjeneren om en rekke egenskaper i DAV:-navnerommet som ikke er definert i protokollen. Utvidelser på protokollen skal ikke benytte DAV:-navnerommet. Siden egenskapene (foreløpig) ikke kolliderer med egenskaper offentlig definert i DAV:-navnerommet vil WebDAV-tjenere kun returnere 404 (Ikke funnet) for alle disse, og det byr ikke på problemer. Sannsynligheten er vel til stede for at Microsoft også her får presset igjennom en standard ved å implementere den i sine produkter.

- **Feiltolkning av DAV:resourcetype-egenskapen.** DAV:resourcetype er definert som "typen til ressursen" og må finnes på alle ressurser, men kan gjerne være tom. Protokollen spesifiserer at (kun) dersom denne er satt til <DAV:collection/> er ressursen en mappe. WebFolder feiltolker dette og antar at alle ressurser som har en resourcetype-egenskap som ikke er tom er mapper. Dette gjør resourcetype-egenskapen unyttig for sitt formål, siden den degraderes til en bolsk er-ressursen-mappe-verdi. Dette er dermed en ganske alvorlig feil, siden mange WebDAV-tjenere vil ønske å benytte resourcetype slik protokollen angir at er mulig. En får håpe at Microsoft retter dette og at det ikke blir tjenerne som må akseptere denne feiltolkningen.

- **Åpne mapper med ikke-ASCII-navn.** Dersom man forsøker å åpne en mappe med et navn som innneholder ikke-ASCII-tegn (for eksempel æ,ø,å) gir feilmelding dersom tjeneren ikke kjører på port 80. Dog virker det som klienten ikke sender noen forespørsel til tjeneren. Dette er mest sannsynlig en bug i WebFolders som vil rettes i senere versjoner.

- **Filnavn på nye filer.** Når nye filer blir vellykket lastet opp vises de i listen med samme navn som den opplastede filen hadde. Det er helt naturlig for vanlige mapper, men siden WebDAV definerer en egenskap DAV:displayname som angir visningsnavn for ressursen er det dette som bør vises. Det er ikke gitt at visningsnavnet er det samme som navnet på den opplastede filen. Dog spesifiserer WebDAV-protokollen at opplastede filer skal få adressen som er oppgitt som måladresse, så denne vil ikke endres. Et forsøk på å hente filen vil

[5] "Embrace and Extend" refererer til strategien å ta i bruk en standard for så å utvide den utover hva standarden tilbyr, slik at egne produkter får fortrinn.

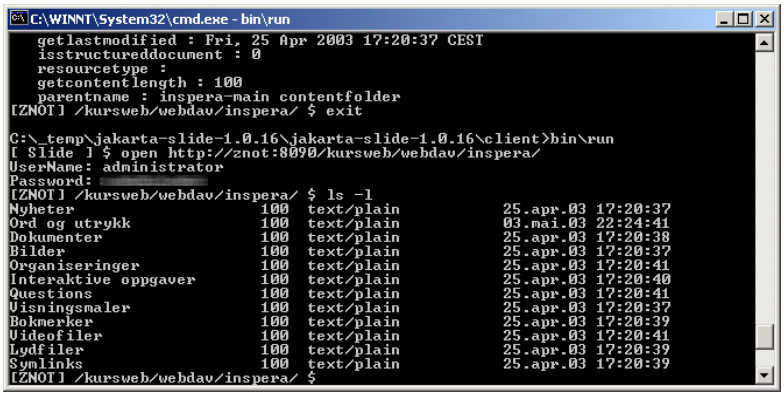
dermed fungere siden filen har riktig adresse, kun feil visningsnavn. Visningsnavnet vil rettes ved neste oppdatering av mappen.

## 3.3 Alternative WebDAV-klienter

Det finnes stadig flere klienter for WebDAV. Under gis en oversikt over noen av dem. Disse er klienter som kan kjøre på en Windows arbeidsstasjon, siden det er platformen som opereres med i denne rapporten, og de kan dermed sees på som alternativer til Microsoft WebFolder. Hovedbegrunnelse for at WebFolder er valgt til fordel for disse er WebFolders tette integrasjon med Explorer og applikasjoner som Word, aspektet som gir det den mest sømløse integrasjonen mulig. For en oversikt over flere WebDAV-klienter, også for andre platformer, henvises til [32]. Alle klienter nevnt under er opensource og gratis.

### 3.3.1 Jakarta Slide

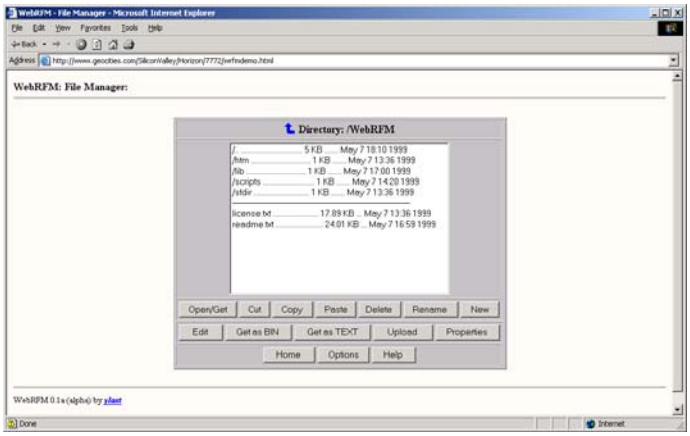
Jakarta Slide [33] er ikke kun en WebDAV-klient, men en javaimplementasjon for klient og tjener. Prosjektet inneholder en enkel, uavhengig kommandolinje-klient med full WebDAV-funksjonalitet, inkludert navigering, oppretting/sletting/kopiering, låsing og properties. Et bilde av grensesnittet til klienten er gitt i Figur 13. Siden klienten kun er kommandolinjebasert er den ikke et gangbart alternativ for ikke-avanserte brukere.




**Figur 13: Slide kommandolinjeklient-grensesnitt**

### 3.3.2 WebRFM

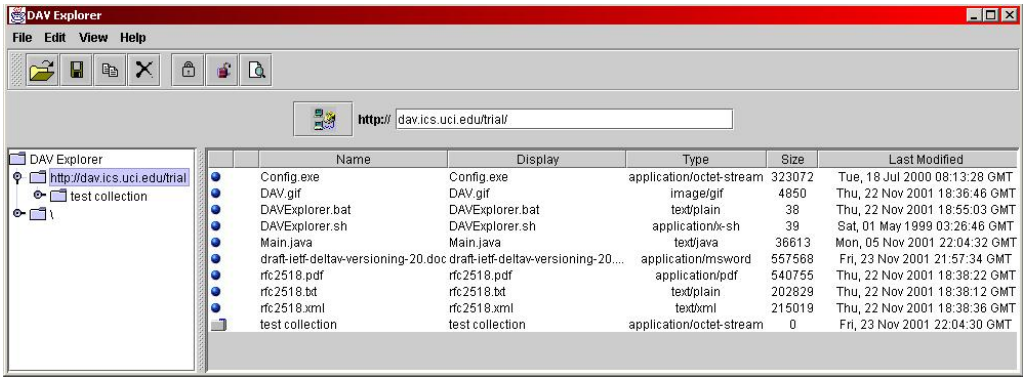
WebRFM (Web-based Remote File Manager, [34]) er en web-basert WebDAV-klient. Den har ikke støtte for låsing eller versjoner, men støtter den vanligste WebDAV-funksjonaliteten som oppretting og sletting av ressurser og mapper, navigering og properties. Figur 14 viser grensesnittet. Klienten er noe primitiv og virker ikke å ha et særlig brukervennlig grensesnitt.




**Figur 14: WebRFM-grensesnitt**

### 3.3.3 DAV Explorer

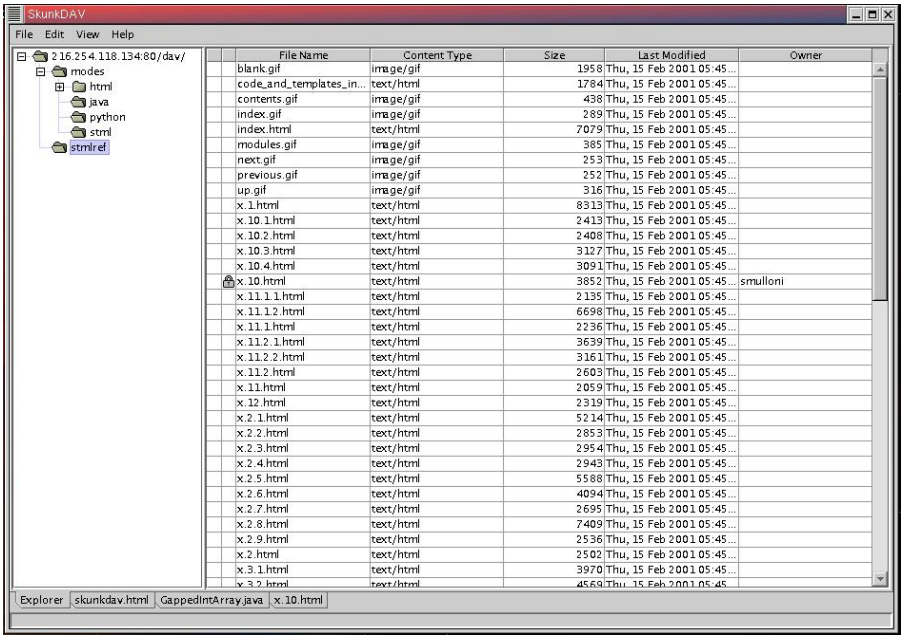
DAV Explorer [35] støtter full WebDAV-funksjonalitet, inkludert navigering, oppretting/sletting/kopiering, låsing og properties, gjennom et Java-basert grensesnitt. Et skjermbilde av grensesnittet er vist i Figur 15. Grensesnittet er konstruert ved hjelp av Java Swing, og forsøker å skape et Utforsker-aktig utseende. Dette gjør at klienten har en mye lavere brukerterskel enn de kommandolinjebaserte klientene. Klienten kan være et mulig alternativ til WebFolder siden den i motsetning til WebFolder har støtte for properties og låsing, og har et ikke avskrekkende grensesnitt. Siden klienten er skrevet i Java støttes den av de fleste operativsystemer med Java installert. I dokumentasjonen hevdes det at den er testet og fungerer på Windows 95-XP, Solaris, Linux og MacOS 9 og X.




**Figur 15: DAV Explorer-grensesnitt**

### 3.3.4 SkunkDAV

SkunkDAV [36] er nok en Java-basert WebDAV-klient og inneholder i tillegg til standard klient-funksjonalitet (navigering, opplasting, sletting, kopiering, låsing, properties og lignende) en innebygd teksteditor som kan benyttes til å redigere tekst og HTML-filer direkte på web-tjeneren. Grensesnittet er vist på Figur 16. Klienten ble skrevet for kommunikasjon mot Apaches mod_dav-modul, og interoperabilitet med andre WebDAV-tjenere er ikke dokumentert men burde i teorien fungere. Fra rapporter på klientens hjemmeside virker det som klienten stadig inneholder en del bugs men er fungerende til bruk.



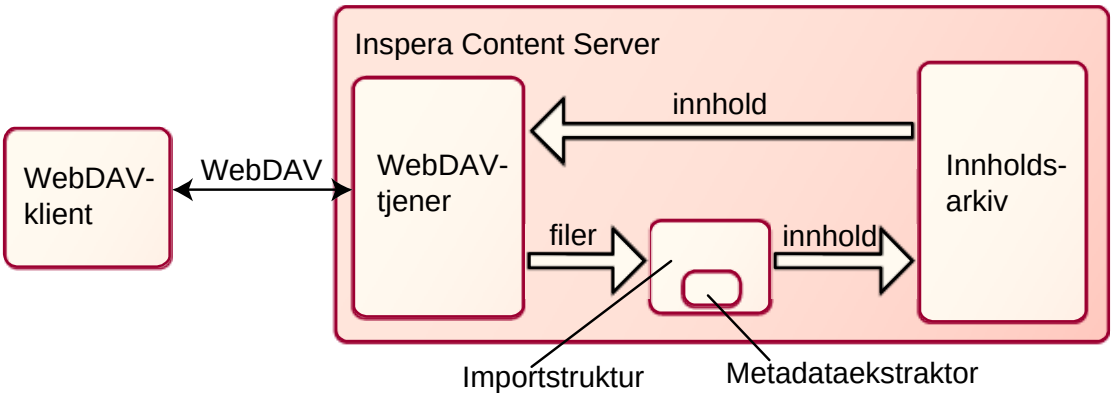

**Figur 16: SkunkDAV-grensesnitt**

# 4 Implementasjon

For å teste WebDAV og WebFolder som basis for sømløs integrasjon er det konstruert en prototypisk WebDAV-tjener ved å implementere WebDAV-støtte på Inspera Content Server. Implementasjonen består av tre deler:

- **WebDAV-støtte**. Den viktigste biten av implementasjon er å implementere støtte for selve WebDAV-protokollen, slik at tjeneren kan kommunisere med WebDAV-klienter.
- **Metadataekstraktor**. Det er implementert en ny modul som kan ekstrahere metadata fra innkomne filer, med foreløpig støtte for Microsoft Office og Adobe XMP. Mulighet for å metadatamerke allerede i produksjonsverktøyet bringer en enda nærmere sømløse integrasjon.
- **Importstruktur**. En generell, utvidbar importstruktur er konstruert. Denne konverterer dokumenter til Inspera Content Servers interne XML-struktur ved import. Dette gjør at selve innholdet i dokumentet blir tilgjengelig i CMS'et, i motsetning til kun en fil på proprietært format, og kan benyttes som innholdsdokument.

En overordnet figur over elementene implementasjonen består av er vist i figur Figur 17. På figuren vises at det er den nye WebDAV-delen av Inspera Content Server som håndterer kommunikasjon med WebDAV-klienter, og den benytter importstrukturen for å konvertere opplastede filer til innhold. Importstrukuren på sin side kan benytte seg av metadataekstraktoren for å hente ut metadata om innkomne filer, og lagre dette på det opprettede innholdselementet.




**Figur 17: Overordnet arkitekturskisse**

## 4.1 WebDAV-støtte på Inspera Content Server

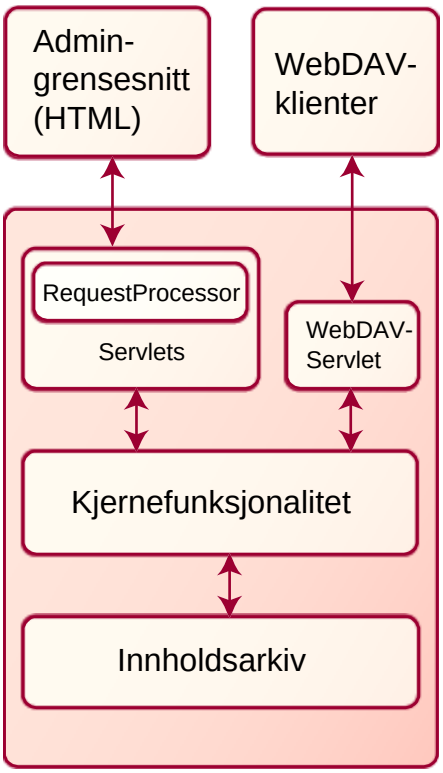
For at Inspera Content Server skal kunne kommunisere med WebDAV-klienter som Microsoft WebFolders må WebDAV-forespørsler håndteres og svar returneres i henhold til protokollen. Dette kapitlet beskriver design og implementasjon av denne nye funksjonaliteten i Inspera Content Server, som håndterer all kommunikasjon med WebDAV-klienter og mapper WebDAV-funksjonalitet over på kjernefunksjonalitet i CMS'et.

### 4.1.1 Design

Selv om protokollen spesifiserer tjenerens oppførsel og hvordan den skal svare på forespørsel sier den selvsagt ingenting om hvordan denne funksjonaliteten skal implementeres. Dette kapitlet beskriver en design for implementasjonen av WebDAV-funksjonalitet på Inspera Content Server, et proprietært CMS. Men siden det i designfasen er forsøkt å holde WebDAV-funksjonaliteten atskilt fra CMS'et bør dette designet kunne benyttes for andre systemer. Designet er basert på Jakarta Slide, en opensource WebDAV-tjener implementert i Java [33].

#### 4.1.1.1 Modell

WebDAV-funksjonaliteten vil ligge som et skall utenpå Inspera Content Servers kjernefunksjonalitet, og vil på samme måte som systemets integrerte administrasjonssider kalle definerte interne metoder i kjernen for å utføre operasjoner på innholdsarkivet. Denne modellen er vist i Figur 18.




**Figur 18: Overordnet funksjonsmodell**

For direkte funksjonalitet gjennom admingrensesnittet blir forespørslene rutet gjennom en RequestProcessor (superklasse til alle servlets) som sørger for autentisering og sesjonshåndtering, og står for generell forespørsel-parsing. Denne overlater så ansvaret til den adresserte servleten. Servleten kaller så metoder i kjernefunksjonaliteten med relevante parametrer. All logikk er inneholdt i kjernen, servletenes ansvar er kun håndtering og videresending av forespørsler, og generering av HTML til presentasjon[6].

[6] Gjøres egentlig i et eget presentasjonslag. Servletene genererer XML som sendes til presentasjonslaget som XSL-transformerer til HTML.

#### 4.1.1.2 Bruk av eksisterende funksjonalitet

WebDAV-delen av Inspera Content Server vil benytte seg av eksisterende metoder for å operere mot innholdsarkivet. Siden all funksjonalitet som WebDAV-protokollen tilbyr allerede er støttet i Inspera Content Server var det ikke nødvendig å implementere nye metoder i kjernefunksjonalitetslaget (Figur 18). Metodene som WebDAV benytter for å aksessere innholdsarkivet er inneholdt i tre klasser: ContentPresentationDataAccess, ContentAdminDataAccess og ContentFolderDataAccess. Disse tre klassene og metoder som WebDAV-klassene vil benytte er beskrevet under.

**ContentPresentationDataAccess:** Ansvarlig for dataaksess relatert til visning og uthenting av innhold. Viktigste metoder er

- getContentItem(contentItemId): Returnerer ContentItem med gitt id. Ved første uthenting hentes det fra databasen og caches.
- getLiveContentRevisionWithEffectiveLanguage(contentItemId, languageId): Returnerer aktiv revisjon for gitt ContentItem på gitt språk. Dersom ContentItem'et ikke har noen revisjoner sjekkes andre språk i prioritert rekkefølge.

**ContentAdminDataAccess:** Ansvarlig for dataaksess relatert til endringer i innholdsarkivet, innbefattet oppretting av nytt innhold, sletting og lignende. Viktigste metoder er

- newContentRevision(revision, languageId, marketplaceId): Metoder som lagrer ett nytt innholdselement med en revisjon. Objekttypen til den gitte revisjonen bestemer innholdstype. Gitt revisjon settes som aktiv.
- newContentRevisionOfExistingItem(revision, languageId, marketplaceId): Metode som lagrer en ny revisjon av et allerede eksisterende innholdselement. Gitt revisjon settes som aktiv.
- deleteContentItem(contentItemId, marketplaceId): Sletter et gitt ContentItem med alle tilhørende revisjoner.
- deleteContentRevision(contentRevisionId): Sletter en gitt revisjon.
- moveContentItem(contentItemId, targetFolderId, marketplaceId): Flytter et ContentItem til en annen mappe.
- copyContentItem(contentItemId, targetFolderId, marketplaceId): Kopierer et ContentItem til en annen mappe.

**ContentFolderDataAccess:** Ansvarlig for endringer i mappestrukturen. Mapper er ContentItems og har også revisjoner. Viktigste metoder er

- newContentFolder(contentFolder): Oppretter en ny mappe hvis denne ikke finnes, hvis den finnes opprettes kun en ny revisjon av denne mappen.
- moveContentFolder(contentFolderId, targetFolderId, marketplaceId): Flytter en mappe (med innhold) til en annen mappe.
- copyFolder(contentFolder, targetFolderId, marketplaceId): Kopierer en mappe til en annen mappe.

Tabell 3 viser hvordan disse metodene vil benyttes av de ulike WebDAV-metodene.

| WebDAV-metode | Beskrivelse | Metoder benyttet |
|---|---|---|
| COPY | Kopierer en ressurs eller mappe. | copyContentItem() eller copyFolder() avhengig om kilden er en ressurs eller en mappe. |
| MOVE | Flytter en ressurs eller en mappe. | moveContentItem() eller moveContentFolder(). |
| PUT | Lagrer en ressurs i en mappe. | newContentRevision() dersom det ikke finnes en ressurs med dette navnet i mappen. Da benyttes newContentRevisionOfExistingItem(). |
| GET | Henter ut en ressurs | getContentItem() og getLiveRevision(). |
| HEAD | Samme som GET, men ressursen returneres ikke. | getContentItem() og getLiveRevision(). |
| DELETE | Sletter en ressurs eller en mappe. | deleteContentItem(). Siden mapper også er ContentItem brukes denne metoden også for å slette mapper. |
| MKCOL | Oppretter en ny mappe. | newContentFolder(). |
| PROPFIND | Gir informasjon om en mappe og dens ressurser, eller en ressurs. | getContentItem() og getLiveRevision(). |

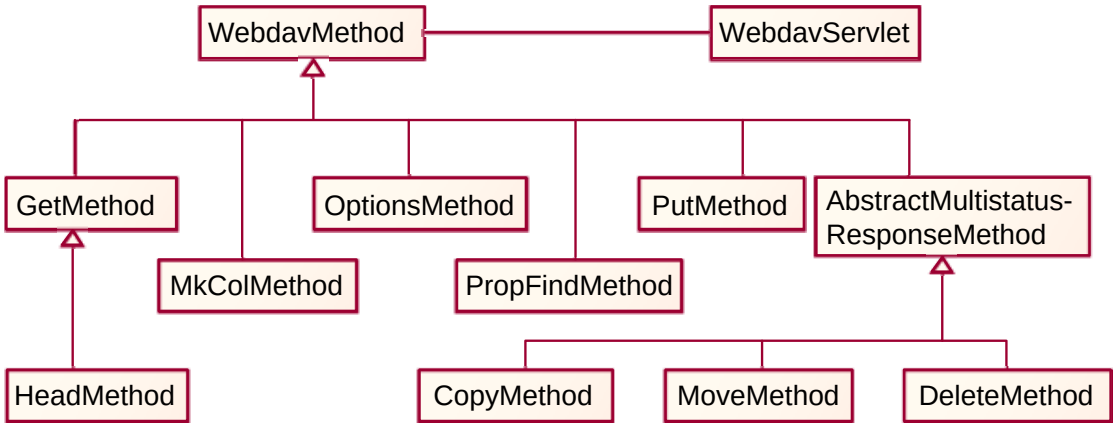
**Tabell 3: Relasjon mellom WebDAV-metoder og eksisterende funksjonalitet**

#### 4.1.1.3 Forespørselshåndtering

Inspera Content Server er bygd på en servlet-arkitektur, det vil si at ulike servlets håndterer ulik funksjonalitet. Hver servlet er mappet opp til en eller flere URI'er, slik at applikasjonsserveren ut fra URI'en vet hvilken servlet som skal håndtere denne forespørselen. Det var dermed naturlig å designe en egen servlet som skulle håndtere alle WebDAV-forespørsler. Denne ble mappet opp på /webdav, slik at en ressurs i innholdsarkivet vil få en URI på formen http://[webdav-tjener]/webdav/[ressurs-adresse].

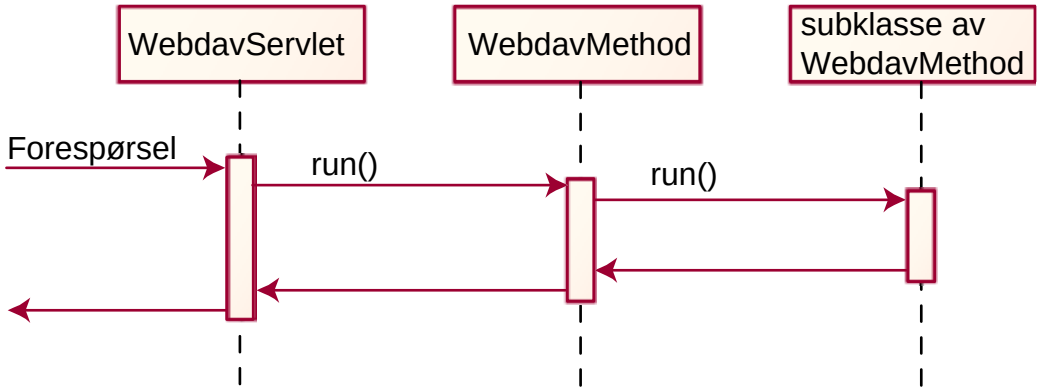
Siden Inspera Content Server støtter flere markedsplasser (portaler) på samme server, må det gjøres en distinksjon i URI'en på hvilken markedsplass som er ønsket. For standard forespørsler gjøres dette ved en request-parameter (eks: http://znot:8080/login?marketplaceId=7), men WebDAV benytter ikke slike parametre. Dermed kodes markedsplassnavnet inn i URI'en, og parses av WebDAV-servleten (for eksempel adresseres ressursen Documents/MyDocument på markedsplassen "inspera" ved http://znot:8080/webdav/inspera/Documents/MyDocument).

Siden implementasjon av WebDAV-støtte innebærer mye funksjonalitet ble det for oversiktlighets skyld bestemt å fordele den på forskjellige klasser. Basert på en struktur benyttet i Jakarta Slide deles det opp i ulike klasser for hver HTTP-metode. Siden en del overordnet forespørselshåndtering vil være felles for alle disse, er det konstruert en abstrakt superklasse som inneholder felles funksjonalitet, og som spesifiserer metodene som må implementeres av subklassene. Et klassediagram for dette er gitt i Figur 19.




**Figur 19: Klassediagram for WebDAV-klasser**

Ved en innkommende forespørsel oppretter WebdavServlet[7] en instans av en subklasse av WebdavMethod basert på HTTP-metoden som er benyttet. Deretter kalles metoden run() på objektet. Denne er implemert i superklassen WebdavMethod og gjør overordnet forespørselshåndtering før den sender kallet videre til metoden implementert i subklassen. En oversikt over sekvensen ved en forespørsel er vist i Figur 20.




**Figur 20: Sekvens ved forespørsel**

En ansvarsbeskrivelse for WebDAV-klassene er gitt under. Implementasjonen av disse klassene er beskrevet i kap. 4.1.2.

**WebdavServlet:** Tar imot forespørselen. Sørger for autentisering, sesjonshåndtering, parsing av adresse og lignende. Oppretter en instans av en subklasse av WebdavMethod basert på HTTP-metode, og kaller metoden run() i denne.

**WebdavMethod:** Abstrakt superklasse for alle metode-klasser. Inneholder generell funksjonalitet for å parse forespørselen, hente ut meldingshoder og annen overordnet forespørselshåndtering. Metoden run() utfører dette, og videresender til metoder deklarert i klassen og implementert i subklasser.

**GetMethod, HeadMethod, OptionsMethod, PostMethod, PropFindMethod, MkcolMethod, DeleteMethod, MoveMethod, CopyMethod, PutMethod:** Tilsvarer metoder definert i HTTP og WebDAV. Ansvar i henhold til nevnte protokoller.

[7] En kan korrekt påpeke at disse klassene i henhold til navnenotasjoner burde hett WebDAVServlet og lignende. Begrunnelsen for nåværende notasjon er helt enkelt at det er raskere å skrive.

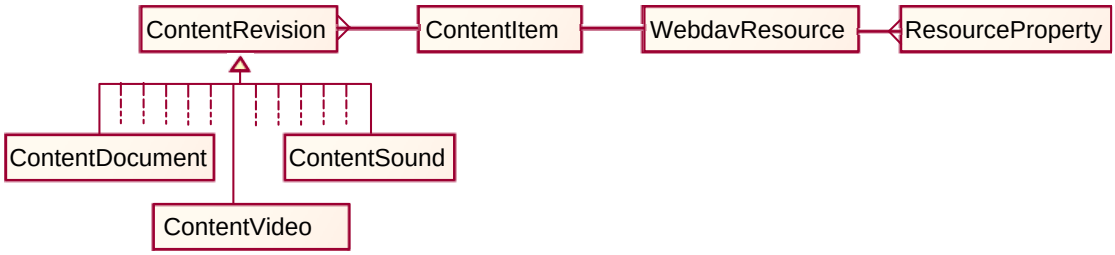
#### 4.1.1.4 Datamodell

Kjernen i WebDAV er ressursene. Alle operasjoner gjøres på en eller flere ressurser, som svarer til innholdselementer lagret i innholdsarkivet. Ressurser i innholdsarkivet representeres i Inspera Content Server ved klassen ContentItem, som har tilhørende revisjoner av klassen ContentRevision, hvorav en er markert som aktiv. ContentRevision er en superklasse for alle innholdstyper. Ulike innholdstyper har ulike subklasser. Eksempler på disse er ContentDocument, ContentSound, og ContentVideo. En kort oversikt over ContentItem og ContentRevision er gitt under.

**ContentItem:** Representerer en ressurs i innholdsarkivet. Inneholder logisk tittel og informasjon som ikke er språk/revisjonsavhengig og en lokal cache av aktive ContentRevision-objekter for ulike språk. Disse caches ved første forespørsel. Mapper er også ContentItems på samme måte som dokumenter og lydklipp, men med en liste av inneholdte ressurser i tillegg.

**ContentRevision:** Representerer en revisjon av et gitt ContentItem på et gitt språk. Inneholder navn, beskrivelse, innhold og lignende info som er generell på tvers av innholdstyper. De ulike innholdstypene har så egne subklasser med info som er spesifikk for denne innholdstypen.

I WebDAV-protokollen opereres det med en enkel datamodell bestående av *ressurser* strukturert i et hierarki, med tilhørende *egenskaper*. Ressursene tilsvarer et element i innholdsarkivet (inkludert mapper), og egenskapene er metadata om disse ressursene. Denne modellen er direkte overført til implementasjonen, der et objekt av klassen WebdavResource svarer til en ressurs, og klassen ResourceProperty tilsvarer egenskaper. Figur 21 viser relasjonen mellom WebDAV-datamodellen og Inspera Content Servers datamodell. En WebdavResoure har en en-til-en relasjon til ContentItemet det svarer til. Dermed vil en WebDAV-ressurs i praksis ha revisjoner siden ContentItem har det, men dette aspektet vil bli usynlig for WebDAV. Dersom Delta-V, WebDAVs versjonshåndteringsutvidelse, var tatt med ville ContentRevision-strukturen også vært synlig for WebDAV, men i foreliggende implementasjon er dette abstrahert bort, og en ressurs anses som et enhetlig element.




**Figur 21: Datamodell for WebDAV-ressurser**

Beskrivelse av de to nye WebDAV-dataklassene er gitt under.

**WebdavResource:** Representerer en WebDAV-ressurs, og omkapsler ett ContentItem. Inneholder metoder for å hente ut Property-informasjon om denne ressursen, gitt som ResourceProperty-objekter.

**ResourceProperty:** Representerer en Property ved en WebDAV-ressurs. Inneholder navn på Property'en inkludert navnerom, og tekstlig verdi.

Det er ikke designet noen cachestruktur for disse objektene. Det opprettes et nytt WebdavResource-objekt hver gang en forespørsel ber om en ressurs, og denne forsvinner etter at forespørselen er ferdig. Alternativt kunne en trestruktur av WebdavResource-objekter opprettes som var tilsvarende mappestrukturen i innholdsarkivet, og denne kunne brukes til å hente ønsket ressurs som i sin tur ga tilgang til innholdselementet. Det er flere grunner til at det ikke er valgt å lage en slik struktur. For det første er ikke effektivitet og skalerbarhet et mål i denne prototypeimplementasjonen, dersom implementasjonen skulle gjøres på produksjonsnivå ville dette aspektet testes og vurderes. ContentItem-objektene er cachet i en egen cache, dermed er ikke ytelsesvinningen så stor på en ekstra cache. For det andre vil en slik struktur være vanskelig å vedlikeholde. Siden WebDAV ikke er eneste innfallsvinkel til innholdsarkivet vil strukturen der endres av andre enn WebDAV-servleten. Det vil si at det måtte lages metoder for å holde cachestrukturen konsistent med strukturen i innholdsarkivet. Uten caching vet man at det WebdavResource-objektet som benyttes minimum var korrekt ved det tidspunktet forespørselen ble mottatt og ressursen hentet.

Heller ikke ResourceProperty-objektene vil caches. Ved uthenting av properties for en ressurs vil ResourceProperty-objekter opprettes for å representere disse egenskapene.

## 4.1.2 Implementasjonsbeskrivelse

Dette kapitlet vil beskrive implementasjonen av WebDAV-støtte på Inspera Content Server. Implementasjonen er så langt som mulig gjort uavhengig av den interne strukturen i kjernefunksjonaliteten på CMS'et. Det vil si at det er forsøkt å holde et klart skille mellom WebDAV-implementasjonen og den eksisterende funksjonalitet. Dette er mulig siden WebDAV-implementasjonen ikke innebærer nyutvikling av kjernefunksjonalitet, kun en ny innfallsvinkel til det eksisterende. De nye klassene som er implementert er lagt i ulike pakker avhengig av funksjonalitet, og vil beskrives pakkevis i de følgende underkapitler, etterfulgt av en oversikt over hvilke forenklinger som er gjort under implementasjonen.

### 4.1.2.1 Uthenting av ressurser

Uthenting av ressurser gjøres av en egen klasse, WebdavDataAccess, som er lagt i pakken for alle klasser i dataaksesslaget, no.inspera.db. Den eneste aksess klassen har mot databasen er for å hente ut rotmapper for alle markedsplasser. Likvevel rettferdiggjøres pakkeplasseringen ytterligere siden uthenting av ressurser gjøres gjennom andre dataaksessklasser i samme pakke. Det er også mulig at utvidelser av denne klassen naturlig vil inneholde databasekall, for eksempel dersom metodikken for uthenting av ressurser endres til å gå direkte mot db. Kildekode for WebdavDataAccess finnes i appendiks A-1.

**WebdavDataAccess**
WebdavDataAccess er ansvarlig for uthenting av ressurser. Alle mapper fra innholdsarkivet har referanser til sine undermapper, så uthentingen vil gå rekursivt gjennom hele adressen. Uthentingen av en ressurs gjøres som følger:

1. Adressen til ressursen deles opp ved ”/”. Hver del tilsvarer en mappe i innholdsarkivet.
2. Den første mappen vil være en undermappe av markedspassens rotmappe. Den hentes ut fra innholdsarkivet.
3. Alle følgende mapper (deler av den originale adressen) vil være undermappe av den foregående, og så hentes ut fra innholdsarkivet
4. Den siste delen vil ikke være en mappe, så siden enden av adressen er nådd returneres denne ressursen.

I tillegg til metoder for uthenting av en ressurs fra en URI inneholder WebdavDataAccess metoder for å hente ut alle ”barn” av en mappe, og for å hente ut et barn med et gitt navn. Det er sistnevnte metode som benyttes av metoden for uthenting av en ressurs fra en adresse.

#### 4.1.2.2 Generell WebDAV-funksjonalitet

Det er laget en ny javapakke under applications-strukturen i Insperas system, *no.inspera.applications.webdav*. Her ligger alt som har med WebDAV-funksjonalitet å gjøre (bortsett fra WebdavDataAccess). Pakken inneholder WebdavServlet, som svarer på WebDAV-forespørsler; WebdavStatus, en hjelperklasse for HTTP- og WebDAV-statuskoder; WebdavException, feilmeldingene som WebDAV-klassene kaster; MultiStatusException, som samler flere (hierarkiske) feilmeldinger for metoder som kaster slike (copy, move og delete). Kildekode for alle klassene i pakken finnes i appendiks A-2.

**WebdavServlet**
WebdavServlet har ansvaret for å håndtere innkommende WebDAV-forespørsler. Når en forespørsel ankommer parser WebdavServlet URI’en som er forespurt, og henter navnet på markedsplassen. Hvis URI’en er korrekt (på formen http://[tjener]/webdav/[markedsplassnavn]/[ressurs-adresse]) forsøker servleten å autentisere brukeren på denne markedsplassen. Dette gjøres ved HTTP Basic-autentisering (beskrevet i [27]). Hvis brukeren ikke er autentisert får hun opp en dialogboks for brukernavn og passord. Når brukeren er korrekt autentisert sjekker servleten hvilken HTTP-metode forespørselen benytter, og oppretter tilsvarende subklasse av WebdavMethod, med forespørsel- og svarobjektene som parametrer. Servleten kaller så metoden run() på dette objektet.

**WebdavStatus**
WebdavStatus inneholder konstanter for alle statuskoder som benyttes i WebDAV-protokollen, og en metode for å hente statustekst for disse. Klassen benyttes til å returnere riktig statuskode til klienten, med statustekst. Hvert svar inneholder en statuskode, som indikerer hva resultatet på forespørselen ble, slik at klienten kan gi en beskrivende feilmelding til brukeren, eventuelt sende en ny, endret forespørsel.

**WebdavException**
Inspera Content Server benytter egne Exception-subklasser for å kunne gi informative feilmeldinger og bedre kunne håndtere feil. WebdavException representerer en feil i WebDAV-prosesseringen.

**MultiStatusException**
WebDAV inneholder metoder som kan utføres på en samling ressurser, ikke bare én. Eksempel er ved sletting av en mappe. Her kan det forekomme feil på alle ressurser i mappen og eventuelle undermapper. Slike feil kan ikke representeres med en enkel statuskode. Dermed er multistatuskoden innført, som gir en samling feilmeldinger i meldingskroppen, sammen med adressen til ressursene feilene oppsto på. MultiStatusException representerer en slik multistatusfeil. Den inneholder metoder for å legge til en statuskode-adresse-tuppel, og returnere en iterator over alle disse.

#### 4.1.2.3 Datamodell

Klassene som representerer datamodellen er skilt ut i en egen pakke, *no.inspera.applications.webdav.datamodel*. Datamodellen beskrevet i kap. 4.1.1.4 utgjør to klasser, WebdavResource og ResourceProperty, som tilsvarer henholdsvins en ressurs og en egenskap. Kildekode til disse klassene finnes i appendiks A-3.

**WebdavResource**
En adresserbar ressurs i WebDAV tilsvarer et ContentItem i innholdsarkivet. WebdavResource inneholder en referanse til ContentItemet denne ressursen representerer og adressen til ressursen. I tillegg til å gi tilgang til selve ressursen har klassen ansvar for å levere ut properties definert på ressursen, ved hjelp av metoden getProperties() som tar inn en liste med navn på properties som ønskes hentet, eller en tom liste for alle properties. Properties gjenkjent av serveren (i praksis alle DAV:-properties) hentes ut dynamisk gjennom mappinger til egenskaper ved ressursen (se mappingstabell i kap. 4.2.2). Det er lagt opp til at properties definert i et Inspera-navnerom skal kunne svare til Insperas egne metadatastruktur (ItemProperties), men dette er ikke implementert.

**ResourceProperty**
ResourceProperty representerer en egenskap (property) ved en ressurs. Den inneholder navn, navnerom og verdi, alle som strenger. I tillegg inneholder klassen en samling ferdige ResoureProperty-objekter som konstanter. Dette er alle egenskapene i DAV:-navnerommet, uten verdi, til bruk ved sammenligning.

#### 4.1.2.4 WebDAV-metoder

I likhet med datamodell-klassene, er også klassene som representerer WebDAV-metoder skilt ut i sin egen pakke, *no.inspera.applications.webdav.method*. Pakken inneholder ulike klasser for alle forespørselsmetoder som forekommer i WebDAV. Hver klasse har ansvar for å håndtere en metode. Alle klassene er subklasser av WebdavMethod, som inneholder noe felles funksjonalitet. Kildekode for alle klassene er i appendiks A-4.

**WebdavMethod**
WebdavMethod er en abstrakt superklasse for klasser som implementerer WebDAV-HTTP-metoder. Klassen inneholder funksjonalitet som er generell for alle metode-klassene, blant annet for å lese innholdet av forespørselen og bygge XML-tre, og lese ut adressen til forespurt ressurs fra URL'en. Den definerer også de to abstrakte metodene som alle subklassene må implementere; parseRequest() og executeRequest(). Disse kalles fra metoden run() som er definert i denne klassen og arves til alle subklasser. Det er denne metoden som kalles fra WebdavServlet.

**AbstractMultistatusResponseMethod**
AbstractMultistatusResponseMethod er en som navnet antyder en abstrakt superklasse for metodeklasser som kan utføre operasjoner på en samling ressurser og dermed må returnere en multistatus-struktur der hver ressurs har sin egen statuskode. Klassen inneholder en metode for å generere en slik multistatus-melding. Subklasser av denne klassen er CopyMethod, DeleteMethod og MoveMethod.

**CopyMethod**
Ved kopiering sjekkes først om mappen der kopien skal legges eksisterer. Så sjekkes det hvorvidt det eksisterer en ressurs i denne mappen med samme navn som kopien. Hvis Overwrite-headeren er satt til true slettes så denne ressursen. Deretter hentes kilderessursen, og copyContentItem() i ContentAdminDataAccess kalles med ressursen og målmappen som parametere. Dersom navnet på kilderessursen skiller seg fra målressursen opprettes det deretter en ny revisjon av den nye kopien med det nye navnet. Protokollen spesifiseres alternativer for hvordan properties skal kopieres, men siden prototypeimplementasjonen kun opererer med dynamiske properties er dette ikke aktuelt. I tillegg spesifiserer protokollen muligheten for rekursiv kopiering, det vil si kopiering med undermapper. Det er ikke støttet i implementasjonen.

**DeleteMethod**
Sletting gjøres i hovedsak av kjernefunksjonalitet. DeleteMethod henter den adresserte ressursen for å finne id til ContentItem, og kaller deleteContentItem i ContentAdminDataAccess for å slette ressursen. Siden selve slettingen gjøres i databasen har ikke Java-siden noen oversikt over feil som måtte dukke opp under sletting av subtrer, slik WebDAV-protokollen spesifiseres at skal returneres til klient. Alternativt måtte slettingen foregå rekursivt gjennom treet med individuelle databasekall for hver ressurs.

**GetMethod**
GetMethod forsøker å hente ressursen forespurt. Dette gjøres ved å bruke WebdavDataAccess til å hente finne den adresserte WebdavResource hvis den eksisterer, og så hente ContentItem ut fra denne. Forespørselen videresendes så til visningsservleten med id'en til dette ContentItem.

**HeadMethod**
HEAD er definert som en GET uten retur av ressursen. Derfor er HeadMethod kun en subklasse av GetMethod, der en boolean settes for å indikere at innholdet av ressursen ikke skal sendes til klienten.

**MkColMethod**
MkColMethod sjekker først at mappen som den nye mappen skal opprettes i eksisterer. Deretter opprettes en ny ContentFolder med den mappen som foreldremappe og navn som spesifisert i adressen, og metoden newContentFolder() i ContentFolderDataAccess kalles for å lagre den nye mappen. På Inspera Content Server har mapper tilordnet ett sett med innholdstyper som kan legges i denne mappen. Siden dette ikke kan spesifiseres over WebDAV arver nye mapper foreldremappens registrerte innholdstyper.

**MoveMethod**
MOVE gjøres tilsvarende som COPY, med unntak av kjernemetoden som kalles (moveContentItem() i stedet for copyContentItem()). Protokollen spesifiserer at et flytt

av et subtre skal gjøres etter beste evne selv om et undertre har feilet, men det er ikke aktuelt her siden en flytting kun innebærer endring av foreldremappe for ressursen og ikke involverer subtrær.

**OptionsMethod**
OPTIONS skal returnere info om tjeneren. OptionsMethod returnerer en liste med støttede metoder (de samme som nevnt i dette kapittel), og info om at denne tjeneren støtter DAV versjon 1. DAV versjon 2 innebærer låsing av ressurser, som ikke er en del av denne prototypen (se kap 4.1.2.5)

**PostMethod**
POST fungerer som en PUT for andre ressurser enn samlinger, der POST ikke er definert. PostMethod er dermed en subklasse av PutMethod, og kaller executeRequest() i denne dersom adressen ikke er en samling. Hvis den er det returneres statuskoden for konflikt (409).

**PropFindMethod**
Denne klassen håndterer PROPFIND, altså uthenting av egenskaper for en gitt ressurs. Metoden håndterer de tre variantene av PROPFIND (alle egenskaper, bestemte egenskaper eller navn på tilgjengelige egenskaper) for seg, men med lignende metodikk. Først hentes egenskapene til den adresserte ressursen, og dersom forespørselen ber om dybde større enn null legges alle barn av ressursen på stakken, og prosessen gjentas. For å hente egenskapene til en ressurs kalles metoden getProperties på WebdavResource-objektet (se 4.1.1.4).

**PutMethod**
PutMethod benytter seg av den nyutviklede importstrukturen beskrevet i kap. 4.3. Denne forutsetter at MIME-type for filen er kjent, så PutMethod forsøker å dedusere dette utfra filetternavn, etter en definert mapping. WebDAV spesifiseres at dersom det allerede ligger en ressurs på måladressen skal denne overskrives, untatt hvis det er en mappe da konflikt oppstår. Dersom den nye og gamle ressursen er av samme innholdstype (for eksempel ContentDocument eller ContentSound) gjøres dog ikke dette, men det opprettes en ny revisjon av ressursen. Hvis de er av ulike innholdstyper slettes den gamle og et nytt innholdselement opprettes.

#### 4.1.2.5 Forenklinger gjort under implementasjonen

Siden implementasjonen av WebDAV på Inspera Content Server er ment som en prototyp for å vurdere gagnetheten av WebDAV i en praktisk situasjon, er det gjort en del forenklinger i forhold til en fullstendig implementasjon av protokollen. Disse forenklingene er beskrevet og begrunnet under, inkludert hvilke implikasjoner de vil ha for bruk av WebDAV.

- **Opplastede filer kan få annen adresse enn måladressen.**
  *Beskrivelse:* I strid med protokollen kan opplastede filer få en annen adresse enn måladressen. Kun siste del (navnet på filen) vil kunne være annerledes.
  *Begrunnelse:* Som URI-del benyttes navn på ContentItem. For dokumenter som inneholder tittel i metadata vil denne bli brukt som navn i stedet for filnavnet, og adressen til den nye ressursen vil avlutte med dette navnet i stedet for filnavnet.
  *Implikasjoner:* Siden WebFolder (korrekt) antar at den opplastede filen vil få samme adresse som måladressen sjekkes ikke dette, og filen antas å ha denne

adressen helt til neste oppdatering av mappen. Dermed vil ikke uthenting av filen fungere før mappen er oppdatert.

- **PROPPATCH-metode ikke støttet.**
  *Beskrivelse:* Prototype-implementasjonen støtter ikke endring og setting av properties ved hjelp av PROPPATCH-metoden.
  *Begrunnelse:* Det ble ikke ansett som nødvnedig å implementere denne metoden siden WebDAV-klienten som brukes ikke støtter den.
  *Implikasjoner:* Utgjør ingen forskjell for WebFolder, men vil gjøre at andre klienter som har mulighet for å endre properties ikke får det til.
- **Begrenset tilgangskontroll.**
  *Beskrivelse:* Prototypen har svært begrenset kontroll av rettigheter. Den eneste rettighetssjekk som gjøres er ved første åpning av en WebFolder. Dersom bruker har tilgang lagres det på sesjonen og ingen videre rettighetssjekk gjøres.
  *Begrunnelse:* Inspera Content Server åpner for at brukere kan tilordnes ulike privilegier på objektnivå, det vil si at det for én eller en gruppe brukere kan gis rettigheter som endring, sletting og lignende for ulike mapper eller ressurser. Begrunnelsen for at dette ikke benyttes er hovedsakelig av tidshensyn, det er ikke snakk om designendringer for å implementere dette, kun en del å kode.
  *Implikasjoner:* For brukertesten og evalueringen utgjør dette ingen forskjell, da sikkerhet ikke er et aspekt som vil vurderes. Men dersom videre bruk og arbeid på systemet påbegynnes bør sikkerhet være blant de første prioritetene, da dette både hindrer uønsket inntrenging og brukerfeil, ved at brukere ikke har mulighet til å gjøre noe de ikke burde.
- **Låsing.**
  *Beskrivelse:* Det er ikke implementert noen funksjonalitet for låsing i prototypen. Dette er spesifisert i protokollen.
  *Begrunnelse:* Låsing er viktig dersom flere brukere vil anvende samme dokumentmengde samtidig. Siden det ikke er tilfelle i brukertesten er ikke låsing implementert. I tillegg støtter ikke WebFolders i Explorer låsing, det gjør derimot Word. Testing har vist at dersom Word oppdager at låsing ikke er implementert ignorerer den låsestrategien og åpner dokumentet. En annen viktig grunn til at låsing ikke er implementert er at det ikke hittil er støttet i Inspera Content Server, og dermed vil en låseimplemetasjon enten bli ad-hoc og prototypisk eller innebære mye design og koding i kjernen.
  *Implikasjoner:* Ingen problemer for testing, da flerbruk ikke er et aspekt som testes. WebDAVs låsestrategi evalueres på papiret.
- **Feilmeldinger.**
  *Beskrivelse:* WebDAV-protokollen spesifiserer hvilke statuskoder som skal returneres til klienten ved ulike feilsituasjoner hos tjeneren. Ikke alle disse er fulgt.
  *Begrunnelse:* Dette er mer en følge av at utføringen av enkelte av operasjonene ikke utføres som protokollen antar. Eksempelvis sletting av mappe, som gjøres på databasesiden i stedet for er rekursivt kall gjennom subtreet. Dermed er det ikke mulig å gi riktig multistatus dersom en feil forekommer dypt i treet. I andre tilfeller er det usikkert hvilken statuskode som best beskriver feilen og som burde returneres. Ikke alle feilsituasjoner som omtales i protokollen er parallelle til de som kan forekomme på tjeneren.
  *Implikasjoner:* Ingen alvorlige implikasjoner, begrenser kun hvorvidt klienten kan gi nyttige tilbakemeldinger til brukeren. Det virker dessuten som WebFolder ikke er så nøye hvilke statuskoder som returneres og returnerer samme feilmeldinger uansett.
- **Avansert mappefunksjonalitet.**

*Beskrivelse:* Under omtale av MKCOL-metoden nevner WebDAV-protokollen at det er mulig å gi avansert funksjonalitet med metoden slik at den oppretter filer i den nye mappen og lignende, men at dette skal spesifiseres i senere dokumenter. Denne funksjonaliteten er ikke implementert.
*Begrunnelse:* Grunnen til at denne funksjonaliteten ikke er implementert er i hovedsak at den ikke er beskrevet.
*Implikasjoner:* Ingen kjente klienter (i alle fall ikke MS WebFolders) benytter denne funksjonaliteten.

- **Rekursiv COPY.**
  *Beskrivelse:* COPY kan utføres med en dybde større enn 1, som angir at mappen og eventuelle inneholdte dokumenter og undermapper skal kopieres. Dette støttes ikke i implementasjonen.
  *Begrunnelse:* Rekursiv kopiering støttes ikke av kjernefunksjonaliteten, og ville derfor utgjort en utvidning av kjernen. Dette er ikke ansett som nødvendig og er satt som en avgrensning av prototypen.
  *Implikasjoner:* Kopiering av mapper i WebFolder fungerer ikke som i Utforsker, der en kopiering av en mappe innebærer en kopiering av alt innhold i mappen.
- **Full properties-støtte.**
  *Beskrivelse:* Implementasjonen støtter kun de properties som er definert i DAV:-navnerommet og som er definert i protokollen, og har ingen støtte for lagring eller henting av andre properties. Heller ikke for å liste ut hvilke properties som er definert på ressursen.
  *Begrunnelse:* Siden properties er såpass begrenset støttet i WebFolders ble property-støtten i prototypen avgrenset. Generelt burde properties lagres som ItemProperties i den interne metadatastrukturen.
  *Implikasjoner:* Siden WebFolder ikke støtter lagring av properties og ikke henter andre enn de i DAV:-navnerommet merkes ikke denne avgrensningen her. Andre klienter som støtter dette vil ikke fungere som ønsket.

### 4.1.3 Diskusjon

Prototypeimplementasjonen av WebDAV-støtte på Inspera Content Server bød ikke på store problemer. Designet kunne knyttes opp til eksisterende funksjonalitet og datamodell og benytte seg av disse, slik at det eneste som måtte designes var datamodellen som skulle benyttes for å representere WebDAV-ressurser og modulen som skulle stå for WebDAV-kommunikasjon. Grunnen til at det ble opprettet en egen datamodell for WebDAV-ressurser i stedet for å benytte den interne modellen var at det forenklet implementasjonen, siden de andre WebDAV-klassene fikk et grensesnitt til ressursene som var spesialisert til WebDAV-funksjonaliteten. Dog var datamodellen for WebDAV-ressurser lite annet enn en innkapsling av innholdsarkivets datamodell. Den eneste ekstra funksjonaliteten ressursobjektene hadde var knytningen mot properties og mulighet til å hente ut disse. En alternativ implementasjon kunne lagt denne funksjonaliteten i en hjelpe-klasse, og benyttet innholdsarkivets datamodell direkte. Det vil nok være enda mer aktuelt dersom en implementasjon av Delta-V-versjonshåndtering skal gjøres, siden det da vil være nødvendig å aksessere revisjonene og ikke bare selve innholdselementet. En egen WebDAV-datamodell vil da redundant duplisere den interne strukturen.

At implementasjonen kunne bygges utenpå den eksisterende funksjonaliteten er nok et tegn på WebDAV-protokollens allmennhet. De metoder og muligheter protokollen tilbyr tilbys i de fleste CMS. Grunnen til dette er nok at denne samlingen metoder er

nødvendige for å bedrive innholdsproduksjon og organisering. Lagring og uthenting er åpenbart nødvendige. Organisering av innhold er vanlig i de fleste systemer, og da trengs metoder for oppretting av mapper, sletting, kopiering og flytting. Metadata er også svært vanlig. Selv om representasjonen av metadata kan variere veldig fra system til system vil et minstekrav være uthenting og setting av atomiske verdier. De nevnte metodene er grunnlaget i hva som tilbys av WebDAV. Siden alle disse allerede var implementert og fungerende i Inspera Content Server kunne WebDAV-protokollen implementeres som en adskilt modul som kalte opp interne metoder for å utføre operasjoner i innhodsarkivet. Denne todelingen gjør også implementasjonen mer robust, siden selv om Inspera Content Server er i stadig utvikling og kjernen nok vil endre seg er det mindre sannsynlig at grensesnittet for de metodene som WebDAV-klassene anvender vil endre seg. Dermed vil endringer i kjernen ikke medføre endringer i WebDAV-klassene. En ideell todeling har ikke vært mulig å gjennomføre, men så langt som mulig har WebDAV-klassene kun anvendt bestemte metoder i kjernefunksjonaliteten. Det kan også være på sin plass å vurdere hvorvidt denne todelingen er det mest gunstige. Som nevnt i forrige avsnitt kan det argumenteres for at WebDAV-klassene heller burde benytte intern datamodell direkte, og dette vil innebære at todelingen faller. Økt effektivitet vil gå på bekostning av vedlikeholdsevne.

En risiko ved implementasjonen er at protokollen kan endre seg. WebDAV er foreløpig på ”Request for Comments”-stadiet på vei mot å bli en Internet Standard, så det kan fortsatt bli gjort endringer i protokollen etter hvert som ytterligere implementasjoner blir gjort og kommentert. Dog har protokollen vært uendret i nesten 5 år, så store endringer i funksjonalitet eller struktur er ikke sannsynlig. Små korreksjoner eller utvidelser er mulig, men det er overhengende sannsynlig at disse vil være bakoverkompatible. Dermed vil de innebære liten eller ingen nødvendighet for endringer i Inspera Content Servers implementasjon, kun dersom den nye funksjonaliteten skulle integreres.

Implementasjonen som er gjort er en prototypeimplementasjon. Full WebDAV-støtte er ikke implementert. Kapittel 4.1.2.5 har beskrevet de forenklinger som er blitt gjort for å avgrense implementasjonen til det som var nødvendig for å kunne utføre evalueringen og brukertesten. Dersom WebDAV skal tas i bruk på Inspera Content Server må implementasjonen fullføres. Annonsert WebDAV-støtte betyr at protokollen er støttet fullt ut, ikke kun til den grad den er nødvendig for bruk av WebFolder, som er tilfelle i skrivende stund. Viktigst av det resterende arbeidet er nok full property-støtte. Prototypeimplementasjonen inneholder kun de live properties som er spesifisert i protokollen, men dette må utvides til å kunne støtte uthenting og setting av vilkårlige, egendefinerte properties. Dette arbeidet er påbegynt og er en del av designet, men implementasjonen må fullføres.

## 4.2 Ekstrahering av metadata

WebDAV tilbyr lagring og henting av metadata i WebDAV-klienter. Men ved å ha mulighet til å hente metadata direkte fra importerte filer kan disse filenes interne metadatastrukturer benyttes. Ikke alle filer kan lagre metadata, men for de som kan (for eksempel Word og Adobe-filer) kan merking av metadata kan gjøres allerede i produksjonsverktøyet, og disse metadata kan lagres i innholdsarkivet. Spesielt ved bruk av WebFolder vil en slik funksjonalitet være essensiell siden det har vist seg at WebFolders ikke støtter merking av metadata.

På grunn av ulike metadatastrukturer i ulike systemer er det ingen selvfølge at metadata kan medfølge et dokuments konverteringer. En gjennomgang av metadataformatene benyttet i denne sammenhengeng er gitt i kapittel 4.2.1. En oversikt over hvordan benyttet metadata mappes mellom disse formatene er gitt i kapittel 4.2.2, og implementasjonen av to metadataekstraktore er vist i kap. 4.2.3.

## 4.2.1 Metadataformater

Ulike systemer og programmer har egne metadatastrukturer, både med hensyn på hvilke metadata som er tatt med og hvordan de representeres. På grunn av dette er det ikke gitt at all metadata kan konverteres fra ett format til et annet. I denne rapportens kontekst er det fire systemer som involveres; Inspera Content Server som innholdsarkiv, WebDAV som protokoll, og Microsoft Word og Adobe Framemaker som sluttbrukerapplikasjoner. Alle disse har ulike metadataformater, og siden det er nødvendig å ta med metadata fra det ene formatet til det neste må metadataformatene kunne tolkes og konverteres. Dette kapitlet beskriver metadataformatene i disse fire systemene.

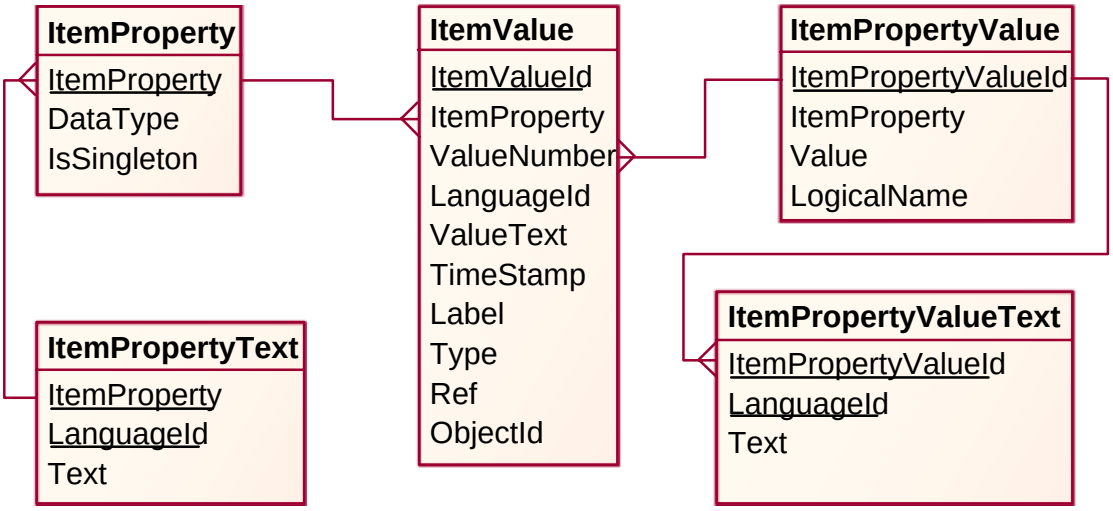
### 4.2.1.1 WebDAV

Metadatamodellen benyttet i WebDAV er utførlig beskrevet i kap. 3.1.3.4, det vil gis et sammendrag her for å forenkle sammenligning med de andre modellene.

WebDAV benytter navn-verdi-tupler for lagring av metadata, kalt *properties*. Det finnes to typer properties, *live* og *dead*. *Live* properties har en semantikk og syntaks som kontrolleres av tjeneren, mens *dead* properties ikke har noen egen semantikk. Dead properties lagres og hentes direkte av tjeneren uten noen tolkning, og har dermed ingen egen struktur. Hvorvidt klienten tillegger property'en en semantikk er usagt, men det kan ikke forventes at alle klienter skal tillegge en dead property samme (eller noen) semantikk. 11 properties er definert i WebDAV-protokollen med tilhørende semantikk og syntaks, og brukere kan selv definere andre properties.

### 4.2.1.2 Inspera Content Server

Inspera Content Server opererer med ItemProperties, som er sitt proprietære metadataformat. Et ER-diagram over ItemProperties-modellen er gitt i Figur 22.




**Figur 22: Insperas ItemProperty-modell**

ItemProperty svarer til ett metadatafelt (for eksempel forfatter). En ItemProperty kan være av typene *boolean* (binær verdi), *number* (tallverdi), *string* (tekstlig verdi), *option value* (ett av alternativer) eller *multiple value* (ett eller flere alternativer). I tillegg har ItemProperty tilordnet en tekstlig beskrivelse på ett eller flere språk, gitt i ItemPropertyText.

En metadatatuppel er representert som en ItemValue. En ItemValue er knyttet til en ItemProperty, som er "navnet", og ItemValue er "verdien". Typen på verdien er gitt av DataType i ItemProperty. Alle typer lagrer verdien tekstlig i ValueText, bortsett fra "option value" og "multiple value"-typene, der det lages en fremmednøkkel i ValueNumber, som viser til en definert ItemPropertyValue (som representerer valgmulighetene for denne ItemProperty). ItemValues knyttes til objekter med ObjectId-fremmednøkkelen.

#### 4.2.1.3 Microsoft Word

Microsoft Word opererer med navn-verdi-tupler. Verdiene kan være av typene tekst, dato, tall eller ja-eller-nei (binær). En samling metadata er automatisk generert av Word, for eksempel Navn, Type, Sist Endret, Forfatter (basert på innstillinger i Word) og lignende. I tillegg er en del metadata definerte men ikke gitt verdi, for eksempel Tittel, Emne, Firma, Nøkkelord og lignende. Disse kan settes av brukeren. Til sist kan brukeren også definere egne metadata-tupler. Her kan brukeren gi egne navn (eller velge fra en liste av 27 alternativer) og sette verdi av valgt type. Disse lagres i dokumentet men tolkes ikke semantisk av Word.

Alle Microsoft Office-filer lagrer metadata på samme måte. Filformatet er basert på Microsofts "OLE2 Compound Document"-format. Her opereres det med et eget filsystem internt i filen, slik at det deles opp i mapper og underfiler. For eksempel inneholder en vanlig Word-fil tre interne filer[8]: "WordDocument", "SummaryInformation" og "DocumentSummaryInformation". Den første inneholder selve dokumentet, mens de to neste inneholder metadata. Mer om dette i kapittel 4.2.3.1.

[8] Vanligvis også flere filer, men disse er de relevante i denne sammenheng.

#### 4.2.1.4 Adobe Framemaker

Adobe bruker en åpen metadatamodell for de fleste av sine programmer, og mener å støtte denne modellen i hele sin programrekke ettersom nye versjoner blir utgitt. Denne modellen er kalt eXtensible Metadata Platform, XMP [37]. Adobe ønsket ikke å konkurrere med W3Cs standarder, og implementasjonen av XMP er dermed bygd på disse. XMP er en implementasjon av W3Cs RDF i XML. Ved å benytte seg av (relativt) allment aksepterte standarder er det større mulighet for Adobes XMP å bli tatt i bruk av andre selskaper og utviklere. XMP benytter dog kun en delmengde av RDF-spesifikasjonen.

XMP integreres direkte i de binære filene gjennom en struktur kalt XMP Packet (parallell til IP-pakker og lignende). Her pakkes XML-metadataen sammen med en merkelapp (med mer) slik at det skal være enkelt for tredjepartsapplikasjoner å finne XMP-pakken. En forenklet struktur av XML-pakken er vist i Figur 23 (figuren er hentet fra [37])


**Figur 23: XMP-pakke**

### 4.2.2 Mapping

For å kunne lagre ekstrahert metadata må det være definert en mapping mellom den metadata som ekstraheres og metadata som kan lagres i Inspera Content Server. Denne mappingen vil også brukes for å inkapse metadata under eksport og visning, slik at mottakerapplikasjonen gjenkjenner feltene. Eksempelvis må riktig info hentes til de forskjellige properties i WebDAV, slik at denne kan vises i klienten. Dersom det interne feltet for dato for siste endring heter ModifiedDate vil ikke den gjenkjennes som siste endrings-dato i WebDAV-klienter, som forventes at slik info ligger i getlastmodified-feltet. På samme måte må metadata mappes både på import og eksport til Word, slik at metadata kan lagres riktig i systemet den ene veien og vises riktig i Word den andre. Tabell 4 viser en mapping av metadata mellom Word, FrameMaker, WebDAV og Inspera Content Server. Siden WebDAV ikke brukes som transportprotokoll for disse metadata, de hentes rett fra dokumentet ved metadataekstraktorer, er det ikke et problem at all anvendt metadata ikke er mappet til WebDAV-properties. De som faktisk er mappet til WebDAV hentes ut og vises i WebDAV-klienter.

| Microsoft Word | Adobe FrameMaker | WebDAV | Inspera Content Server | Beskrivelse |
|---|---|---|---|---|
| Title | Title | displayname | title | Tittel |
| Subject | Description | | description | Beskrivelse |
| Author | Author | | contributor | Forfatter |
| Keywords | Keywords | | keywords | Nøkkelord |
| Comments | | | comments | Kommentarer til revisjonen |
| Creation Date | CreateDate | creationdate | creationDate | Dato for oppretting |
| Last Save Date | ModifyDate | getlastmodified | lastModified | Dato for siste endring |
| Application | CreatorTool | | sourceApplicationName | Applikasjonen dokumentet opprinnelig er laget i. |
| Revision Number | | | sourceRevisionNumber | Revisjonsnummeret til det opprinnelige dokumentet. |

**Tabell 4: Mapping mellom anvendte metadatastrukturer**

### 4.2.3 Implementasjonsbeskrivelse

Import av metadata foregår i sammenheng med import av dokumenter/filer til innholdsarkivet (se kap. 4.3). Eventuell metadata som kan ekstraheres fra filene vil mappes til tilsvarende metadata i Inspera Content Server. To metadataekstraktorer er implementert:

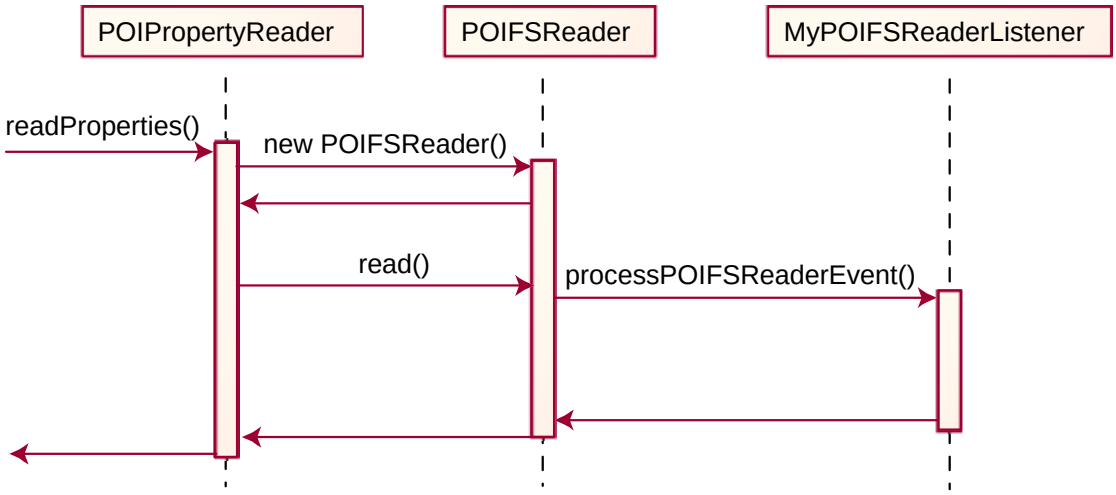
- **POIPropertyReader**. Ekstraherer metadata fra Microsoft Office-filer, inkludert Word.
- **XMPPropertyReader**. Ekstraherer medata fra alle filer som inneholder XMP-metadata, som de alle nyere Adobe-produkter, inkludert FrameMaker.

Begge ekstraktorene baserer seg på en kjent mengde metadata, og dermed en kjent mapping av disse til Inspera Content Server. Håndtering av egendefinert metadata er utenfor skopet. Kildekode for begge klassene er gitt i vedlegg A-5.

#### 4.2.3.1 POIPropertyReader

Til ekstrahering av metadata fra Word-filer benyttes Apache POI [39], som er et open-source prosjekt for javabasert lesing og skriving til Office-filer, for eksempel Word-filer. Prosjektet er ennå ungt, men er kommet langt når det gjelder lesing av metadata (skriving er ikke implementert). POI tilbyr klassen POIFSReader som er ansvarlig for lesing av OLE2CD-formatet. Forskjellige strømmer (delfiler i det interne filsystemet) leses ved å tilordne lyttere som implementerer grensesnittet POIFSReaderListener til dem, og deretter kalle metoden read() på POIFSReader-instansen. Metoden processPOIFSReaderEvent() i lytteren brukes til å prosessere det som blir lest.

POIPropertyReader er implementert i Inspera Content Server og benytter POI for å ekstrahere metadata fra Word-filer. Denne inneholder en metode, readProperties(), som tar inn som parameter binær data (Word-filen) og en ContentRevision som ekstrahert data skal lagres på. Med andre ord må Word-filen være importert til en ContentRevision før POIPropertyReader benyttes. Hvordan dette gjøres er beskrevet i kap. 4.3. readProperties oppretter en instans av POIFSReader og registrerer en lytter av klassen MyPOIFSReaderListener på strømmen SummaryInformation. Denne klassen er privat for POIPropertyReader, og implementerer POIFSReaderListener-grensesnittet. Metoden processPOIFSReaderEvent() i denne klassen står for mappingen av metadata. Her opprettes et objekt av klassen SummaryInformation (fra POI), som tilbyr metoder for å hente all kjent metadata fra OLE2CD-filer. Metadata som hentes lagres på ContentRevision-objektet etter mappingen beskrevet i kap. 4.2.2. Sekvensdiagram for ekstrahering av metadata med POIPropertyReader er vist i Figur 24.




**Figur 24: Ekstrahering av metadata med POIPropertyReader**

Under brukertesten ble ikke Word-importøren benyttet, da filene ble lagret i Microsofts RTF-format. Dermed var det nødvendig å støtte ekstrahering av metadata fra RTF-filer. Dette er en noe enklere sak da RTF er et åpent format. Importøren som ble benyttet for RTF-import støttet ekstrahering av metadata direkte (se kap. 4.3.2)

#### 4.2.3.2 XMPPropertyReader

Det er implementert en XMPPropertyReader for å lese metadata fra XMP-pakker, som benyttet i FrameMaker 7.0 og andre nyere Adobe-produkter. Denne benytter en opensource javabasert XMP-parser konstuert av Knud Kegel [40]. XMPParser har en metode getXpacket() som tar inn en binær datastrøm (filen der det skal søkes etter XMP-pakke) og returnerer en tekstlig datastrøm (XMP, på XML-format).

XMPPropertyReader tilbyr metoden readProperties() for å lese metadata fra en fil dersom denne inneholder en XMP-pakke. Tilsvarende som for POIPropertyReader tar denne inn binær data og revisjonen som metadata skal lagres på. Først opprettes en instans av XMPParser, og metoden getXPacket benyttes for å hente ut XMP fra filen. Dersom XMP er funnet genereres et XML-tre av XMP-teksten, og det søkes i dette etter relevant metadata som så lagres etter mappingen gitt i kap. 4.2.2.

### 4.2.4 Diskusjon

I dette kapitlet er det vist hvordan metadata kan ekstraheres for to av de mest brukte verktøyene for dokumentproduksjon. Muligheten for å gjøre metadatamerking i arbeidsverktøyet i motsetning til under lagring til innholdsarkivet er nok et skritt på veien mot sømløs integrasjon.

Implementasjonen beskrevet i dette kapitlet har fokusert på en kjent mengde metadata. Dermed har ekstraheringen kunnet basere seg på hardkodinger i uthentingsmetodikken, siden metodene for å hente ut disse elementene var kjent. I mange praktiske applikasjoner vil en avgrenset og kjent mengde metadata være tilstrekkelig. Størrelsen og elementene i denne mengden vil kunne variere, men så lenge det er en kjent mengde kan bekreftede metoder for å hente ut denne mengden hardkodes. Det er åpenbart ingen god løsning. Dersom samme system skal brukes i ulike applikasjoner må enten benyttet metadata være de samme fra anvendelse til anvendelse, eller ulike hardkodinger tas i bruk for hver applikasjon. Det er derimot ønskelig at ekstraktoren henter all metadata lagret i filen. Dette er ikke umulig, og implementasjonen kunne greit vært utvidet til å gjøre dette. Problemet er at uthentet metadata må mappes mot metadatastrukturen som benyttes i innholdsarkivet. En løsningsskisse for Inspera Content Server vil være at kjent metadata (tittel, beskrivelse og lignende) mappes direkte, mens ukjent metadata oprettes som navn-verdi-tupler i den interne metadatastrukturen. Hvordan dette skal løses i andre systemer er implementasjonsspesifikt, men disse generelle trekkene bør kunne overføres til andre systemer.

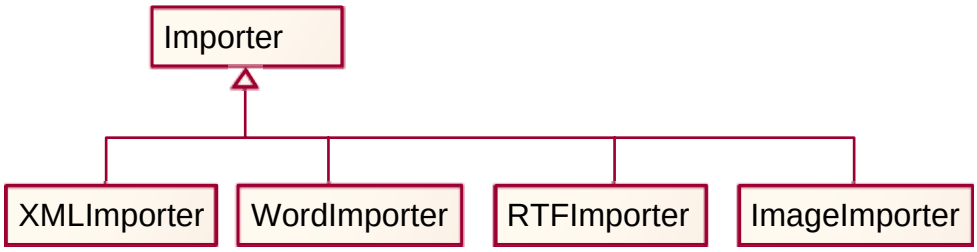
Et annet viktig moment som ikke er vurdert i denne rapporten er eksport av metadata. For at det skal kunne opprettholdes konsistens bør metadata lagret på et innholdselement følge med i alle formater som elementet kan eksporteres til. Dette er ikke noe problem ved enkel XML-basert eksport, men med andre formater blir situasjonen mer komplisert. I tillegg til dette kommer andre innholdstyper, som for eksempel bilder. Støtten for metadata i standard bildeformater som GIF og JPEG er begrenset, dog skal XMP kunne benyttes også i disse formatene. Siden XMP-leseren som er implementert ikke er avhengig av filformatet, kun hvorvidt filen inneholder en XMP-pakke, vil metadata eksportert ved XMP i GIF eller JPEG kunne leses inn igjen.

## *4.3 Import av filer til CMS*

For å integrere innholdsarkivet med arbeidsverktøyene er det en forutsetning at innhold produsert i verktøyet kan leses av innholdsarkivet. For lyd, bilder, video og lignende er ikke dette et særlig problem da disse ikke skal tolkes av arkivet, kun lagres. Men for dokumenter må innholdet og den logiske strukturen i dokumentet parses og ekstraheres. Hvordan dette gjøres avhenger av format. For dokumenter lagret som XML er problemet trivielt, da det kun dreier seg om å kjenne DTD'en og hvordan den mappes til intern struktur. For andre, proprietære formater er problemet noe helt annet. I dette kapitlet vil en proof-of-consept-implementasjon for importering av filer fra Microsoft Word og Adobe Framemaker beskrives. Det er utviklet en ny og generell importstruktur for Inspera Content Server, beskrevet i kap. 4.3.1, som lett kan utvides til å støtte ytterligere formater.

### 4.3.1 Importstruktur

Under implementasjonen var det ønskelig at det ble designet en ny generell importstruktur, som lett kunne utvides til å støtte ytterlige formater. Klassediagram for den nye strukturen er gitt i Figur 25. Strukturen baserer seg rundt en abstrakt superklasse, Importer, og subklasser som representerer faktiske importere for forskjellige formater. Superklassen Importer spesifiserer den generelle importmetoden som må implementeres av alle subklasser. Metoden doImport() er definert som å ta inn binær data og returnere en ContentRevision av korrekt subtype (for eksempel ContentDocument eller ContentSound). Hver av subklassene vil da implementere denne metoden til å importere sitt filformat.




**Figur 25: Klassediagram for ny importstruktur**

For å abstrahere konseptet med ulike importører for forskjellige filformater har superklassen Importer en metode getImporter() som tar som parameter en MIME-type, og returnerer importøren som er definert for denne typen. MIME-type er en Internett-standard for å adskille forskjellige filformater på internett [38]. Dersom klienten ikke spesifiserer MIME-type for opplastet fil, benyttes etternavnet på filen og en tabell som mapper filetternavn til MIME-type. Importer-subklassen som returneres av getImporter() kan så benyttes til å importere filen. På denne måten er det ikke nødvendig å ha adskilt funksjonalitet for å importere ulike filformater, det er abstrahert inn i importstrukturen. Kildekode for nevnte klasser er i appendiks A-5.

Det er implementert enkelte importører basert på den gamle importstrukturen på Inspera Content Server. Disse er: XMLImporter, for import av filer på Insperas XML-format; ImageImporter, for import av enkle bildeformater (GIF, JPEG, BMP); og SoundImporter, for import av enkle lydformater (WAV, AIFF, MP3). I tillegg er tre nye importører implementert, som utgjør ny funksjonalitet for Inspera Content Server: WordImporter og RTFImporter for import fra Microsoft Word og FrameImporter for import fra Adobe FrameMaker. Disse er beskrevet i etterfølgende kapitler.

### 4.3.2 RTFImporter

Microsofts Rich Text Format (RTF, [13]) er et åpent format, og det kan dermed tolkes og importeres basert på dokumentasjonen av formatet. Da det ikke er ansett som nødvendig å finne opp hjulet på ny, benyttes en opensource java-basert RTF-til-XML-konvertør kalt Majix [41]. Denne benytter seg av to brukerdefinerte mappinger under konvertering: En for konvertering fra stiler i RTF-filen til tags på det interne formatet, og en for mapping mellom interne tags og tags i det intene formatet til XML-tags etter egenspesifisert DTD. Det betyr at stilene som benyttes i RTF-filen må være kjente på forhånd, og må ha en kjent mapping til det ønskede XML-formatet. Det interne formatet er ikke editerbart, og dette setter enkelte begrensninger, ved at settet av ulike tagger som kan benyttes er begrenset av settet i det interne formatet, og enkelte av taggene har spesialisert

funksjonalitet. Sistnevnte er begrenset dokumentert, men under implementasjonen ble det konstatert at overskriftstaggene (<h1>, <h2>, etc.) på det interne formatet ble til en seksjonstag (for eksempel <section level="1">) som inkluderte en titteltag (for eksempel <title>). Da Inspera Content Server ikke benytter seg av en slik hierarkisk avsnittsstruktur[9] må strukturen modifiseres ved en XSL-transformasjon etter konvertering.

I RTFImporter gjøres konverteringen i de to steg nevnt i forrige avsnitt. Den private metoden convertFromRTFToDocBook() konverterer en RTF-fil til DocBook-XML-format [42]. Begrunnelsen for at det er valgt å gå innom DocBook er at mappinger fra RTF til DocBook medfølger Majix. Denne er brukt med kun små endringer for å få med all nødvendig informasjon. Eneste endring som er gjort i mappingen er å ta med nivå-informasjon som attributt i seksjons-taggene, slik at de enkelt kan transformeres til overskrifter av korrekt nivå. I tillegg ekstraherer nevnte metode metadata fra RTF-filen ved hjelp av metoder tilbudt i Majix. Metadata innkapsles i XML'en slik at den blir med videre i prosessen. Neste steg er altså å få XML'en over på Inspera Content Servers XML-struktur. Dette gjøres i metoden convertFromDocBookToInsperaXML() ved hjelp av en XSL-transformasjon som bringer XML'en fra DocBook-struktur til Insperas struktur.

Kildekode for RTFImporter er gitt i vedlegg A-5. Mappingsskjemaene fra RTF til DocBook finnes i Majix-distribusjonen og er ikke tatt med her. Stylesheet brukt for å konvertere fra DocBook til Inspera XML er vist i vedlegg B-1.

### 4.3.3 WordImporter

Ved bruk av RTFImporter kan Word-dokumenter lagret som RTF importeres direkte. Det er likevel ønskelig at dokumenter lagret i Words doc-format også skal kunne importeres. POI-prosjektet beskrevet i forbindelse med ekstrahering av metadata fra Word-dokumenter (kap. 4.2.3.1) jobber også med lesing av innhold fra dokumentene. Dog er dette arbeidet i skrivende stund på et så tidlig stadie at det ikke er praktisk å stole på import gjennom POI. Dermed vil man heller la Word selv gjøre konverteringsarbeidet. Det gjøres ved hjelp av java-prosjektet JACOB [43], som gjør det mulig å kalle metoder i COM-applikasjoner, som for eksempel Word. Hvis JACOB benyttes på en Windows-tjener som har Word installert vil det kunne bruke Word direkte til å åpne en fil og lagre denne som RTF. Det er implementert en liten fristående applikasjon som kun har som funksjon å benytte JACOB og Word til å konvertere doc-filer til RTF. Denne kjøres så på en Windows-tjener, og kan kalles fra hovedtjeneren (som ikke nødvendigvis kjører på Windows-platformen) ved hjelp av Java Remote Method Invocation (RMI, [44]). WordImporter sender så den returnerte RTF-filen videre til RTFImporter for import. Kildekode for WordImporter er vist i vedlegg A-5.

### 4.3.4 FMImporter

FMImporter er konstruert for å importere dokumenter lagret i XML fra FrameMaker. Det var på et tidlig stadie planlagt å lage en fullstendig importør for FrameMakers FM-dokumenter, men dette ble etter hvert nedprioritert ettersom det ble tydelig at Word var applikasjonen som ville fokuseres på. Det er likevel implementert en importør for FrameMakers XML-format, for å vise at import av dokumenter fra FrameMaker er

[9] Noe heller ikke Word gjør. Word (i likhet med Inspera Content Server) opererer kun med overskrifter av ulike nivå og avsnitt.

mulig. Siden FrameMaker i denne kontekst lagrer i XML, er det eneste nødvendige å bringe denne XML-strukturen over på Insperas interne format. Det gjøres med en enkel XSL-transformasjon. Et enkelt stylesheet for dette er vist i vedlegg B-2. Dette stylesheet'et er ikke ment å være fullstendig, kun en indikasjon på at denne importmetodikken er gjennomførbar, og hvordan den utføres. Et komplett stylesheet ville måtte mappe alle XML-elementene FrameMaker produserer over på tilsvarende elementer i Inspera Content Server. Kildekode for FMImporter er vist i appendiks A-5.

### 4.3.5 Diskusjon

Designen av den nye importstrukturen stammet fra et ønske om å generalisere import av innhold. Insperas tidligere importfunksjonalitet baserte seg på forskjellige og spesialiserte klasser for hver innholdstype, og var tett integrert i webgrensesnittet. Denne funksjonaliteten kunne ikke gjenbrukes av WebDAV-modulen, så en ny struktur måtte uansett utvikles. Det ble dermed avgjort å konstruere en struktur som kunne benyttes både av WebDAV og av den eksisterende importfunksjonaliteten i webgrensesnittet. Import gjennom WebDAV har fungert som proof-of-concept på at denne importstrukturen fungerer.

Et moment med den nye importstrukturen som må utvikles videre er mappingen av importører til MIME-type. Hver importør kan bare importere en filtype, og MIME-typen bestemmer hvilken importør som skal benyttes til å importere en fil. Det vil ikke finnes en allmenn mapping mellom MIME-type og importør. Avhengig av bruk vil i noen tilfeller en Word-fil ønskes å importeres og konverteres til innhold, mens den i andre tilfeller vil ønskes lagret som binær fil i innholdsarkivet. I prototypen er denne mappingen hardkodet i Import-superklassen. Videre arbeid vil hente denne fra databasen, slik at hver portal som kjører på Inspera Content Server kan ha sin mapping.

Siden Adobe FrameMaker ble nedprioritert i løpet av perioden var hovedfokuset på import av Wordfiler, men FrameMaker-importøren som ble konstruert greier å importere enkle .FM-filer. Import fra Word var tenkt gjort gjennom WordImporter som beskrevet i dette kapitlet. Men under testing viste det seg at JACOB ikke fungerte tilfredsstillende og det bød på problemer å benytte Word til å konvertere fra Word-dokumenter til RTF. Dermed ble dette ikke benyttet i brukertesten, og brukeren lagret og hentet filer som RTF. Tanken bak Word-konverteringen er nok god, men det bør vurderes hvorvidt man skal bruke tid på å videreutvikle denne fullt ut, både siden den innebærer bruk av en ekstra Windows-server (siden applikasjonen kjører på UNIX) og det fungerer like bra å la Word håndtere RTF. Det som virker som den beste løsning er å benytte RTF frem til POI er videreutviklet med bedre støtte for Word. Da vil POI gi tilgang til Word-filens interne strukturer, inkludert innholdet, og det vil forhåpentligvis kunne parses og importeres fullt ut.

Importering av RTF-filer blir gjort ved hjelp av Majix. Selv om Majix viste seg å være relativt enkel å tilpasse og greide å konvertere RTF-filer til Insperas XML-struktur, har denne metoden enkelte mangler. En av de viktigste er metadata. RTF har muligheten til å lagre egendefinert metadata, men Majix henter kun ut en delmengde av de ferdigdefinerte metadata. For å kunne benytte Word til dokumentproduksjon må all metadata kunne importeres og eksporteres til filformatet. Etter å ha studert kildekoden til Majix ble det vurdert at denne kunne utvides til å støtte egendefinert metadata, men at det ville bety en del arbeid. Det er derfor ikke gjort i denne omgang. Majix' strategi med ferdigdefinert mapping mellom stiler og XML-tags er også litt problematisk. Den betyr at egendefinerte

stilnavn i RTF-filen ikke støttes, kun en ferdigdefinert mengde. Det er ikke noe problem dersom det er tilfelle at dokumentene som produseres vil baseres på et definert stilkart, men en mer generell løsning ville være å bygge en XML-struktur basert på stilnavn som tagger. Dermed ville all stilinformasjon kunne overføres til XML, og en XSL-transformasjon benyttes for å føre denne XML-filen over på ønsket DTD. XSL'en ville måtte kjenne stilnavnene for å vite hvilke tags de skulle mappes til, men dette vil uansett være en mer vedlikeholdbar og utvidbar løsning. Men siden det ikke er funnet noen gratis javakomponenter som gjør dette benyttes Majix som en brukbar løsning.

# 5 Evaluering

Denne rapporten har beskrevet en prototype på sømløs integrasjon mellom arbeidsverktøy og et proprietært innholdsarkiv ved hjelp av WebDAV-protokollen. I dette kapitlet vil prototypen evalueres. Det er tre aspekter som er aktuelle for evaluering:

- Sømløst integrert arbeidsmiljø
- Meta-merking i tredjeparts applikasjoner
- XML-integrasjon mellom arbeidsverktøy og Content Management System.

Alle tre punktene er evaluert med hensyn på den tekniske gjennomføringen, det vil si hvorvidt WebDAV og prototypen har løst de tekniske utfordringene på en tilfredsstillende måte. Den tekniske evalueringen er beskrevet i kapittel 5.1. I tillegg er det gjennomført en brukertest hos forlagshuset Aschehoug, som allerede benytter seg av Inspera Content Server til publisering av digitale læremidler. Her er det i hovedsak opplevelsen av den sømløse integrasjonen som testes. Metadatamerking benyttes ikke særlig i dette caset, og XML-integrasjonen er usynlig for brukeren. Begge disse punktene er likevel en forutsetning for at den sømløse integrasjonen skal kunne gjennomføres fullt ut. Brukertesten er beskrevet i kapittel 5.2.

## 5.1 Teknisk evaluering

Prototypen består av fire deler. I tillegg til de tre implementert i forbindelse med denne rapporten; en modul for WebDAV-støtte, ekstraktorer for metadata og importstrukturen, er også den fra før eksisterende implementasjonen på Inspera Content Server en del av helheten. Alle disse må fungere sammen for at prototypen skal fungere som planlagt.

Implementasjonen av WebDAV-støtten ble kun testet mot Microsoft WebFolder, siden det var den klienten som skulle benyttes i brukertesten. Det betyr at implementasjonen kan ha feil og mangler som ikke er nevnt i listen over forenklinger gjort. Disse vil ikke avdekkes før den eventuelt testes mot flere ulike klienter i ulike situasjoner. Likevel er det viktig å huske at siden det kun er en prototype og den ikke er ment å kjøre mot andre klienter er ikke slike feil et ankepunkt mot prototypen, så lenge de lar seg korrigere. Derimot vil den kunne kritiseres dersom det oppdages større feil som bunner i dårlig design, men det antas at slike feil ville dukket opp under testingen mot WebFolder. All funksjonalitet fungerte som forventet, forenklingene tatt i betraktning.

Den nyutviklede importstrukturen var en forutsetning for at opplasting av innhold gjennom WebDAV skulle fungere mot innholdsarkivet. Så vidt WebDAV er bekjent opereres det kun med filer, men innhold pakket i en filstruktur er av ingen nytte for innholdsarkivet. Et problem som dukket opp under testingen av import var dedusering av MIME-type. WebFolder gjorde ikke denne deduseringen, så meldingshodet som skal angi MIME-type ble ikke satt under opplasting. Dermed ble det nødvendig for systemet å selv dedusere MIME-type ut fra en mapping mellom filetternavn og type. Dersom filetternavnet er kjent og det er tilordnet en importør til denne filtypen, skal filen kunne importeres. Importørene som ble implementert tjente to formål; å sørge for at filer opplastet gjennom WebDAV kunne konverteres til innhold, og å vise at denne importstrukturen kan erstatte deler av eksisterende funksjonalitet. Sistnevnte er begrunnelsen for at importører for andre formater som lyd og bilde er implementert, for prototypen og brukertestens formål hadde det holdt med en importør for Word-

dokumenter, men de resterende viste at importøren kunne fungere for alle import-typer som allerede var støttet på Inspera Content Server. Siden WebDAV ved bruk av disse kunne importere alle innholdstyper som fra før var støttet på Inspera Content Server er det ansett at importstrukturen holdt mål for sine formål, både som grunnlag for import gjennom WebDAV men også som en fremtidig løsning for all import i Inspera Content Server.

Metadataekstraktorene ble ikke testet så grundig som ønsket. Bagrunnen for dette er at de i praksis ikke ble benyttet. FrameMaker-vinklingen forsvant til fordel for fokus på Word, og ekstraktoren som henter metadata fra Word-filer viste seg unyttig, siden Word-importørens problemer gjorde at all Word-integrasjon ble gjort med bruk av RTF. Dermed måtte ekstraktorene testes i kunstige omgivelser. Ved forsøk på test-dokumenter så de ut til å ekstrahere angitt metadata, men med problemene diskutert under implemtasjonen. Siden disse ekstraktorene ikke ble anvendt av import-strukturen er det verd å diskutere hvorvidt slik funksjonalitet er nødvendig. Import av metadata fra opplastede filer er åpenbart nødvendig, men dette kan like godt gjøres direkte av importklassene. Det er vurdert at metadataimporten ikke trenger en egen struktur, men kan like godt ligge i importklassene. De to ekstraktorene som er implementerte har likevel livets rett som egne klasser, siden de ikke ekstraherer metadata fra kun en filtype. POIPropertyImporter kan benyttes på alle Office-filer, og kan dermed anvendes av eventuelle fremtidige Excel- eller PowerPoint-importører. Samme gjelder XMPPropertyReader, som kan anvendes for alle filer som kan inneholde XMP-metadata. I første rekke gjelder dette Adobe-filer, men XMP er ment å kunne støttes også i andre utvidbare filtyper som JPEG og lignende.

Som nevnt i diskusjonen til WebDAV-implementasjonen har ikke effektivitet og skalerbarhet vært et mål under desgin og implementasjon. Prototypen er ment for å teste funksjonalitet. Dersom implementasjonen skal tas i bruk på produksjonsnivå bør skalerbarhetsaspektet vurderes. Men hovedparten av arbeidet blir gjort av kjernemetodene utenfor WebDAV-modulen betyr det at hvis disse skalerer vil også WebDAV-støtten skalere. Prosessering i WebDAV-modulen er såpass moderat at det ikke antas at den vil ha særlig effekt. Dersom det ikke holder, kan alternativet med en direkte bruk av intern datamodell i stedet for en egen WebDAV-datamodell benyttes, som skissert i implementasjonsdiskusjonen. Dette fjerner ytterligere prosessering vekk fra WebDAV-modulen.

## 5.2 Bruksevaluering

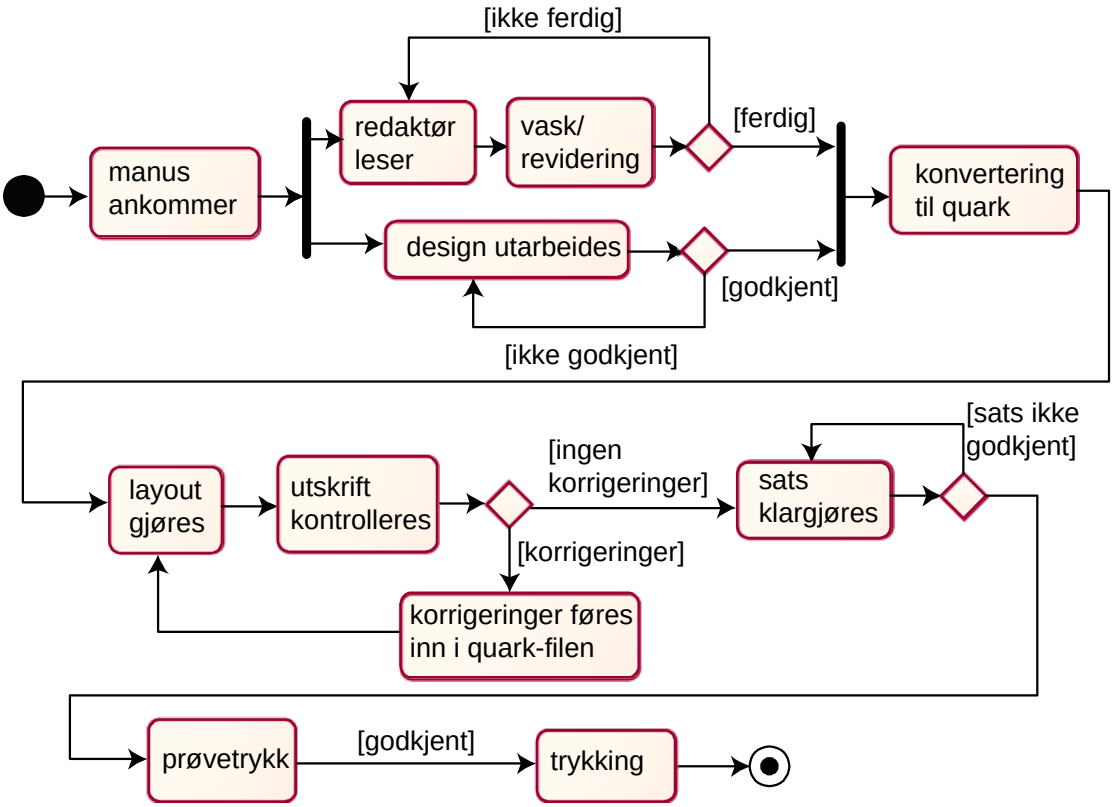
For å evaluere brukbarheten til WebDAV-støttet sømløs integrasjon i en praktisk anvendelse er prototypeimplementasjonen på Inspera Content Server testet av en redaksjonssekretær (heretter: brukeren) hos forlagshuset Aschehoug. En beskrivelse av innholdsproduksjonsprosessen og hennes arbeidsoppgaver er gitt i kapittel 5.2.1. Her omtales både prosessen for trykte og digitale medier. I brukertesten ble det fokusert på digitale medier, da det ved skrivende stund benyttes Inspera Content Server til digital publisering ved Aschehoug og brukeren dermed er kjent med systemet. Omtale av brukertesten er gitt i kapittel 5.2.2, og resultatene er beskrevet i kapittel 5.2.3.

### 5.2.1 Testcase

Forlagshuset Aschehoug er mest kjent som forlegger av trykte medier, men har i den senere tid også satt fokus på digitale medier. Forlaget har produsert flere

innholdsnettsteder, basert på teknologi levert av Inspera. Produksjonsprosessen varierer noe basert på resultatmedie, men fasen for manusvasking og klargjøring er parallell.

Prosessen for produksjon av bøker er gitt i Figur 26, og forklares under. Dokumentet lever i to faser, som markert på figuren: Først som Word-dokument[10], deretter som Quark-dokument under ombrekking. Word-dokumentet har ingen funksjon etter at konverteringen til Quark er gjort, og slettes.



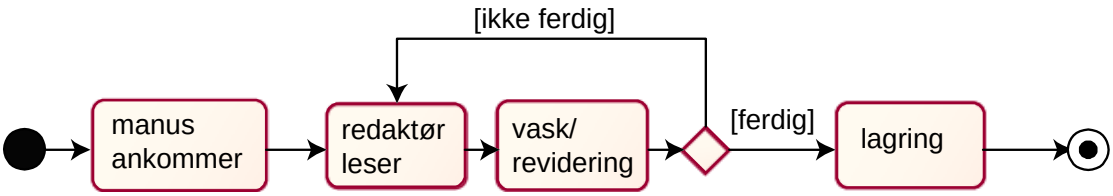

**Figur 26: Bokproduksjon hos Aschehoug, prosessmodell**

- **Manus kommer fra forfatteren.** Forfatter leverer manus i Word-format.
- **Redaktøren leser en papirutskrift.** I denne papirutskriften gjør redaktøren sin korrektur. I tillegg kan han sette opp hvilke stiler som skal benyttes, men kun med indikasjoner i papirutskriften, ikke digitalt.
- **Manus vaskes og revideres.** Vasking av manus innebærer fjerning av illustrasjoner, konsekvens i bruk av mellomrom, blanklinjer og lignende. I tillegg korrekturrettes manus. Stilene indikert av redaktøren inkorporeres også her. Manusvasking gjøres av en sekretær.
- **Design utarbeides.** Dette arbeidet foregår parallelt med bearbeiding av manus. Redaktør og formgiver møtes for å enes om dette, og formgiver utarbeider forslag til godkjenning av redaktøren. Samme gjelder for omslag. En del av arbeidet innebærer utarbeiding av en stilmal for Quark Xpress, programmet som benyttes for ombrekking.

[10] I denne sammenheng er det antatt at forfatter leverer manus som mer eller mindre stilet Word-dokument. Dette kan variere, men ved andre tilfeller vil uansett et stilet Word-dokument fremkomme av manusvaskingen. Forenklet antas det derfor at det alltid opereres med Word-dokumenter.

- **Konvertering til Quark Xpress.** Manus gis til ombrekker som har fått en ombrekkingsmal fra forgiver. Ombrekkeren importer manus i Quark Xpress, og Word-stilene konverteres til stiler i Quark (kun basert på navn, det vil si at en stil med navn "Overskrift1" mappes til en stil med samme navn i Quark). Dette er den eneste stilinformasjonen som tas med (stilinformasjon inkluderer tekstformateringselementer som halvfet, kursiv og lignende).
- **Layout/ombrekking gjøres.** Fra ombrekkingsmalen vil enkelte deler av lay-out'en falle på plass automatisk, men hver side må bearbeides for seg, med tanke på orddeling, mellomrom mellom bokstaver og lignende.
- **Utskrift kontrolleres.** Ombrekker oversender en papirutskrift av Quark-dokumentene slik at redaktør og forfatter kan kontrollere disse. Her kontrolleres ting som korrekturfeil, at all tekst er kommet med på riktige sted og at det er konsekvens i bruk av overskriftsgrader. Det kan være nødvendig med omskrivninger av tekst for å få den til å passe.
- **Korrigeringer av Quark-filen.** Korrigeringene som var gjort på papirutskriften legges inn i Quark.filen, og filen sendes til ombrekker som gjør nye justeringer.
- **Sats klargjøres.** Når redaktøren godkjenner den endelige versjonen klargjør ombrekker den elektroniske satsen, vanligvis Quark-filene eller trykkeklar PDF, som så må godkjennes av prosjektkoordinator.
- **Prøvetrykk.** Prosjektkoordinator sender den elektroniske satsen til trykkeriet som kontrollerer at den er i henhold til deres tekniske krav, og lager et prøvetrykk av boka. Dette sendes så til godkjenning hos forlaget, for å kontrollere at ingen ting har forskjøvet seg eller lignende. Det er fortsatt mulig å gjøre opprettinger i teksten, men vanligvis bare minimale endringer.
- **Trykking.** Boken trykkes, trykkarkene falses, skjæres til og bindes inn. Boken er klar til utsending.

Når målet er et digitalt medium endrer prosessen seg noe. Den første fasen er parallell med bokproduksjon, men ombrekkingsprosessen utgår siden slik publisering baserer seg på maler som benyttes til å stile innholdet. Dermed er prosessen mer parallell, der den ene biten går på produksjon av innholdet og det andre utforming av utseende. I tillegg til dette kommer utforming av nettstedet (navigering, struktur og lignende). I Figur 27 er prosessen for innholdsproduksjon vist. De andre parallelle prosessene er utelatt da de ikke er relevant i denne kontekst. Som det går frem av figuren er denne identisk med første fase under bokproduksjon (Figur 26), men det ferdige dokumentet lagres i databasen i stedet for å gå videre til ombrekking.




**Figur 27: Modell for innholdsproduksjon**

Som testcase i denne brukertesten ble fokusert på den første fasen, fra dokumentet ankommer som Word-fil til det ender opp enten klar til Quark-konvertering eller lagret i innholdsarkivet for digital publisering. Siden Inspera Content Server kun benyttes av Aschehoug ved digital publisering, er det valgt å se på slik publisering i denne brukertesten.

### 5.2.2 Brukertest

Testen ble utført på brukerens kontor. For evaluering av bruk av Inspera Content Server ble brukerens egen datamaskin benyttet, mens bruk av WebDAV måtte gjøres på en egen pc der systemet var implementert. Disse to maskinene var utstyrt med ulike versjoner av Word, men det er ikke ansett at det har hatt noen betydning siden funksjonaliteten som benyttes er til stede i begge utgavene, med unntak av funksjonalitet direkte tilknyttet WebDAV-støtte. Brukeren viste først sine arbeidsoppgaver slik de utføres mot Inspera Content Server, og fikk deretter se og prøve hvordan disse kunne gjøres gjennom Internet Explorer og WebFolder. Arbeidsoppgavene som ble utført var gjennomgang og revidering av ankommet Word-dokument, og etterfølgende lagring av dokumentet i Inspera Content Server i påvente av digital publisering. Det ble også indikert hvilke andre handliger som gjøres i forkant av dette.

Siden Word-importøren ikke fungerte tilfredstillende ble RTF-formatet benyttet i Word. Det er ikke ansett at det har hatt noen betydelig effekt på evalueringen, siden dette er en innstilling som kun gjøres ved første lagring.

#### 5.2.2.1 Brukeren

Brukeren som deltok i brukertesten er redaksjonssekretær. Hun har jobbet noe med Windows og Word og føler seg relativt komfortabel i disse omgivelsene. Hun benytter Windows 2000 og Microsoft Word 97. I tillegg har hun kjennskap til HTML fra tidligere prosjekter, men ikke detaljert. Inspera Content Server har blitt benyttet i flere prosjekter og hun har gjort seg kjent med hvordan hun utfører sine oppgaver i det, men ikke mer.

#### 5.2.2.2 Bruk av Inspera Content Server

Brukerens oppgaver ved publisering til digitalt medie er i første rekke å vaske det ankomne manus, det vil si å sørge for konsekvens i bruk av blanklinjer, avsnittsformater, stiler og lignende. Deretter vil ofte manus gå gjennom revidering hos redaktør og korrekturlesning. Til slutt mottar brukeren det ferdige manus, og hun har ansvar for å lagre dette i innholdsarkivet i påvente av publisering. Lagringen foregår ved at det opprettes et nytt dokument i en mappestruktur angitt av redaktøren. Brukeren vet hvordan man oppretter mapper og lignende, men benytter dette sjelden. Når dokumentet opprettes brukes WYSIWYG-editoren i Inspera Content Server (Figur 8, side 14) til å legget til innholdet. Dette gjøres ved klipp-og-lim, det vil si at hun velger teksten i Word dokumenter, kopierer og limer den inn i WYSIWYG-editoren. Grunnen til at klipp-og-lim blir benyttet er at teksten i Word-dokumentet er korrekturlest, og det må unngås at nye feilkilder introduseres. Enkelte små dokumenter ankommer samlet i et større dokument, og brukeren oppretter da de individuelle dokumentene og klipper-og-limer delene av manuset.

Når teksten limes inn i WYSIWYG-editoren skal tekstformateringer følge med, men det er brukerens erfaring at dette ikke alltid er tilfelle. Dersom dette ikke skjer må brukeren gå gjennom dokumentet i editoren og markere alle stiler på ny. Under brukertesten ble det gjort enkelte forsøk på dette, og i disse ble formateringen korrekt overført. Det antas at hvorvidt dette fungerer avhenger av om Word-dokumentet benytter stiler korrekt. I så fall kan problematikken unngås ved at dokumenter som kopieres fra kun benytter et definert stilsett, noe som ifølge brukeren sjelden er tilfelle på dokumenter som

ankommer fra forfatterne. Brukeren kjenner muligheten til å kontrollere XML-koden til dokumentet for å sjekke at alt er blitt riktig, men synes det er uoversiktlig og benytter det sjelden.

Etter at innholdet i dokumentet er lagt inn, gir brukeren dokumentet et navn etter gitte konvensjoner, eventuelt som spesifisert av redaktør. Annen metadata benyttes sjelden eller ikke, avhengig av prosjektet. Dersom metadata skal benyttes er det indikert i dokumentet, og brukeren vet hvordan metadata legges inn men er ikke komfortabel med grensesnittet.

Revisjoner er heller ikke noe brukeren benytter seg av. Hun kjenner til at det lages revisjoner for hver endring, men ser kun på dokumentet som ett dokument (siste revisjon). Siden hun aldri benytter seg av muligheten til å hente frem eller se på tidligere revisjoner ignorerer hun i praksis det faktum at de finnes. Hun konstaterer at det nok kan være nyttig for redaktører eller andre å kunne se dokumentets historikk, men mener at det for de fleste brukere er enklest og mest hensiktsmessig å se på dokumentet som en enhet og glemme det faktum at det finnes eldre revisjoner lagret.

Helhetlig synes brukeren at det går greit å arbeide mot Inspera Content Server. Hun har lært seg hvordan hun gjør sine oppgaver, og bryr seg ikke med annen funksjonalitet. Hennes hovedinnvendinger går på detaljer i grensesnittet som inkonsekvens i tab-rekkefølge og mangelen på hurtigtaster. Siden hun jobber mye med tastaturet liker hun ikke å måtte bruke musen til å flytte fokus eller trykke på knapper.

#### 5.2.2.3 Bruk av WebFolder

Etter en kort presentasjon av tanken bak sømløs integrasjon ved WebDAV fikk brukeren se hvordan det fungerte i praksis. Hun fikk se hvordan et Word-dokument kunne lagres i Utforsker og fikk ved å kontrollere CMS'et bekreftet at dokumentet faktisk var korrekt importert, konvertert og lagret i innholdsarkivet. Hun fikk også se hvordan et dokument kunne åpnes direkte i Word fra Utforsker og at endringer man gjorde ble reflektert i innholdsarkivet som nye revisjoner når man lagret dokumentet. På samme måte som for WYSIWYG-editoren må overskrifter og lignende være korrekt stilet for at de skal importeres riktig. Siden RTF-importøren baserer seg på et definert stilsett (i prototypen en delmengde av Words ferdigdefinerte stiler) må man sørge for at dokumentet benytter kun disse stilene. Det gjør at manusvaskingen fortsatt innebærer kontroll av stiler.

Sammen ble det utarbeidet en oversikt over hvordan hennes oppgaver kunne løses gjennom Windows Utforsker[11]. Word-dokumentet som ankommer må fortsatt vaskes, deretter kan det lagres i innholdsarkivet ved å åpne det i Utforsker, bla seg frem til riktig mappe (og opprette nye mapper der nødvendig) og lagre dokumentet der, enten fra Word eller ved kopi eller dra-og-slipp med filen. Dette tilsvarer opprettingen av dokumentet i den gamle prosessen og gjøres på et tidligere tidspunkt. Revidering kan nå gjøres direkte i dokumentet. Redaktører eller andre som vil endre dokumentet kan åpne det gjennom sin Utforsker og Word, gjøre sine endringer og lagre, og endringene vil være lagret som en ny revisjon i innholdsarkivet. Siden dokumentet allerede er lagret i innholdsarkivet vil ikke denne fasen avsluttes med at brukeren lagrer det ferdige dokumentet. Når dokumentet anses ferdig og godkjent kan det publiseres digitalt direkte.

[11] Egentlig Explorer og WebFolder, men siden grensesnittet er (tilnærmet) likt Utforsker ble det brukt som begrep i brukertesten, og anvendes også her.

Dersom det mot formodning skulle måtte gjøres endringer på dokumentet etter at det er publisert, vil disse endringene kunne gjøres i Word tilsvarende revideringsfasen, og etter lagring vil endringene automatisk reflekteres på nettstedet der det er publisert.

Siden brukeren ikke benytter metadata i noen særlig grad, var ikke muligheten til å importere metadata direkte fra Word noe som ble vektlagt. De metadata som benyttes (tittel, forfatter og lignende) settes like enkelt rett i Inspera Content Server som i Word. Selvsagt, dersom kun WebFolder benyttes, er det essensielt at metadata kan settes og hentes fra Word siden WebFolder ikke tilbyr noen mulighet for endring av metadata. Som vist i kap. 3.1.3.4 tilbyr WebDAV denne funksjonaliteten, men WebFolder har altså ikke noe grensesnitt for slikt. Heller ikke tidligere revisjoner eller dokumenthistorikk er tilgjengelig. Men som nevnt er brukeren ikke interressert i disse.

### 5.2.3 Resultater

Brukertesten viste at det er mulig for brukeren å gjennomføre sine arbeidsoppgaver kun gjennom Word og WebFolder, med få unntak. Å erstatte CMS'ets grensesnitt med WebFolder begrenser stort hvilke muligheter som finnes, men brukerens arbeidsoppgaver lar seg løse. De to hovedbegrensningene som settes er tilgangen til tidligere revisjoner og til metadata. Brukeren har erklært seg uinteressert i tidligere revisjoner og velger å anse siste revisjon som selve dokumentet. Endringer på dokumentet gjennom Word og WebFolder skaper en ny revisjon og setter denne som aktiv, dermed oppleves det for brukeren som om dokumentet kun eksisterer som én revisjon som endringene gjøres på. Dette bidrar til å abstrahere bort revisjonsfunksjonaliteten som brukeren ikke benytter. Hvis derimot brukeren skulle ønske å se på tidligere revisjoner har WebFolder ingen støtte for det.

En annen markant begrensning i WebFolder er metadata-støtten. Selv om WebDAV-protokollen åpner for editering av metadata er det ikke støttet i WebFolder. Standard metadata som visningsnavn, størrelse, sist endret og lignende hentes og vises, men ingen grensesnitt for å endre dette tilbys. Men Word har god støtte for metadata, og som vist i kap. 4.2.3.1 hentes disse metadata ut fra filen og lagres i innholdsarkivet. Dermed kan metadata lagres fra Word-dokumenter. Men for bilder og andre innholdstyper kan det ikke forventes at metadata kan lagres eller ekstraheres. Dermed betyr det at bruk av metadata for generelt innhold ikke kan revideres gjennom WebFolder. Brukeren så ikke på dette som noe stort problem da hun ikke brukte metadata i noen særlig grad, men for andre prosjekter kan dette by på problemer. Metadataproblematikken bør derfor undersøkes nærmere, også for andre WebDAV-klienter.

Selv om brukertesten har vist at brukerens arbeidsoppgaver kan gjennomføres gjennom WebFolder med enkelte tillempninger, gjenstår et viktig spørsmål: Er det nyttig? Tjener brukeren noe på å benytte seg av denne typen integrasjon i stedet for arbeid direkte mot CMS'et? Her er det hovedsakelig to momenter som spiller inn; hvorvidt arbeidsprosessen forenkles og hvorvidt brukeren trives bedre med den nye arbeidsmåten.

Arbeidsprosessen blir noe endret ved bruk av Utforsker som arbeidsmiljø i motsetning til CMS'et. Opprettingen av dokumenter er forenklet da det ikke er noe skille mellom selve opprettingen av dokumentet og innlegging av innhold. Brukeren slipper å manuelt opprette dokumentet, gi det navn og kopiere inn innhold. Siden opprettingen kun innebærer å kopiere en fil inn i en mappe kan det tenkes at dette kan gjøres direkte av forfatterne, dersom de har tilgang til nett. Forfatteren får oppgitt en adresse til en

nettmappe, og kopierer bare dokumentet sitt inn. Forutsetningen for at dette skal fungere er at dokumentet er stilet riktig slik at det kan importeres. Denne forutsetningen gjelder også hvis brukeren selv skal legge inn dokumentet, men hun antas å være mer kapabel til å kontrollere og stile dokumentet. Hun forteller at forfatterne i dag leverer dokumenter av svært ulik strukturell standard. Dersom forfatterne hadde et ferdig stilsett og beskjed om å benytte det ville det hjulpet. Feilrisikoen ved å la forfatterne selv importere dokumentet er nok for stor, men hvis dokumentene er korrekt stilet når de ankommer brukeren forenkler det hennes arbeid.

For revidering av dokumenter er blir ikke arbeidsprosessen forenklet i noen særlig grad. Det samme må gjøres selv om det gjøres i WYSIWYG-editoren i Inspera Content Server eller om det gjøres i Word. Men det utgjør heller ikke en komplisering av prosessen, det er fortsatt like mange steg ved fremhenting og lagring. Så selv om brukeren var fascinert over hvordan en tilsynelatende vanlig lagring i Word medførte at en oppdatert versjon ble lagret i innholdsarkivet, er dette ikke nødvendigvis noe enklere enn å utføre samme endringer direkte i WYSIWYG-editoren.

Siden arbeidsprosessene for brukeren selv ble ansett å være omtrent tilsvarende ved bruk av WebFolder eller Inspera Content Server avhenger resultatet av brukertesten av hva brukeren syntes om det å arbeide i Utforsker kontra Inspera Content Server. Hovedoppfatningen var at hun var blitt vant til Inspera Content Server og var relativt fornøyd med det, og derfor ikke så noen trang til å endre rutinene sine. Hun syntes det så greit ut å jobbe mot Utforsker siden hun allerede var kjent med arbeidsmiljøet (Windows) og visste hvordan ting gjøres. Altså trenges lite eller ingen ekstra opplæring for å benytte systemet. Dette ga seg utslag i at da hun skulle bli vist hvordan man kunne laste opp en fil med å dra og slippe, ville hun heller gjøre det på sin måte ved å kopiere filen med CTRL-C, Alt-Tab for å skifte til Utforsker-vinduet og CTRL-V for å lime inn. Dette fungerte selvsagt like bra som foreslått metode, og står som et eksempel på at hun lett kunne gå i gang med å benytte dette systemet. Konklusjonen ved bruk av Utforsker som arbeidsmiljø var at det var veldig greit å anvende siden hun var kjent med det og det er brukervennlig, mer så enn Inspera Content Server. Det negative med det nye systemet var bruk av kun Word som editeringsverktøy. Word er et bra og brukervennlig program, og brukeren hadde ingen problemer med å benytte det siden hun allerede var erfaren med det. Problemet var at hun hadde vanskelig for å stole på at dokumentene ble korrekt lagret. Gjennom bruk av Inspera Content Server hadde hun blitt oppmerksom på at enkelte ting ikke ble lagret slik de var ment, da i hovedsak stilinformasjon. Om dette er tilfelle og om det i så fall var grunnet tekniske eller menneskelige feil er irrelevant, poenget er at hun ikke stolte helt på systemet. Dermed hadde hun lært seg at hun kunne benytte forhåndsvisningsfunksjonaliteten i Inspera Content Server til å sjekke at alle stiler var på plass. Denne muligheten fantes ikke i det nye systemet. Eneste kontrollmulighet er å åpne filen på ny i Word, som både er tungvint og ikke helt tilfredsstiller mistanken. Dog vil et slikt problem forsvinne ved bruk av systemet over lengre tid, forutsatt at importen faktisk virker. Brukeren vil da overbevises om at systemet fungerer.

Sammenfattet ga brukertesten ingen åpenbaring i noen retning. Brukeren så at det kunne være greit å jobbe mot Utforsker, men hadde uansett en grei hverdag med Inspera Content Server slik at hun ikke så noen stor grunn til å skifte. Både Inspera Content Server og det nye systemet har sine problemer; Inspera Content Server med brukervennlighet og tilvenning, og Utforsker med begrenset funksjonalitet. Brukeren mente at selv om Utforsker som arbeidsmiljø ikke nødvendigvis var noe for henne,

kunne det være mer nyttig for andre som ikke hadde benyttet Inspera Content Server tidligere og ikke trengte avansert funksjonalitet.

## 5.3 Diskusjon

Implementasjonen er en fungerende prototype på WebDAV-funksjonalitet på Inspera Content Server. Den tekniske evalueringen har vist at prototypen møter sin funksjon, nemlig å kunne evalueres og anvendes til brukertesting. Testing av prototypen viste begrensningene som lå i WebFolder, spesielt med hensyn på egenskaper. Det var antatt at den, i likhet med de fleste andre WebDAV-klienter, hadde støtte for WebDAVs property-modell. Siden Windows Utforsker har et grensesnitt for å vise og endre egenskaper for kjente filtyper (for eksempel Word) er det underlig at ikke dette grensesnittet blir benyttet av WebFolder til å vise og endre WebDAV-properties. Dette bød ikke på problemer for brukertesten siden brukeren uansett ikke benyttet metadata i særlig grad, og det uansett var mulig å lagre metadata i Word som så ble importert. Dog hadde det vært gunstig om WebFolder hadde støttet metadata, siden denne metoden ikke kan benyttes på andre innholdstyper (lyd, bilder) og sømløs integrasjon med bruk av WebFolder dermed blir umulig på prosjekter der metadata brukes aktivt.

Hovedpunktet som ble testet av brukeren var sømløs integrasjon gjennom Word, det vil si bruk av Word direkte mot innholdsarkivet. Her ble det konstatert at selv om det var mulig å integrere Word direkte mot innholdsarkivet, var det enkelte problemer involvert. Hovedproblemet hang sammen med importeringen av dokumenter til innholdsarkivet, der stiler knyttes til logisk struktur i det interne innholdsformatet. Denne metoden gjør at man må påkreve en strikt stilbruk, der kun en definert mengde stiler kan benyttes. Det gjør at hensikten med den sømløse integrasjonen faller litt bort. Sømløs integrasjon med innholdsarkivet skal skjerme brukerne for et komplisert CMS-grensesnitt, og dermed senke bruksterskelen. Men dersom det forutsetter spesiell håndtering i Word (altså stilbruk) heves terskelen for bruk av verktøyet. Riktig bruk av stiler er ikke en selvfølge blant mindre kyndige brukere. I brukertesten poengterte brukeren at forfattere (som potensielt kunne dratt nytte av sømløs integrasjon ved å skjermes for innholdsarkivet) har en svært varierende grad av stilbruk. Korrekt stilbruk kan kontrolleres ved å påkreve at en stilmal følges, og at kun stiler definert i denne malen benyttes. Igjen er dette ikke en ideell løsning siden det potensielt trengs opplæring i å benytte denne stilmalen, og vinningen ved sømløs integrasjon minker. Et annet alternativ er å *anbefale* bruk av en definert stilmal, slik at de som føler seg komfortable med denne anvender den, mens andre kan velge å ikke gjøre det. For de sistnevnte kan det bli problemer med stilene ved import, og manusvaskingen blir mer omfattende enn i de tilfellene der all stilinformasjon er korrekt lagret.

Et aspekt som ikke er vurdert er samforfatterskap. Brukertestens testcase anvendte ikke samforfatterskap, og WebDAVs metoder for samtidighetskontroll er heller ikke implementert i prototypen. En overfladisk evaluering av WebDAVs muligheter for samforfatterskap må da gjøres på bakgrunn av protokollspesifikasjonen. Det er likevel grunn til å berøre dette aspektet, siden samforfatterskap er en del av mange dokumentproduksjonsprosesser. WebDAV løser samtidighetsproblematikken med låser som, dersom de anvendes korrekt, kan eliminere ubevisst overskrivning. Dessverre ser ikke WebFolder ut til å støtte låsing, men Word forsøker å låse dokumentet ved åpning. Hvis Word benyttes som innfallsvinkel til innholdsarkivet kan dermed flere brukere operere på samme dokumentmengde uten å frykte å overskrive hverandres endringer.

Resultatet av brukertesten var tvetydig. Brukeren hadde ikke noe imot arbeidsmåten med sømløs integrasjon som ble testet, men hun hadde heller ikke noe imot sin eksisterende arbeidssituasjon med Inspera Content Server. Dermed så hun ikke noen stor nødvendighet eller trang til å skifte. Dette er ikke ensbetydende med at den sømløse arbeidsmåten ikke er nyttig. Brukerens refleksjoner er selvsagt subjektive, og det er mulig at en annen bruker ville ha sett saken annerledes, spesielt dersom denne brukeren ikke hadde tidligere erfaringer med Inspera Content Server. Da er det mulig at denne brukeren ville sette pris på det mer intuitive grensesnittet man oppnår gjennom sømløs integrasjon siden hun ikke allerede har satt seg inn i Inspera Content Server. Dessverre var det ikke mulighet til å utføre ytterligere brukertester. I tillegg til brukeren må arbeidsoppgaven tas med i betraktning når resultatene fra brukertesten skal vurderes. Det er mulig at andre situasjoner er bedre egnet til sømløs integrasjon. De største begresningene i brukertesten var metadatastøtte i WebFolder og nødvendigheten for strikt stiling i Word. I situasjoner der disse aspektene ikke er aktuelle (med andre ord uten metadata og med kun enkle dokumenter, eventuelt bilder) vil den nye arbeidsmåten få flere fordeler foran Inspera Content Servers grensesnitt. Også i tilfeller med andre arbeidsverktøyer kan situasjonen fortone seg annerledes. I brukertesten ble WebFolder og Word benyttet siden det var disse verktøyene brukeren anvendte til daglig. Andre brukere kan ha andre WebDAV-kompatible arbeidsverktøy som kan fungere bedre.

En aspekt som brukertesten ikke berører er hva Inspera tjener på å implementere WebDAV-støtte, altså hvilke fordeler WebDAV kan gi en CMS-tilbyder. Mulighet for å tilby sømløs integrasjon gjennom Word og WebFolder er allerede vist, men i tillegg kommer alternativ bruk av protokollen. En engangsimplementasjon av WebDAV-protokollen gjør at CMS'et støtter alle WebDAV-klienter, eksisterende og kommende. Nye arbeidsverktøyer som støtter protokollen vil automatisk kunne integreres med Inspera Content Server. Ulike brukere kan benytte de (WebDAV-kompatible) arbeidsverktøyer som best passer deres bruk. Denne iteroperabiliteten og utvibarheten er en av de store fordelene med en åpen, standardisert protokoll.

# 6 Konklusjon

Rapporten har hatt som mål å vise og vurdere en prototype på sømløs integrasjon ved hjelp av WebDAV. Implementasjonen viste at siden funksjonaliteten WebDAV tilbyr allerede fantes i Inspera Content Server trengtes det ingen nyutvikling av kjernefunksjonalitet, kun en ekstra modul som sørget for kommunikasjon med WebDAV-klienter og knyttet forespørsler fra disse til funksjonalitet i kjernen. Det antas at det samme vil gjelde andre CMS/innholdsarkiver, da funksjonaliteten WebDAV tilbyr er basis for all innholdsforvaltning.

Metadataekstraktorene som er implementert har vist at det er mulig å benytte produksjonsverktøyer til merking av metadata, og importere disse til innholdsarkivet. Applikasjonene det er implementert og testet for er Microsoft Word og Adobe FrameMaker, to av de mest brukte verktøyene for innholdsproduksjon. Det vil være mulig å gjøre slik ekstrahering for alle filtyper som lagrer metadata og er mulig å importere til innholdsarkivet, en forutsetning for at de skal kunne brukes til innholdsproduksjon.

Brukbarheten for sømløs integrasjon vil variere fra anvendelse til anvendelse. Situasjonen evaluert i brukertesten ga ingen klare svar. Brukeren var ikke negativ til sømløs integrasjon slik det ble testet, men hadde ingen trang til å gå bort fra sin eksisterende situasjon. Det er naturlig for en bruker som er opplært i et system å nøle med å skifte. At brukeren ikke var negativ til den nye arbeidsmåten kan dermed tyde på at brukere uten tidligere opplæring i et CMS vil se mer postitivt på sømløs integrasjon i forhold til CMS'ets grensesnitt. I tillegg evaluerte brukertesten kun en variant, bruk av Word og WebFolder. WebDAV muliggjør sømløs integrasjon med alle kompatible arbeidsverktøyer. Selv om brukertesten ikke viste seg å være en arbeidsoppgave som var umiddelbart egnet for sømløs integrasjon ga den såpass positive resultater at det er rom til å anta at det finnes andre situasjoner der sømløs integrasjon kan benyttes med hell.

Det er en fordel for bedrifter å støtte WebDAV-protokollen i sine Content Management Systemer. Fra en engangsinvestering gir bedriften sine CMS-brukere mulighet til å benytte alle eksisterende og kommende WebDAV-kompatible produkter direkte mot innholdsarkivet. Som nevnt er det er ikke like aktuelt i alle situasjoner, men ettersom flere og flere produkter støtter protokollen vil antallet tilfeller der sømløs integrasjon er foretrukket øke. Dersom WebDAV fortsetter å vinne terreng vil dette føre til at den vil bli benyttet i stadig flere produkter. Foreløpig støttes den i flere tjenere enn klienter, som nok nettopp er fordi CMS-tilbydere vil tilby sømløs integrasjon (antagelig gjennom WebFolder, siden Windows er den dominerende platformen) til sine kunder. Det er ingen grunn til å tro at denne trenden vil snu, siden det tyder på at WebDAV har klart å etablere seg som en standard.

En naturlig videre utvidning av systemet vil åpenbart være å fullføre implementasjonen til full WebDAV-støtte. Ut fra konklusjonene gjort i rapporten anbefales dette, spesielt siden Inspera nå har en fungerende prototyp som kan utvides. Videre utvidelser enn full WebDAV-støtte kan være Delta-V, versjonshåndtering. Det vil være naturlig for Inspera å tilby dette siden funksjonalitet for revisjoner allerede eksisterer i Inspera Content Server. Men så lenge WebFolder er den definitivt mest brukte WebDAV-klienten og den ikke støtter Delta-V bør ikke dette prioriteres.

# 7 Bibliografi


[1] "What is a Content Management System." URL: http://www.contentmanager.eu.com/history.htm

[2] "Content Management Concepts." URL: http://www.metatorial.com/papers/cm_concepts.asp

[3] M. Nichani. "LCMS = LMS + CMS [RLOs]." URL: http://www.elearningpost.com/features/archives/001022.asp

[4] P. Browning, M. Lowndes. "Content Management Systems: Who needs them?" URL: http://www.ariadne.ac.uk/issue30/techwatch/

[5] L. Ede & A. Lunsford. "Singular Text/Plural Authors: Perspectives on Collaborative Authoring." Southern Illinois University Press, Carbondale, 1990

[6] M. Sharpies. "Adding a Little Structure to Collaborative Writing". CSCW in Practice: An Introduction and Case Studies, Diaper and C. Sanger (Eds.). Springer-Verlag, 1993.

[7] Hjemmeside for Concurrent Versions System. URL: http://www.cvshome.org

[8] "Guide to Workflow Management and Collaborative Authoring." URL: http://www.diffuse.org/workflow.html

[9] A. J. Gilliland-Swetland. "Introduction to Metadata: Setting the Stage." Getty Institute, 2000.

[10] Hjemmeside for MARC Standards, http://www.loc.gov/marc/

[11] R. Lasher & D. Cohen. "A Format for Bibliographic Records." URL: http://www.ietf.org/rfc/rfc1807.txt

[12] Hjemmeside for Dublin Core Metadata Initiative. http://dublincore.org

[13] "Rich Text Format Specification." URL: http://msdn.microsoft.com/library/en-us/dnrtfspec/html/rtfspec.asp

[14] Hjemmeside for Basic Support for Collaborative Work (BSCW). http://bscw.gmd.de/

[15] Hjemmeside for Microsoft Sharepoint Portal Server. URL: http://www.microsoft.com/sharepoint/

[16] Hjemmeside for Inspera Content Server. URL: http://www.inspera.no/

[17] Hjemmeside for mod_dav: a DAV module for Apache. http://www.webdav.org/mod_dav/

[18] Hjemmeside for The K Desktop Environment. http://www.kde.org

[19] Hjemmeside for Goliath: A website management application for MacOS. URL: http://www.webdav.org/goliath/

[20] E. J. Whitehead "Collaborative Authoring on the Web: Introducing WebDAV." URL: http://www.asis.org/Bulletin/Oct-98/webdav.html

[21] E. J. Whitehead & Y. Goland. "WebDAV: A network protocol for remote collaborative authoring on the Web." Proceedings of the Sixth European Conference of Computer Supported Cooperative Work, Copenhagen, Denmark.

[22] E. J. Whitehead. "A brief introduction to WebDAV." URL: http://ftp.ics.uci.edu/pub/ietf/webdav/intro.html

[23] O. Lassila. "Distributed Authoring Scenarios." URL: http://www.ics.uci.edu/~ejw/authoring/scenarios/draft-ietf-webdav-scenarios-00.txt

[24] J. Slein et al. "Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web." URL: http://www.ietf.org/rfc/rfc2291.txt

[25] Y. Goland et al. "HTTP Extensions for Distributed Authoring – WebDAV." URL: http://www.ietf.org/rfc/rfc2518.txt

[26] R. Fielding et al. "Hypertext Transfer Protocol -- HTTP/1.1" URL: http://www.ietf.org/rfc/rfc2616.txt
[27] J. Franks et al. "HTTP Authentication: Basic and Digest Access Authentication." URL: http://www.ietf.org/rfc/rfc2617.txt
[28] Y. Goland. "Why IE's Web Folders are accessed through File/Open." URL: http://lists.w3.org/Archives/Public/w3c-dist-auth/2000JanMar/0247.html
[29] A. Hopmann & L. Lippert. "Additional WebDAV Collection Properties." URL: http://www.ics.uci.edu/~ejw/authoring/props/draft-hopmann-collection-props-00.txt
[30] Hjemmeside for Microsoft Developer Network, MSDN. URL: http://msdn.microsoft.com
[31] "Web Folder Client (MSDAIPP.DLL) List." URL: http://greenbytes.de/tech/webdav/webfolder-client-list.html
[32] Hjemmeside for WebDAV Projects. URL: http://www.webdav.org/projects/
[33] Hjemmeside for Jakarta Slide. URL: http://jakarta.apache.org/slide
[34] Hjemmeside for WebFRM URL: http://www.geocities.com/SiliconValley/Horizon/7772/webrfm.html
[35] Hjemmeside for DAV Explorer. URL: http://www.ics.uci.edu/~webdav
[36] Hjemmeside for SkunkDAV. URL: http://skunkdav.sourceforge.net/
[37] Hjemmeside for Adobe XMP - eXstensible Metadata Platform. URL: http://www.adobe.com/products/xmp
[38] N. Freed & N. Borenstein. "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types." URL: http://www.ietf.org/rfc/rfc2046.txt
[39] Hjemmeside for Apache POI. URL: http://jakarta.apche.com/poi
[40] Hjemmeside for Buero fuer mediendesign. URL: http://kegel-mediendesign.de
[41] Hjemmeside for Tetrasix Majix. URL: http://www.tetrasix.com/
[42] Hjemmeside for DocBook Document Type. URL: http://www.oasis-open.org/docbook/xml/
[43] Hjemmeside for The JACOB Project: A Java-COM Bridge. URL: http://danadler.com/jacob/
[44] Hjemmeside for Java Remote Method Invokation. URL: http://java.sun.com/products/jdk/rmi/

# Vedlegg A: Kildekode

## A-1 no.inspera.db

### WebdavDataAccess

```
package no.inspera.db;

import no.inspera.applications.publish.datamodel.ContentItem;
import no.inspera.applications.publish.util.PublishConstants;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.services.log.LogServices;
import org.apache.log4j.Category;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.*;

/**
 * Handles all data access for the webdav functionality.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: WebdavDataAccess.java,v 1.8 2003/06/02 00:32:21 naimdjon Exp $
 */
public class WebdavDataAccess extends DataAccess{

       private static Category logCat =
LogServices.getCategory(WebdavDataAccess.class);

    /**
     * Singleton instance
     */
       private static WebdavDataAccess dataAccess = null;

    /**
     * The resourcebundle contains all sql.
     */
       private static String bundleName = "WebdavDataAccess";
       private static PropertyResourceBundle bundle;

    /**
     * marketplaceId -> rootResource
     */
    private HashMap rootResources = null;

       /**
     * Static initializers are read when the class is loaded
     */
       static {
           try {
               bundle = getPropertyResourceBundle(bundleName);
           } catch(MissingResourceException e) {
               logCat.debug("Error when initializing Webdav");
               e.printStackTrace(System.err);
           }
       }

       /**
     * Returns an instance of WebdavDataAccess
     */
       public static WebdavDataAccess getDataAccess() {
               if(dataAccess == null)
                       dataAccess = new WebdavDataAccess();
               return dataAccess;
       }

       /**
     * Returns an instance of WebdavDataAccess
     */
       public static WebdavDataAccess getInstance(){
               if(dataAccess == null)
                       dataAccess = new WebdavDataAccess();
               return dataAccess;
```

```
        }

    /**
     * Gets the resource at the given URI. URI is given from marketplaceroot.
     * @param uri URI of resource to fetch.
     * @param marketplaceId MPID of resource.
     * @return WebdavResource at the URI, or null if not found.
     */
    public WebdavResource getResource(String uri, long marketplaceId) {
        logCat.debug("looking for resource at "+uri);
        if (uri.endsWith("/"))
            uri = uri.substring(0,uri.length()-1);
        StringTokenizer st = new StringTokenizer(uri,"/");
        if (rootResources == null) initRootResources();
        WebdavResource current = (WebdavResource)rootResources.get(new
Long(marketplaceId));
        String processedUri = "";
        // todo: det er vel ikke nødvendig å opprette webdavresource for alle?
        while (current != null && st.hasMoreTokens()) {
            String uriPart = st.nextToken();
            logCat.debug("searching for "+uriPart+", processed so far is
"+processedUri);
            current =
getChild(current.getContentItem().getObjectId(),uriPart,current.getUri());
            if (current != null) processedUri += "/"+uriPart;
        }
        logCat.debug("url found: "+processedUri);
        logCat.debug("find fem feil: "+processedUri+"="+uri+"?");
        logCat.debug("resource found? "+(processedUri.equals(uri)));
        if (!processedUri.equals(uri))
            if (uri.lastIndexOf(".")>0) return
getResource(uri.substring(0,uri.lastIndexOf(".")), marketplaceId);
            else return null;
        else return current;
    }

    /**
     * Returns a WebdavResource representing the child of name childUri of the
resource with id parentId, or null
     * if no such resource exists.
     * @param parentId Id of resource to get child of
     * @param childUri Name of child
     * @param parentUri Full URI of parent, used to generate child URI.
     * @return WebdavResource representing the child, or null
     */
    public WebdavResource getChild(long parentId, String childUri, String parentUri) {
        logCat.debug("getting child "+childUri+" for ci "+parentId);
        ContentPresentationDataAccess cpda =
ContentPresentationDataAccess.getInstance();
        ContentItem ci = cpda.getContentItem(parentId);
        if (!ci.getContentType().equals(PublishConstants.OBJECTTYPE_CONTENTFOLDER))
            return null;
        ArrayList children = ci.getItemIdsInFolder();
        WebdavResource child = null;
        int k = 0;
        while (child == null && k<children.size()) {
            if
(cpda.getContentItem(((Long)children.get(k)).longValue()).getName().equals(childUri))
                child = new
WebdavResource(cpda.getContentItem(((Long)children.get(k)).longValue()),parentUri+"/"+
childUri+WebdavResource.getExtension(cpda.getContentItem(((Long)children.get(k)).longV
alue())));
            k++;
        }
        return child;
    }

    /**
     * Gets an Iterator over all the children of this resouce.
     * @param parentId Id of the resource
     * @param parentUri URI of the resource, used to generate child URI
     * @return Iterator over the children.
     */
    public Iterator getChildren(long parentId, String parentUri) {
        ArrayList a = new ArrayList();
        logCat.debug("getting children for ci "+parentId);
```

```
        ContentPresentationDataAccess cpda =
ContentPresentationDataAccess.getInstance();
        ContentItem ci = cpda.getContentItem(parentId);
        if (!ci.getContentType().equals(PublishConstants.OBJECTTYPE_CONTENTFOLDER))
            return a.iterator();
        ArrayList children = ci.getItemIdsInFolder();
        ContentItem child = null;
        for (int i = 0; i < children.size(); i++) {
            child = cpda.getContentItem(((Long) children.get(i)).longValue());
            logCat.debug("adding child: "+child.getName());
            logCat.debug("extension: "+WebdavResource.getExtension(child));
            a.add(new WebdavResource(child,
parentUri+"/"+child.getName()+WebdavResource.getExtension(child)));
        }
        logCat.debug("returning "+a.size()+" children");
        return a.iterator();
    }

    /**
     * Initializes the root resources for all marketplaces from db. Only run once
since the root resources
     * do not change.
     */
    public void initRootResources() {
        rootResources = new HashMap();
        String webdavServletURL = "http://znot:8090/kursweb/webdav"; //TODO: get
webdavservleturl
        Connection con = null;
               PreparedStatement ps = null;
               try {
                       con = getConnection();
                       String query = (String)bundle.getObject("getRootFolders");
                       ps = con.prepareStatement(query);
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                rootResources.put(new Long(rs.getLong("marketplaceid")), new
WebdavResource(rs.getLong("contentitemid"),
webdavServletURL+"/"+rs.getString("marketplacename")));//
(rs.getString("contenttype").equals("contentfolder"))));
            }
        }catch(Exception e) {
            logCat.debug("Exception init'ing root resources: ",e);
            rootResources = new HashMap();
        }finally {
            close(con);
        }
    }
}
```

## A-2 no.inspera.applications.webdav

### WebdavServlet

```
package no.inspera.applications.webdav;

import no.inspera.Marketplace;
import no.inspera.applications.webdav.method.*;
import no.inspera.db.LanguageDataAccess;
import no.inspera.db.MarketplaceDataAccess;
import no.inspera.db.UserDataAccess;
import no.inspera.services.exception.BaseException;
import no.inspera.services.languagenegotiator.LanguageNegotiator;
import no.inspera.services.log.LogServices;
import no.inspera.utils.Parameters;
import org.apache.log4j.Category;
import org.apache.xerces.utils.Base64;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Enumeration;


/**
```

```
 * Webdav servlet
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: WebdavServlet.java,v 1.5 2003/05/05 12:58:14 johnarne Exp $
 */
public class WebdavServlet extends HttpServlet {

    private static Category logCat = LogServices.getCategory(WebdavServlet.class);

    /**
     * Service-method run by the application server.
     *
     * @param request
     * @param response
     * @throws ServletException
     * @throws IOException
     */
    protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        System.out.println("got request");
        Parameters params = new Parameters();
        params.parseInput(request);
        logCat.debug("has mpid: "+(params.getMarketplaceId()>0));
        logCat.debug("path: "+request.getPathInfo());
        String pathInfo = request.getPathInfo();
        if (pathInfo.startsWith("/")) pathInfo = pathInfo.substring(1);
        // finding marketplace
        MarketplaceDataAccess mpda = MarketplaceDataAccess.getInstance();
        Marketplace mp = mpda.getMarketplace(params.getMarketplaceId());
        if (!(params.getMarketplaceId()>0) ||
!(pathInfo.startsWith(mp.getMarketplaceName()))) {
            try {
                Marketplace[] mps =
MarketplaceDataAccess.getInstance().getAllMarketplaces();
                for (int i = 0; i < mps.length; i++) {
                    if (mps[i].getMarketplaceId() > 0 &&
pathInfo.startsWith(mps[i].getMarketplaceName())) {
                        params.setMarketplaceId(mps[i].getMarketplaceId());
                        mp = mps[i];
                        break;
                    }

                }
            } catch (SQLException e) {
                logCat.warn("Couldn't get marketplaces.", e);
            }
        }else {
            mp =
MarketplaceDataAccess.getDataAccess().getMarketplace(params.getMarketplaceId());
        }
        if (!(params.getMarketplaceId()>0)) {
            response.setStatus(404);
            logCat.debug("marketplaceId not specified, returning 404");
            return;
        }
        logCat.debug("marketplaceId: "+params.getMarketplaceId());
        pathInfo = pathInfo.substring(mp.getMarketplaceName().length());
        logCat.debug("path after removing mp: "+pathInfo);
        logCat.debug("userId: "+params.getUserId());
        if (!(params.getUserId()>0)) {
            long userId = -1;
            String auth = request.getHeader("Authorization");
            if (auth != null && !auth.equals("")) {
                logCat.debug("Auth-header: "+auth);
                logCat.debug("credentials: "+auth.substring(6));
                auth = new String(Base64.decode(auth.substring(6).getBytes()));
                logCat.debug("decoded creds: "+auth);
                int i = auth.indexOf(":");
                if (i>0){
                    logCat.debug("username: "+auth.substring(0,i)+" password:
"+auth.substring(i+1));
                    try {
                        Long uId =
UserDataAccess.getInstance().checkPassword(auth.substring(0,i),auth.substring(i+1),mp.
getMarketplaceId(),
LanguageDataAccess.getInstance().getLocaleFromLanguageId(LanguageNegotiator.getHighest
PriorityLanguageId(mp.getMarketplaceId())));
```

```
                        userId = (uId == null?-1:uId.longValue());
                    } catch (BaseException e) {
                        logCat.warn("error checking password", e);
                        // ignore exception, effect is userId == -1
                    }
                }
            }
            if (!(userId>0)){
                logCat.debug("marketplaceId: "+mp.getMarketplaceName());
                response.setStatus(401); //TODO: maybe add digest.
                response.addHeader("WWW-Authenticate","Basic
realm=\""+mp.getMarketplaceName()+"-webdav\"");
                logCat.debug("returning authentication challenge");
                return;
            }
            params.setUserId(userId);
        }
        logCat.debug("userId: "+params.getUserId());
        logCat.debug("got request.");
        logCat.debug("request info:");
        logCat.debug("Method: "+request.getMethod());
        logCat.debug("---headers---");
        Enumeration e = request.getHeaderNames();
        while (e.hasMoreElements()) {
            String headerName = (String)e.nextElement();
            logCat.debug(headerName+": "+request.getHeader(headerName));
        }

        String method = request.getMethod();
        WebdavMethod webdavmethod = null;

        if (method.equalsIgnoreCase("OPTIONS")) {
            webdavmethod = new OptionsMethod(request,response);
        } else if (method.equalsIgnoreCase("PROPFIND")) {
            webdavmethod = new PropFindMethod(request,response);
        } else if (method.equalsIgnoreCase("GET")) {
            webdavmethod = new GetMethod(request,response);
        } else if (method.equalsIgnoreCase("HEAD")) {
            webdavmethod = new HeadMethod(request,response);
        } else if (method.equalsIgnoreCase("MKCOL")) {
            webdavmethod = new MkcolMethod(request,response);
        } else if (method.equalsIgnoreCase("DELETE")) {
            webdavmethod = new DeleteMethod(request,response);
        } else if (method.equalsIgnoreCase("COPY")) {
            webdavmethod = new CopyMethod(request,response);
        } else if (method.equalsIgnoreCase("MOVE")) {
            webdavmethod = new MoveMethod(request,response);
        } else if (method.equalsIgnoreCase("PUT")) {
            webdavmethod = new PutMethod(request,response);
        } else if (method.equalsIgnoreCase("POST")) {
            webdavmethod = new PostMethod(request,response);
        } else {
            logCat.debug("method "+method+" not implemented yet...");
            response.setStatus(WebdavStatus.SC_NOT_IMPLEMENTED);
        }

        if (webdavmethod != null)
            try {
                webdavmethod.run();
            } catch (WebdavException e1) {
                logCat.warn("Exception running method:", e1);
            }
    }
}
```

### WebdavStatus

```
package no.inspera.applications.webdav;

import java.util.Hashtable;

/**
 * Statuscodes used by webdav
 *
 * @version $Id: WebdavStatus.java,v 1.2 2003/04/30 14:58:57 johnarne Exp $
 */
public class WebdavStatus {
```

```
public static final int SC_CONTINUE = 100;
public static final int SC_SWITCHING_PROTOCOLS = 101;
public static final int SC_PROCESSING = 102;

public static final int SC_OK = 200;
public static final int SC_CREATED = 201;
public static final int SC_ACCEPTED = 202;
public static final int SC_NON_AUTHORITATIVE_INFORMATION = 203;
public static final int SC_NO_CONTENT = 204;
public static final int SC_RESET_CONTENT = 205;
public static final int SC_PARTIAL_CONTENT = 206;
public static final int SC_MULTI_STATUS = 207;
// This one colides with HTTP 1.1
// "207 Parital Update OK"

public static final int SC_MULTIPLE_CHOICES = 300;
public static final int SC_MOVED_PERMANENTLY = 301;
public static final int SC_MOVED_TEMPORARILY = 302;
public static final int SC_SEE_OTHER = 303;
public static final int SC_NOT_MODIFIED = 304;
public static final int SC_USE_PROXY = 305;

public static final int SC_BAD_REQUEST = 400;
public static final int SC_UNAUTHORIZED = 401;
public static final int SC_PAYMENT_REQUIRED = 402;
public static final int SC_FORBIDDEN = 403;
public static final int SC_NOT_FOUND = 404;
public static final int SC_METHOD_NOT_ALLOWED = 405;
public static final int SC_NOT_ACCEPTABLE = 406;
public static final int SC_PROXY_AUTHENTICATION_REQUIRED = 407;
public static final int SC_REQUEST_TIMEOUT = 408;
public static final int SC_CONFLICT = 409;
public static final int SC_GONE = 410;
public static final int SC_LENGTH_REQUIRED = 411;
public static final int SC_PRECONDITION_FAILED = 412;
public static final int SC_REQUEST_TOO_LONG = 413;
public static final int SC_UNSUPPORTED_MEDIA_TYPE = 415;
public static final int SC_REQUESTED_RANGE_NOT_SATISFIABLE = 416;
public static final int SC_EXPECTATION_FAILED = 417;
//public static final int SC_UNPROCESSABLE_ENTITY = 418;
// This one colides with HTTP 1.1
// "418 Reauthentication Required"
public static final int SC_INSUFFICIENT_SPACE_ON_RESOURCE = 419;
// This one colides with HTTP 1.1
// "419 Proxy Reauthentication Required"
public static final int SC_METHOD_FAILURE = 420;
public static final int SC_UNPROCESSABLE_ENTITY = 422;
public static final int SC_LOCKED = 423;
public static final int SC_FAILED_DEPENDENCY = 424;

public static final int SC_INTERNAL_SERVER_ERROR = 500;
public static final int SC_NOT_IMPLEMENTED = 501;
public static final int SC_BAD_GATEWAY = 502;
public static final int SC_SERVICE_UNAVAILABLE = 503;
public static final int SC_GATEWAY_TIMEOUT = 504;
public static final int SC_HTTP_VERSION_NOT_SUPPORTED = 505;
public static final int SC_INSUFFICIENT_STORAGE = 507;


private static Hashtable mapStatusCodes = new Hashtable();

static {
    // HTTP 1.0 Server status codes -- see RFC 1945
    addStatusCodeMap(SC_OK, "OK");
    addStatusCodeMap(SC_CREATED, "Created");
    addStatusCodeMap(SC_ACCEPTED, "Accepted");
    addStatusCodeMap(SC_NO_CONTENT, "No Content");
    addStatusCodeMap(SC_MOVED_PERMANENTLY, "Moved Permanently");
    addStatusCodeMap(SC_MOVED_TEMPORARILY, "Moved Temporarily");
    addStatusCodeMap(SC_NOT_MODIFIED, "Not Modified");
    addStatusCodeMap(SC_BAD_REQUEST, "Bad Request");
    addStatusCodeMap(SC_UNAUTHORIZED, "Unauthorized");
    addStatusCodeMap(SC_FORBIDDEN, "Forbidden");
    addStatusCodeMap(SC_NOT_FOUND, "Not Found");
    addStatusCodeMap(SC_INTERNAL_SERVER_ERROR, "Internal Server Error");
    addStatusCodeMap(SC_NOT_IMPLEMENTED, "Not Implemented");
```

```
        addStatusCodeMap(SC_BAD_GATEWAY, "Bad Gateway");
        addStatusCodeMap(SC_SERVICE_UNAVAILABLE, "Service Unavailable");

        // HTTP 1.1 Server status codes -- see RFC 2048
        addStatusCodeMap(SC_CONTINUE, "Continue");
        addStatusCodeMap(SC_METHOD_NOT_ALLOWED, "Method Not Allowed");
        addStatusCodeMap(SC_CONFLICT, "Conflict");
        addStatusCodeMap(SC_PRECONDITION_FAILED, "Precondition Failed");
        addStatusCodeMap(SC_REQUEST_TOO_LONG, "Request Too Long");
        addStatusCodeMap(SC_UNSUPPORTED_MEDIA_TYPE, "Unsupported Media Type");

        addStatusCodeMap(SC_SWITCHING_PROTOCOLS, "Switching Protocols");
        addStatusCodeMap(SC_NON_AUTHORITATIVE_INFORMATION,
                         "Non Authoritative Information");
        addStatusCodeMap(SC_RESET_CONTENT, "Reset Content");
        addStatusCodeMap(SC_GATEWAY_TIMEOUT, "Gateway Timeout");
        addStatusCodeMap(SC_HTTP_VERSION_NOT_SUPPORTED,
                         "Http Version Not Supported");

        // WebDAV Server-specific status codes
        addStatusCodeMap(SC_PROCESSING, "Processing");
        addStatusCodeMap(SC_MULTI_STATUS, "Multi-Status");
        addStatusCodeMap(SC_UNPROCESSABLE_ENTITY, "Unprocessable Entity");
        addStatusCodeMap(SC_INSUFFICIENT_SPACE_ON_RESOURCE,
                         "Insufficient Space On Resource");
        addStatusCodeMap(SC_METHOD_FAILURE, "Method Failure");
        addStatusCodeMap(SC_LOCKED, "Locked");
        addStatusCodeMap(SC_INSUFFICIENT_STORAGE , "Insufficient Storage");
        addStatusCodeMap(SC_FAILED_DEPENDENCY, "Failed Dependency");
    }

    public static String getStatusText(int nHttpStatusCode) {
        Integer intKey = new Integer(nHttpStatusCode);

        if (!mapStatusCodes.containsKey(intKey)) {
            // No information
            return null;

        } else {
            return (String) mapStatusCodes.get(intKey);
        }
    }

    private static void addStatusCodeMap(int nKey, String strVal) {
        mapStatusCodes.put(new Integer(nKey), strVal);
    }
}
```

### WebdavException

```
package no.inspera.applications.webdav;

import no.inspera.services.exception.BaseException;

import java.util.Locale;


/**
 * Webdav exception class.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: WebdavException.java,v 1.3 2003/04/30 14:58:57 johnarne Exp $
 */
public class WebdavException extends BaseException {


    /** Creates a new WebDAVException.
     *
     * @param messageId   A key to a property-file which will store localized versions
of the message to show the user, affects the output from getUserErrorMessage().
     * @param locale      The Locale to get the error message to show end user
     * @param originator  A Class-object representing the originator of the
error/exception, used for logging and tracing
     */
    public WebdavException(String messageId, Locale locale, Class originator) {
        super(messageId, locale, originator);
    }
```

```
    /** Creates a new WebDAVException with the originator e as the previous exception
in the chain.
     *
     * @param messageId   A key to a property-file which will store localized versions
of the message to show the user, affects the output from getUserErrorMessage().
     * @param locale      The Locale to get the error message to show end user
     * @param originator  A Class-object representing the originator of the
error/exception, used for logging and tracing
     */
    public WebdavException(String messageId, Locale locale, Class originator,Exception
e) {
        super(messageId, locale, originator,e);
    }

}
```

### MultiStatusException

```
package no.inspera.applications.webdav;

import java.util.ArrayList;
import java.util.Iterator;

/**
 * Container for a multistatus. Contains a list of statuscodes and corresponding URIs.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: MultistatusException.java,v 1.2 2003/04/30 14:58:57 johnarne Exp $
 */
public class MultiStatusException extends Throwable{

    /**
     * List of Ex-objects, the exceptions.
     */
    private ArrayList exs = null;

    /**
     * Default constructor.
     */
    public MultiStatusException() {
        exs = new ArrayList();
    }

    /**
     * Adds a new status to this multistatus.
     *
     * @param statusCode The status
     * @param uri URI of the resource that caused the status
     */
    public void addException(int statusCode, String uri) {
        exs.add(new Ex(statusCode, uri));
    }

    /**
     * @return an Iterator over the exceptions.
     */
    public Iterator getExceptions() {
        return exs.iterator();
    }

    /**
     * Private class used to represent a status-uri pair.
     */
    public class Ex {
        public int statusCode;
        public String uri;
        public Ex(int statusCode, String uri) {
            this.statusCode = statusCode;
            this.uri = uri;
        }
    }

}
```

## A-3 no.inspera.applications.webdav.datamodel

### WebdavResource

```
package no.inspera.applications.webdav.datamodel;

import no.inspera.applications.publish.datamodel.ContentItem;
import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.applications.publish.util.ImportFileWriter;
import no.inspera.applications.publish.util.PublishConstants;
import no.inspera.applications.webdav.datamodel.ResourceProperty;
import no.inspera.db.ContentPresentationDataAccess;
import no.inspera.db.LanguageDataAccess;
import no.inspera.services.log.LogServices;

import java.text.SimpleDateFormat;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Locale;


/**
 * Wrapper for a webdav resource. The object is a ContentItem.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 */
public class WebdavResource {

    private static org.apache.log4j.Category logCat =
LogServices.getCategory(WebdavResource.class);

    /** uri of this resource */
    private String uri;

    /** contentitem that tis resource represents */
    private ContentItem ci = null;

    /**
     * Constructor
     * @param contentItemId Id of the ContentItem
     * @param uri URI of the ContentItem
     */
    public WebdavResource(long contentItemId, String uri) {
        this.ci =
ContentPresentationDataAccess.getInstance().getContentItem(contentItemId);
        this.uri = uri;
    }

    /**
     * Constructor
     * @param ci ContentItem
     * @param uri URI of the ContentItem
     */
    public WebdavResource(ContentItem ci, String uri) {
        this.ci = ci;
        this.uri = uri;
    }

    /**
     * @return true if this resource is a collection
     */
    public boolean isCollection() {
        return ci.getContentType().equals(PublishConstants.OBJECTTYPE_CONTENTFOLDER);
    }

    /**
     * @return The ContentItem this resource represents
     */
    public ContentItem getContentItem() {
        return ci;
    }

    /**
     * @return the URI of this resource
     */
    public String getUri() {
        return uri;
```

```
    }

    /**
     * Gets the properties specified in the set, or all if null. The set should
contain ResourceProperty-objects
     * with name and namespace set.
     * @param names The properties that should be retrieved or null for all
     * @param languageId Language languagedepent properties should use
     * @return HashMap of ResourceProperty
     */
    public HashMap getProperties(HashSet names, long languageId) {
        HashMap h = new HashMap();
        SimpleDateFormat isoDate = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss-00:00",
Locale.US);
        SimpleDateFormat httpDate = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss
z", Locale.US);
        ContentRevision cr =
ContentPresentationDataAccess.getInstance().getLiveContentRevisionWithEffectiveLanguag
e(ci.getObjectId(), languageId, ci.getMarketplaceId());
        if (names == null || names.contains(ResourceProperty.CREATIONDATE)){
            // dav:creationdate
            String dateString = isoDate.format(cr.getCreationDate().getTime());
            logCat.debug("creationdate: "+dateString);
            h.put(ResourceProperty.CREATIONDATE, new
ResourceProperty("creationdate",dateString,"DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.DISPLAYNAME)){
            // dav:displayname
            String title = cr.getTitle()+getExtension(cr);
            logCat.debug("title: "+title);
            h.put(ResourceProperty.DISPLAYNAME, new
ResourceProperty("displayname",title,"DAV:"));

        }
        if (names == null || names.contains(ResourceProperty.GETCONTENTLANGUAGE)){
            Locale locale =
LanguageDataAccess.getDataAccess().getLocaleFromLanguageId(languageId);
            h.put(ResourceProperty.GETCONTENTLANGUAGE, new
ResourceProperty("getcontentlanguage",locale.toString(), "DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.GETCONTENTLENGTH)){
            long length = 0;
            if (ci.getLiveContentRevision(languageId)!=null)
                length = ci.getLiveContentRevision(languageId).getContentSize();
            logCat.debug("getcontentlength: "+length);
            if (length <= 0) length = 100;
            h.put(ResourceProperty.GETCONTENTLENGTH, new
ResourceProperty("getcontentlength",""+length,"DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.GETCONTENTTYPE)){
            String mimeType = cr.getMimeType();
            h.put(ResourceProperty.GETCONTENTTYPE, new
ResourceProperty("getcontenttype", mimeType==null?"text/xml":mimeType, "DAV:"));
        }
        //if (names == null || names.contains(ResourceProperty.GETETAG)){
        //    // TODO dav:getetag
        //}
        if (names == null || names.contains(ResourceProperty.GETLASTMODIFIED)){
            // dav:getlastmodified
            String dateString = httpDate.format(cr.getLastModified().getTime());
            logCat.debug("lastmodified: "+dateString);
            h.put(ResourceProperty.GETLASTMODIFIED, new
ResourceProperty("getlastmodified", dateString, "DAV:"));
        }
        //if (names == null || names.contains(ResourceProperty.LOCKDISCOVERY)){
        //    // TODO dav:lockdiscovery
        //}
        if (names == null || names.contains(ResourceProperty.RESOURCETYPE)){
            // dav:resourcetype
            String resourceType = "";
            if (ci.getContentType().equals(PublishConstants.OBJECTTYPE_CONTENTFOLDER))
                resourceType = "<collection/>";
            logCat.debug("resourcetype: "+resourceType);
            h.put(ResourceProperty.RESOURCETYPE, new
ResourceProperty("resourcetype",resourceType,"DAV:"));
        }
        //if (names == null || names.contains(ResourceProperty.SOURCE)){
```

```
        //    // TODO dav:source
        //}
        //if (names == null || names.contains(ResourceProperty.SUPPORTEDLOCK)){
        //    // TODO dav:supportedlock
        //}
        if (names == null || names.contains(ResourceProperty.NAME)){ // this and the
following are ms-spesific dav: properties.
            h.put(ResourceProperty.NAME, new ResourceProperty("name",ci.getName(),
"DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.PARENTNAME)){
            if (ci.getParentFolder() != null)
                h.put(ResourceProperty.PARENTNAME, new
ResourceProperty("parentname",ci.getParentFolder().getName(), "DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.HREF)){
            h.put(ResourceProperty.HREF, new ResourceProperty("href",uri, "DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.ISHIDDEN)){
            h.put(ResourceProperty.ISHIDDEN, new
ResourceProperty("ishidden","0","DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.ISREADONLY)){
            h.put(ResourceProperty.ISREADONLY, new
ResourceProperty("isreadonly","0","DAV:"));
        }
        //if (names == null || names.contains(ResourceProperty.CONTENTCLASS)){
        //
        //}
        //if (names == null || names.contains(ResourceProperty.LASTACCESSED)){
        //
        //}
        if (names == null || names.contains(ResourceProperty.ISCOLLECTION)){
            h.put(ResourceProperty.ISCOLLECTION, new
ResourceProperty("iscollection",ci.getContentType().equals(PublishConstants.OBJECTTYPE
_CONTENTFOLDER)?"1":"0","DAV:"));
        }
        if (names == null || names.contains(ResourceProperty.ISSTRUCTUREDDOCUMENT)){
            h.put(ResourceProperty.ISSTRUCTUREDDOCUMENT, new
ResourceProperty("isstructureddocument","0","DAV:"));
        }
        //if (names == null || names.contains(ResourceProperty.DEFAULTDOCUMENT)){
        //
        //}
        //if (names == null || names.contains(ResourceProperty.ISROOT)){
        //
        //}
        {
            // todo: other properties
        }
        return h;
    }

    /**
     * Gets extension to use for displaying filename.
     * @param cr The revision
     * @return extensionstring starting with .
     */
    public static String getExtension(ContentRevision cr){
        String st = "";
        if (cr.getMimeType() != null &&
!cr.getObjectType().equals(PublishConstants.OBJECTTYPE_CONTENTFOLDER) &&
!cr.getObjectType().equals(PublishConstants.OBJECTTYPE_CONTENTURL))
            if (cr.getMimeType().equals("text/plain"))
                st = ".rtf";
            else
                st = "."+ImportFileWriter.getExtension(cr.getMimeType());
        logCat.debug("returning extension "+st);
        return st;
    }

    public static String getExtension(ContentItem ci){
        return
getExtension(ContentPresentationDataAccess.getInstance().getLiveContentRevisionWithEff
ectiveLanguage(ci.getObjectId(), ci.getLanguageId(), ci.getMarketplaceId()));
    }
}
```

### ResourceProperty

```
package no.inspera.applications.webdav.datamodel;

import no.inspera.services.log.LogServices;

import java.io.Serializable;

/**
 * Represents a webdav property.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: ResourceProperty.java,v 1.4 2003/04/30 14:58:57 johnarne Exp $
 */

public final class ResourceProperty implements Serializable, Cloneable {
    org.apache.log4j.Category logCat =
LogServices.getCategory(ResourceProperty.class);


    public static final String DEFAULT_NAMESPACE = "DAV:";
    public static final String INSPERA_NAMESPACE = "http://www.inspera.no/webdav-ns";

    // the following are static constants of the dav properties, used for comparison.
    public static final ResourceProperty CREATIONDATE = new
ResourceProperty("creationdate", null, "DAV:");
    public static final ResourceProperty DISPLAYNAME = new
ResourceProperty("displayname", null, "DAV:");
    public static final ResourceProperty GETCONTENTLANGUAGE = new
ResourceProperty("getcontentlanguage", null, "DAV:");
    public static final ResourceProperty GETCONTENTLENGTH = new
ResourceProperty("getcontentlength", null, "DAV:");
    public static final ResourceProperty GETCONTENTTYPE = new
ResourceProperty("getcontenttype", null, "DAV:");
    public static final ResourceProperty GETETAG = new ResourceProperty("getetag",
null, "DAV:");
    public static final ResourceProperty GETLASTMODIFIED = new
ResourceProperty("getlastmodified", null, "DAV:");
    public static final ResourceProperty LOCKDISCOVERY = new
ResourceProperty("lockdiscovery", null, "DAV:");
    public static final ResourceProperty RESOURCETYPE = new
ResourceProperty("resourcetype", null, "DAV:");
    public static final ResourceProperty SOURCE = new ResourceProperty("source", null,
"DAV:");
    public static final ResourceProperty SUPPORTEDLOCK = new
ResourceProperty("supportedlock", null, "DAV:");
    // MS-specific properties
    public static final ResourceProperty CONTENTCLASS = new
ResourceProperty("contentclass", null, "DAV:");
    public static final ResourceProperty DEFAULTDOCUMENT = new
ResourceProperty("defaultdocument", null, "DAV:");
    public static final ResourceProperty HREF = new ResourceProperty("href", null,
"DAV:");
    public static final ResourceProperty ISCOLLECTION = new
ResourceProperty("iscollection", null, "DAV:");
    public static final ResourceProperty ISHIDDEN = new ResourceProperty("ishidden",
null, "DAV:");
    public static final ResourceProperty ISREADONLY = new
ResourceProperty("isreadonly", null, "DAV:");
    public static final ResourceProperty ISROOT = new ResourceProperty("isroot", null,
"DAV:");
    public static final ResourceProperty ISSTRUCTUREDDOCUMENT = new
ResourceProperty("isstructureddocument", null, "DAV:");
    public static final ResourceProperty LASTACCESSED = new
ResourceProperty("lastaccessed", null, "DAV:");
    public static final ResourceProperty NAME = new ResourceProperty("name", null,
"DAV:");
    public static final ResourceProperty PARENTNAME = new
ResourceProperty("parentname", null, "DAV:");

    /**
     * Constructor.
     *
     * @param name Name
     * @param value Value
     */
```

```
    public ResourceProperty(String name, String value) {
        setName(name);
        setValue(value);
        this.namespace = DEFAULT_NAMESPACE;
    }


   /**
     * Constructor.
     *
     * @param name Name
     * @param value Value
     * @param namespace Namespace
     */
    public ResourceProperty(String name, String value, String namespace) {
        this(name, value);
        setNamespace(namespace);
    }

    /**
     * ResourceProperty name.
     */
    private String name;


    /**
     * Namespace of the property.
     */
    private String namespace;


    /**
     * ResourceProperty value.
     */
    private String value;

    /**
     * ResourceProperty name accessor.
     *
     * @return String property name
     */
    public String getName() {
        return this.name;
    }


    /**
     * ResourceProperty name mutator.
     *
     * @param name Name
     */
    void setName(String name) {
        if (name == null) {
            this.name = new String();
        } else {
            this.name = name;
        }
    }


    /**
     * Namespace accessor.
     *
     * @return String definition
     */
    public String getNamespace() {
        return this.namespace;
    }


    /**
     * Namespace mutator.
     *
     * @param namespace Namepsace
     */
    void setNamespace(String namespace) {
        if (namespace == null) {
```

```
            this.namespace = "";
        } else {
            this.namespace = namespace;
        }
    }


    /**
     * Value accessor.
     *
     * @return String value
     */
    public String getValue() {
        return value;
    }


    /**
     * Value mutator.
     *
     * @param value Value
     */
    void setValue(String value) {
        if (value == null) {
            this.value = new String();
        } else {
            this.value = value;
        }
    }

    /**
     * Clone.
     *
     * @return Object clone
     */
    ResourceProperty cloneObject() {
        ResourceProperty result = null;
        try {
            result = (ResourceProperty) super.clone();
        } catch(CloneNotSupportedException e) {
        }
        return result;
    }


    /**
     * Hash Code.
     *
     * @return int Hash code value
     */
    public int hashCode() {
        return getName().hashCode();
    }


    /**
     * String representation of the permission.
     * <p/>
     * Format : ObjectUri-SubjectUri-ActionUri-InheritanceFlag
     *
     * @return String String representation
     */
    public String toString() {
        return getName();
    }


    /**
     * Equals.
     *
     * @param obj Object to test
     * @return boolean True if the two object are equal :
     * <li>obj is of type NodeProperty and is not null</li>
     * <li>The property names are equal</li>
     */
    public boolean equals(Object obj) {
        boolean result = false;
```

```
        if ((obj != null) && (obj instanceof ResourceProperty)) {
            ResourceProperty property = (ResourceProperty) obj;
            //logCat.debug("comparing objs: name1:"+this.name+"
name2:"+property.getName()+" ns1:"+this.namespace+" ns2:"+property.getNamespace());
            result = (this.name.equals(property.getName())) &&
(this.namespace.equals(property.getNamespace()));
        }
        return result;
    }


}
```

## A-4 no.inspera.applications.webdav.method

### WebdavMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.Marketplace;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.db.MarketplaceDataAccess;
import no.inspera.services.log.LogServices;
import no.inspera.utils.Parameters;
import org.apache.log4j.Category;
import org.w3c.dom.Document;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import java.io.IOException;
import java.io.InputStream;
import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;


/**
 * WebDAV method. Based on corresponding class in Jakarta Slide.
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: WebdavMethod.java,v 1.4 2003/04/30 14:59:40 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 * @author Christopher Lenz
 */
public abstract class WebdavMethod {

    public static final String INSPERA_NAMESPACE = "http://www.inspera.no/webdav-ns/";
    public static final String INSPERA_NAMESPACE_ABBREV = "I";

    public static final int DEPTH_LIMIT = 3;

    private static Category logCat = LogServices.getCategory(WebdavMethod.class);

    /**
     * Requested Uri.
     */
    protected String requestUri;


    /**
     * Servlet request.
     */
    protected HttpServletRequest req;


    /**
     * Servlet response.
     */
    protected HttpServletResponse resp;
```

```
    /**
     * Request body.
     */
    protected String requestBody;

    protected Parameters params;


    /**
     * Method constructor.
     */
    public WebdavMethod(HttpServletRequest req, HttpServletResponse resp) {

        this.req = req;
        this.resp = resp;

        params = new Parameters();
        params.parseInput(req);

        String relativePath = req.getPathInfo();
        Marketplace mp =
MarketplaceDataAccess.getDataAccess().getMarketplace(params.getMarketplaceId());
        if (mp != null && relativePath.startsWith("/"+mp.getMarketplaceName()))
            relativePath = relativePath.substring(mp.getMarketplaceName().length()+1);

        this.requestUri = relativePath;

    }

    /**
     * Exceute method.
     *
     * @exception WebdavException
     */
    public void run() throws WebdavException {

        try {
            logCat.debug("webdavmethod.run");
            parseRequest();
            executeRequest();
        } catch (WebdavException ex) {
            // Assume that the response code is already set.
            logCat.warn("webdavex:", ex);
        } catch (Exception ex) {
            logCat.warn("Exception in webdavmethod", ex);
            resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
        }
    }


    /**
     * Read request contents.
     */
    protected void readRequestContent() {

        if (req.getContentLength() == 0)
            return;

        try {
            requestBody = new String(readFromStream(req.getInputStream()),
getEncodingString(req.getCharacterEncoding()));
        }
        catch (Exception e) {
            logCat.warn("Exception when reading req.content: ",e);
        }
        logCat.debug("request:\n"+requestBody);
    }



    /**
     * Translate the encoding string into a proper Java encoding String.
     */
    public static String getEncodingString(String httpEncoding) {
        String result = httpEncoding;
        if (result == null) result = System.getProperty("file.encoding");
        if (result.startsWith("\"")) result = result.substring(1, result.length());
```

```
    if (result.endsWith("\""))   result = result.substring(0, result.length()-1);
    return result;
}

/**
 * Parse WebDAV XML query.
 *
 * @exception WebdavException
 */
protected abstract void parseRequest()
    throws WebdavException, IOException;


/**
 * Parse WebDAV XML request body, and return the DOM representation.
 *
 * @exception ParserConfigurationException If the JAXP factory cannot
 *             configure a parser given its current configuration
 *             parameters.
 * @exception SAXException If any parse errors occur.
 * @exception IOException If any IO errors occur.
 */
protected Document parseRequestContent()
    throws ParserConfigurationException, SAXException, IOException {

    if (requestBody == null) {
        readRequestContent();
        if (requestBody == null) {
            // there is no request content body, so there's nothing to
            // parse
            return null;
        }
    }

    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    factory.setNamespaceAware(true);

    DocumentBuilder builder = factory.newDocumentBuilder();
    return builder.parse(new InputSource(new StringReader(requestBody)));

}


/**
 * Generate XML response.
 *
 * @exception WebdavException
 */
protected abstract void executeRequest()
    throws WebdavException, IOException;

/**
 * Read the data from the stream and return the result in a byte array.
 * Return null in case of an error.
 */
public static  byte[] readFromStream(InputStream inputStream) throws IOException {
    int CHUNK = 1024*4;

    byte[] chunk;
    byte[] all;
    int chunkLen;
    int allLen;
    List chunks;
    int i;
    int max;
    int ofs;

    allLen = 0;
    chunk = new byte[CHUNK];
    chunkLen = inputStream.read(chunk);
    chunks = new ArrayList();
    while (chunkLen != -1) {
        chunks.add(new Integer(chunkLen));
        chunks.add(chunk);
        allLen += chunkLen;
        chunk = new byte[CHUNK];
        chunkLen = inputStream.read(chunk);
```

```
        }

        all = new byte[allLen];
        ofs = 0;
        max = chunks.size();
        for (i = 0; i < max; i += 2) {
            chunkLen = ((Integer) chunks.get(i)).intValue();
            chunk = (byte[]) chunks.get(i + 1);
            System.arraycopy(chunk, 0, all, ofs, chunkLen);
            ofs += chunkLen;
        }
        return all;
    }

}
```

### AbstractMultistatusResponseMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.webdav.MultiStatusException;
import no.inspera.applications.webdav.datamodel.ResourceProperty;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.services.log.LogServices;
import no.inspera.db.MarketplaceDataAccess;
import no.inspera.utils.CommonTools;
import org.apache.log4j.Category;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.util.Iterator;

/**
 * Abstract class for methods which return a multistatus error report, like
 * MOVE, DELETE, COPY, LOCK.
 * Based on corresponding class in Jakarta Slide
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: AbstractMultistatusResponseMethod.java,v 1.1 2003/04/30 14:59:39
johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 * @author Juergen Pill
 */

public abstract class AbstractMultistatusResponseMethod extends WebdavMethod {

    private static Category logCat =
LogServices.getCategory(AbstractMultistatusResponseMethod.class);

    /**
     * Id of the resosurce or collection which is to be copied.
     */
    protected String sourceUri;


    /**
     * Uri of the target.
     */
    protected String destinationUri;


    /**
     * Overwrite.
     */
    protected boolean overwrite;


    /**
     * Constructor.
     *
     * @param req HTTP request
     * @param resp HTTP response
     */
    public AbstractMultistatusResponseMethod
        (HttpServletRequest req, HttpServletResponse resp) {
```

```
        super(req, resp);
    }


    /**
     * Parse request.
     *
     * @exception WebdavException Does not happen
     */
    protected void parseRequest() throws WebdavException {

        sourceUri = requestUri;
        if (sourceUri == null) {
            sourceUri = "/";
        }

        logCat.debug("sourceUri: "+sourceUri);

        destinationUri = req.getHeader("Destination");
        destinationUri = CommonTools.unescapeUTF8(destinationUri);

        int protocolIndex = destinationUri.indexOf("://");
        if (protocolIndex >= 0) {
            // if the Destination URL contains the protocol, we can safely
            // trim everything upto the first "/" character after "://"
            int firstSeparator =
                destinationUri.indexOf("/", protocolIndex + 4);
            if (firstSeparator < 0) {
                destinationUri = "/";
            } else {
                destinationUri = destinationUri.substring(firstSeparator);
            }
        } else {
            String hostName = req.getServerName();
            if ((hostName != null) && (destinationUri.startsWith(hostName))) {
                destinationUri = destinationUri.substring(hostName.length());
            }

            int portIndex = destinationUri.indexOf(":");
            if (portIndex >= 0) {
                destinationUri = destinationUri.substring(portIndex);
            }

            if (destinationUri.startsWith(":")) {
                int firstSeparator = destinationUri.indexOf("/");
                if (firstSeparator < 0) {
                    destinationUri = "/";
                } else {
                    destinationUri = destinationUri.substring(firstSeparator);
                }
            }
        }

        logCat.debug("destinationUri: "+destinationUri);

        String contextPath = req.getContextPath();
        if ((contextPath != null) &&
                (destinationUri.startsWith(contextPath))) {
            destinationUri = destinationUri.substring(contextPath.length());
            logCat.debug("destUri after removing context: "+destinationUri);
        }

        String pathInfo = req.getPathInfo();
        if (pathInfo != null) {
            String servletPath = req.getServletPath();
            if ((servletPath != null) &&
                    (destinationUri.startsWith(servletPath))) {
                destinationUri = destinationUri.substring(servletPath.length());
                logCat.debug("destUri after removing pathinfo: "+destinationUri);
            }
        }
        String mpname =
MarketplaceDataAccess.getInstance().getMarketplace(params.getMarketplaceId()).getMarke
tplaceName();
        if (destinationUri.startsWith("/"+mpname)) {
            destinationUri = destinationUri.substring(mpname.length()+1);
            logCat.debug("destUri after removing mpname: "+destinationUri);
```

```
        }

        String overwriteHeader = req.getHeader("Overwrite");

        if (overwriteHeader != null) {
            if (overwriteHeader.equalsIgnoreCase("T")) {
                overwrite = true;
            } else {
                overwrite = false;
            }
        } else {
            overwrite = true;
        }

    }


    /**
     * Generate an XML error message.
     *
     * @param msex Nested exception
     * @return String XML message
     */
    protected String generateErrorMessage(MultiStatusException msex) {

        StringBuffer errorMessage = new StringBuffer();

        errorMessage.append("<d:multistatus
d=\""+ResourceProperty.DEFAULT_NAMESPACE+">");

        Iterator msexIt = msex.getExceptions();
        while (msexIt.hasNext()) {

            errorMessage.append("<d:response>");
            MultiStatusException.Ex ex =(MultiStatusException.Ex) msexIt.next();
            errorMessage.append("<d:href>"+ex.uri+"</d:href>");
            errorMessage.append("<d:status>"+ex.statusCode+"
"+WebdavStatus.getStatusText(ex.statusCode)+"</d:status>");
            errorMessage.append("</d:response>");

        }

        errorMessage.append("</d:multistatus>");

        return errorMessage.toString();
    }
}
```

## CopyMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.applications.webdav.MultiStatusException;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.db.WebdavDataAccess;
import no.inspera.db.ContentAdminDataAccess;
import no.inspera.db.ContentPresentationDataAccess;
import no.inspera.services.log.LogServices;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

import org.apache.log4j.Category;

/**
 * COPY Method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 * @author Juergen Pill
 */
public class CopyMethod extends AbstractMultistatusResponseMethod {
```

```
    private static Category logCat = LogServices.getCategory(CopyMethod.class);

    // ----------------------------------------------------------- Constructors


    /**
     * COPY method constructor.
     *
     * @param req HTTP request
     * @param resp HTTP response
     */
    public CopyMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }


    // ----------------------------------------------------- Protected Methods



    /**
     * Execute request.
     *
     * @exception WebdavException Unrecoverable error occured while copying
     */
    protected void executeRequest() throws WebdavException {

        boolean isCollection = false;
        try {
            MultiStatusException msex = new MultiStatusException();
            String parentUri =
destinationUri.substring(0,destinationUri.lastIndexOf("/"));
            WebdavDataAccess wdda = WebdavDataAccess.getInstance();
            logCat.debug("getting destination folder, "+parentUri);
            WebdavResource destFolder = wdda.getResource(parentUri,
params.getMarketplaceId());
            if (destFolder == null){
                // returning 404. correct?
                logCat.debug("destination folder not found.");
                msex.addException(WebdavStatus.SC_NOT_FOUND, destinationUri);
                throw msex;
            }
            logCat.debug("got destination folder.");
            String destName =
destinationUri.substring(destinationUri.lastIndexOf("/")+1);
            logCat.debug("getting destination resource, "+destName);
            WebdavResource destResource =
wdda.getChild(destFolder.getContentItem().getObjectId(), destName,
destFolder.getUri());
            ContentAdminDataAccess cada = ContentAdminDataAccess.getInstance();
            if (destResource != null) {
                logCat.debug("destination resource found, checking overwrite");
                if (overwrite) {
                    // if overwrite we just delete the old resource first
(simplification)
                    try {
                        logCat.debug("overwrite; deleting destination resource.");

cada.deleteContentItem(destResource.getContentItem().getObjectId(),
params.getMarketplaceId());
                    } catch (Exception e) {
                        logCat.warn("couldn't delete ci
"+destResource.getContentItem().getObjectId(), e);
                        msex.addException(WebdavStatus.SC_INTERNAL_SERVER_ERROR,
destResource.getUri());
                        throw msex;
                    }
                }else{
                    logCat.debug("no overwrite, returning forbidden.");
                    // returning forbidden. correct?
                    msex.addException(WebdavStatus.SC_FORBIDDEN, destinationUri);
                    throw msex;
                }
            }
            logCat.debug("getting source resource, "+sourceUri);
            WebdavResource sourceResource = wdda.getResource(sourceUri,
params.getMarketplaceId());
```

```
            if (sourceResource == null) {
                logCat.debug("source resource not found, returning 404");
                msex.addException(WebdavStatus.SC_NOT_FOUND, destinationUri);
                throw msex;
            }
            isCollection = sourceResource.isCollection();
            try {
                logCat.debug("copying...");
                cada.copyContentItem(sourceResource.getContentItem().getObjectId(),
destFolder.getContentItem().getObjectId(), params.getMarketplaceId(),
params.getLocale());
                ContentRevision rev =
ContentPresentationDataAccess.getInstance().getLiveContentRevisionWithEffectiveLanguag
e(sourceResource.getContentItem().getObjectId(), params.getLanguageId(),
params.getMarketplaceId());
                rev.setTitle(destName);
                rev.getContentItem().setName(destName);
                logCat.debug("renaming...");
                cada.newContentRevision(rev, params.getUserId(),
params.getLanguageId(),params.getMarketplaceId(), params.getLocale());
            } catch (Exception e) {
                logCat.warn("couldn't copy ci
"+sourceResource.getContentItem().getObjectId()+" to folder
"+destFolder.getContentItem().getObjectId(), e);
                msex.addException(WebdavStatus.SC_INTERNAL_SERVER_ERROR, sourceUri);
            }
            logCat.debug("copy complete.");
            if (overwrite) {
                resp.setStatus(WebdavStatus.SC_NO_CONTENT);
            } else {
                resp.setStatus(WebdavStatus.SC_CREATED);
            }
        } catch (MultiStatusException msex) {
            // If it's not a collection, we don't want to give a 207,
            // because it's silly, and it confuses many clients (such as
            // MS Web Folders).
            if (isCollection) {
                String errorMessage = generateErrorMessage(msex);
                // Write it on the servlet writer
                resp.setStatus(WebdavStatus.SC_MULTI_STATUS);
                try {
                    resp.getWriter().write(errorMessage);
                } catch(IOException ex) {
                    // Critical error ... Servlet container is dead or something
                    ex.printStackTrace();
                    resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
                }
            } else {
                // Returning 207 on non-collection requests is generally
                // considered bad. So let's not do it, since this way
                // makes clients generally behave better.

resp.setStatus(((MultiStatusException.Ex)msex.getExceptions().next()).statusCode);
            }
        }
    }
}
```

## DeleteMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.webdav.MultiStatusException;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.db.ContentAdminDataAccess;
import no.inspera.db.WebdavDataAccess;
import no.inspera.services.log.LogServices;
import org.apache.log4j.Category;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * DELETE method. Based on corresponding class in Jakarta Slide
```

```
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: DeleteMethod.java,v 1.1 2003/04/30 14:59:39 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 */
public class DeleteMethod extends AbstractMultistatusResponseMethod {

    private static Category logCat = LogServices.getCategory(DeleteMethod.class);


    /**
     * Id of the resosurce or collection which is to be deleted.
     */
    private String toDelete;

    /**
     * DELETE method constructor.
     *
     * @param req HTTP request
     * @param resp HTTP response
     */
    public DeleteMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }

    /**
     * Parse request.
     *
     * @exception WebdavException Does not happen
     */
    protected void parseRequest()
        throws WebdavException {
        toDelete = requestUri;
        if (toDelete == null) {
            toDelete = "/";
        }
    }


    /**
     * Execute request.
     *
     * @exception WebdavException Unrecoverable error while deleting
     */
    protected void executeRequest() throws WebdavException {

        boolean isCollection = false;
        try {
            WebdavResource deleteMe =
WebdavDataAccess.getInstance().getResource(toDelete, params.getMarketplaceId());
            isCollection = deleteMe.isCollection();
            // todo: pl/sql deletes recursively. we should really do this on the java
side, so that we can
            // create appropriate multistatusexception.
            try {

ContentAdminDataAccess.getInstance().deleteContentItem(deleteMe.getContentItem().getOb
jectId(), params.getMarketplaceId());
            } catch (Exception e) {
                logCat.warn("exception when deleting ci
"+deleteMe.getContentItem().getObjectId(), e);
                throw new MultiStatusException(); //using empty msex for now.
            }
            resp.setStatus(WebdavStatus.SC_NO_CONTENT);
        } catch (MultiStatusException msex) {
            // If it's not a collection, we don't want to give a 207,
            // because it's silly, and it confuses many clients (such as
            // MS Web Folders).
            if (isCollection) {
                String errorMessage = generateErrorMessage(msex);
                // Write it on the servlet writer
                resp.setStatus(WebdavStatus.SC_MULTI_STATUS);
                try {
                    resp.getWriter().write(errorMessage);
                } catch(IOException ex) {
                    // Critical error ... Servlet container is dead or something
                    ex.printStackTrace();
                    resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
```

```
                }
            } else {
                // Returning 207 on non-collection requests is generally
                // considered bad. So let's not do it, since this way
                // makes clients generally behave better.

resp.setStatus(((MultiStatusException.Ex)msex.getExceptions().next()).statusCode);
            }
        }

    }

}
```

## GetMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.publish.datamodel.ContentItem;
import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.applications.publish.datamodel.ContentSymlink;
import no.inspera.applications.publish.util.PublishConstants;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.db.ContentPresentationDataAccess;
import no.inspera.db.WebdavDataAccess;
import no.inspera.services.importer.Exporter;
import no.inspera.services.log.LogServices;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

/**
 * GET method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: GetMethod.java,v 1.3 2003/05/20 14:26:43 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 */
public class GetMethod extends WebdavMethod {

    private static org.apache.log4j.Category logCat =
LogServices.getCategory(PropFindMethod.class);

    protected final int BUFFER_SIZE = 2048;


    /**
     * The set of SimpleDateFormat formats to use in getDateHeader().
     */
    protected static final SimpleDateFormat formats[] = {
        new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss zzz", Locale.US),
            new SimpleDateFormat("EEEEEE, dd-MMM-yy HH:mm:ss zzz", Locale.US),
            new SimpleDateFormat("EEE MMMM d HH:mm:ss yyyy", Locale.US)
    };


    /**
     * Print content.
     */
    protected boolean printContent = true;


    /**
     * Resource to be retrieved.
     */
    private String resourcePath;


    /**
     * GET Method constructor.
     *
     * @param req HTTP request
```

```
     * @param resp HTTP response
     */
    public GetMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }


    /**
     * Parse XML request.
     */
    protected void parseRequest() throws WebdavException {
        resourcePath = requestUri;
        if (resourcePath == null) {
            resourcePath = "/";
        }
    }


    /**
     * Execute request.
     *
     * @exception WebdavException Can't access resource
     */
    protected void executeRequest() throws WebdavException {

        try {
            logCat.debug("looking for resource at "+resourcePath);
            // Then we must get object contents ...
            WebdavDataAccess wdda = WebdavDataAccess.getInstance();
            WebdavResource object = wdda.getResource(resourcePath,
params.getMarketplaceId());

            // if not found, try without extension
            if (object == null)
                object =
wdda.getResource(resourcePath.substring(0,resourcePath.lastIndexOf(".")),
params.getMarketplaceId());

            if (object != null) {

                ContentPresentationDataAccess cpda =
ContentPresentationDataAccess.getInstance();
                ContentItem ci = object.getContentItem();
                if (ci != null &&
ci.getContentType().equals(PublishConstants.OBJECTTYPE_CONTENTSYMLINK))
                    ci = cpda.getContentItem(((ContentSymlink)ci).getTargetId());

                if (ci!=null) {
                    try {
                        resp.setStatus(WebdavStatus.SC_OK);
                        ContentRevision cr =
cpda.getLiveContentRevisionWithEffectiveLanguage(ci.getObjectId(),
params.getLanguageId(), params.getMarketplaceId());
                        if
(ci.getContentType().equals(ContentPresentationDataAccess.OBJECTTYPE_CONTENTSOUND)){
                            // TODO
                        } else if
(ci.getContentType().equals(ContentPresentationDataAccess.OBJECTTYPE_CONTENTIMAGE)){
                            byte[]b=cr.getBinaryContent();
                            String mimeType= cr.getMimeType();
                            OutputStream out = resp.getOutputStream();
                            resp.setContentType(mimeType);
                            //try to help client-side cache by setting this header...
                            resp.setDateHeader("Expires",new Date().getTime() + 24 *
3600 * 1000);
                            logCat.debug("mimeType "+mimeType);
                            if (b != null && printContent)  {
                                out.write(b);
                                logCat.debug("writing out bin.content ");
                                out.flush();
                                out.close();
                            } else resp.setStatus(WebdavStatus.SC_NO_CONTENT);
                        } else if
(ci.getContentType().equals(ContentPresentationDataAccess.OBJECTTYPE_CONTENTURL)){
                            byte[]b=cr.getBinaryContent();
                            String mimeType= cr.getMimeType();
                            if (mimeType != null && !mimeType.equals(""))
```

```
                                resp.setContentType(mimeType);
                            //try to help client-side cache by setting this header...
                            Date d = new Date();
                            resp.setDateHeader("Expires",d.getTime() + 24 * 3600 *
1000);
                            logCat.debug("mimeType "+mimeType);
                            if (b != null && printContent)  {
                                OutputStream out = resp.getOutputStream();
                                out.write(b);
                                logCat.debug("writing out bin.content ");
                                out.flush();
                                out.close();
                            }else resp.setStatus(WebdavStatus.SC_NO_CONTENT);
                        } else if
(ci.getContentType().equals(ContentPresentationDataAccess.OBJECTTYPE_CONTENTDOCUMENT))
{
                            if (printContent){
                                Exporter e = Exporter.getExporter(params.getUserId(),
params.getMarketplaceId(), params.getLanguageId());
                                OutputStream out = resp.getOutputStream();
                                out.write(e.export(cr));
                                out.flush();
                                out.close();
                            }
                        } else {
                            // TODO
                        }
                    }catch (Exception e) {
                        logCat.error("error redirecting to the correct item", e);
                        resp.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
                    }
                } else {
                    logCat.debug("resource w/o ci, returning 404");
                    resp.setStatus(HttpServletResponse.SC_NOT_FOUND);
                }
            } else {
                logCat.debug("resource not found, returning 404");
                resp.setStatus(HttpServletResponse.SC_NOT_FOUND);
            }

        } catch (Exception e) {
            logCat.debug("exception, returning 500");
            resp.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        }

    }
}
```

### HeadMethod

```
package no.inspera.applications.webdav.method;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * HEAD method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: HeadMethod.java,v 1.1 2003/04/10 11:10:01 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 */
public class HeadMethod extends GetMethod {

    /**
     * HEAD Method constructor.
     *
     * @param req HTTP request
     * @param resp HTTP response
     */
    public HeadMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
        printContent = false;
    }
}
```

### MkColMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.applications.publish.datamodel.ContentFolder;
import no.inspera.applications.objecttype.ObjectType;
import no.inspera.services.log.LogServices;
import no.inspera.services.exception.BaseException;
import no.inspera.db.WebdavDataAccess;
import no.inspera.db.ContentPresentationDataAccess;
import no.inspera.db.ContentFolderDataAccess;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;

import org.apache.log4j.Category;

/**
 * MKCOL Method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: MkcolMethod.java,v 1.1 2003/04/10 10:59:30 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 */
public class MkcolMethod extends WebdavMethod {

    private static Category logCat = LogServices.getCategory(MkcolMethod.class);

    public MkcolMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }


    /**
     * Parse XML request.
     */
    protected void parseRequest() throws WebdavException {
    }


    /**
     * Execute request.
     *
     * @exception WebdavException Bad request
     */
    protected void executeRequest() throws WebdavException {

        logCat.debug("will create folder at "+requestUri);
        String parentUri = requestUri.substring(0,requestUri.lastIndexOf("/"));
        logCat.debug("uri of parent: "+parentUri);
        WebdavResource parent =
WebdavDataAccess.getInstance().getResource(parentUri,params.getMarketplaceId());
        if (parent == null) {
            logCat.debug("parentfolder not found, returning!");
            // todo: return some statuscode
            return;
        }
        logCat.debug("cid of parent: "+parent.getContentItem().getObjectId());
        String name = requestUri.substring(requestUri.lastIndexOf("/")+1);
        logCat.debug("name of folder to create: "+name);

        ContentFolder f = new ContentFolder();
        f.getContentItem().setParentFolder(parent.getContentItem());
        f.getContentItem().setName(name);
        f.setTitle(name);
        f.setCreationUser(params.getUserId());
        f.setCreationIP(req.getRemoteAddr());
        f.setMarketplaceId(params.getMarketplaceId());
        f.setLanguageId(params.getLanguageId());

        ContentFolderDataAccess cfda = ContentFolderDataAccess.getInstance();
        try {
            cfda.newContentFolder(f, params.getLocale());
```

```
            ArrayList regTypes =
cfda.getRegisteredTypes(parent.getContentItem().getObjectId(),
params.getMarketplaceId(), params.getLocale());
            logCat.debug("found "+regTypes.size()+" registered contenttypes on
parent.");
            long ciid =
ContentPresentationDataAccess.getInstance().getContentItemIdFromContentRevisionId(f.ge
tObjectId());
            for (int i = 0; i < regTypes.size(); i++) {
                cfda.registerContentType(new Long(ciid),
((ObjectType)regTypes.get(i)).getObjectType());
            }
        } catch (Exception e) {
            logCat.warn("Exception trying to create new folder. Returning SC-500", e);
            resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
            return;
        }

        // If everything is ok : 201 - Created / OK
        resp.setStatus(WebdavStatus.SC_CREATED);
        logCat.debug("folder created (as far as i know...)");
    }

}
```

## MoveMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.webdav.MultiStatusException;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.db.ContentAdminDataAccess;
import no.inspera.db.WebdavDataAccess;
import no.inspera.db.ContentPresentationDataAccess;
import no.inspera.services.log.LogServices;
import org.apache.log4j.Category;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * MOVE Method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 * @author Juergen Pill
 */
public class MoveMethod extends AbstractMultistatusResponseMethod {

    private static Category logCat = LogServices.getCategory(MoveMethod.class);

    /**
     * MOVE method constructor.
     *
     * @param req HTTP request
     * @param resp HTTP response
     */
    public MoveMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }

    /**
     * Execute request.
     *
     * @exception WebdavException Unrecoverable error while moving the files
     */
    protected void executeRequest() throws WebdavException {

        boolean isCollection = false;
        try {
            MultiStatusException msex = new MultiStatusException();
            String parentUri =
destinationUri.substring(0,destinationUri.lastIndexOf("/"));
            WebdavDataAccess wdda = WebdavDataAccess.getInstance();
```

```
            logCat.debug("getting destination folder, "+parentUri);
            WebdavResource destFolder = wdda.getResource(parentUri,
params.getMarketplaceId());
            if (destFolder == null){
                logCat.debug("destination folder not found.");
                msex.addException(WebdavStatus.SC_NOT_FOUND, destinationUri);
                throw msex;
            }
            logCat.debug("got destination folder.");
            String destName =
destinationUri.substring(destinationUri.lastIndexOf("/")+1);
            logCat.debug("getting destination resource, "+destName);
            WebdavResource destResource =
wdda.getChild(destFolder.getContentItem().getObjectId(), destName,
destFolder.getUri());
            ContentAdminDataAccess cada = ContentAdminDataAccess.getInstance();
            if (destResource != null) {
                logCat.debug("destination resource found, checking overwrite");
                if (overwrite) {
                    // if overwrite we just delete the old resource first
(simplification)
                    try {
                        logCat.debug("overwrite; deleting destination resource.");

cada.deleteContentItem(destResource.getContentItem().getObjectId(),
params.getMarketplaceId());
                    } catch (Exception e) {
                        logCat.warn("couldn't delete ci
"+destResource.getContentItem().getObjectId(), e);
                        msex.addException(WebdavStatus.SC_INTERNAL_SERVER_ERROR,
destResource.getUri());
                        throw msex;
                    }
                }else{
                    logCat.debug("no overwrite, returning forbidden.");
                    msex.addException(WebdavStatus.SC_FORBIDDEN, destinationUri);
                    throw msex;
                }
            }
            logCat.debug("getting source resource, "+sourceUri);
            WebdavResource sourceResource = wdda.getResource(sourceUri,
params.getMarketplaceId());
            if (sourceResource == null) {
                logCat.debug("source resource not found, returning 404");
                msex.addException(WebdavStatus.SC_NOT_FOUND, destinationUri);
                throw msex;
            }
            isCollection = sourceResource.isCollection();
            try {
                logCat.debug("moving...");
                // remove extension from name:
                if (destName.indexOf(".") > -1) destName =
destName.substring(0,destName.lastIndexOf("."));
                cada.moveContentItem(sourceResource.getContentItem().getObjectId(),
destFolder.getContentItem().getObjectId(), params.getMarketplaceId(),
params.getLocale());
                ContentRevision rev =
ContentPresentationDataAccess.getInstance().getLiveContentRevisionWithEffectiveLanguag
e(sourceResource.getContentItem().getObjectId(), params.getLanguageId(),
params.getMarketplaceId());
                rev.setTitle(destName);
                rev.getContentItem().setName(destName);
                logCat.debug("renaming...");
                cada.newContentRevision(rev, params.getUserId(),
params.getLanguageId(),params.getMarketplaceId(), params.getLocale());
            } catch (Exception e) {
                logCat.warn("couldn't move ci
"+sourceResource.getContentItem().getObjectId()+" to folder
"+destFolder.getContentItem().getObjectId(), e);
                msex.addException(WebdavStatus.SC_INTERNAL_SERVER_ERROR, sourceUri);
            }
            logCat.debug("move complete.");
            if (overwrite) {
                resp.setStatus(WebdavStatus.SC_NO_CONTENT);
            } else {
                resp.setStatus(WebdavStatus.SC_CREATED);
            }
```

```
        } catch (MultiStatusException msex) {
            // If it's not a collection, we don't want to give a 207,
            // because it's silly, and it confuses many clients (such as
            // MS Web Folders).
            if (isCollection) {
                String errorMessage = generateErrorMessage(msex);
                // Write it on the servlet writer
                resp.setStatus(WebdavStatus.SC_MULTI_STATUS);
                try {
                    resp.getWriter().write(errorMessage);
                } catch(IOException ex) {
                    // Critical error ... Servlet container is dead or something
                    ex.printStackTrace();
                    resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
                }
            } else {
                // Returning 207 on non-collection requests is generally
                // considered bad. So let's not do it, since this way
                // makes clients generally behave better.

resp.setStatus(((MultiStatusException.Ex)msex.getExceptions().next()).statusCode);
            }
        }
    }
}
```

## OptionsMethod

```
package no.inspera.applications.webdav.method;

import org.apache.log4j.Category;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import no.inspera.services.log.LogServices;
import no.inspera.applications.webdav.WebdavException;


/**
 * OPTIONS Method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: OptionsMethod.java,v 1.5 2003/05/05 12:58:24 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 */
public class OptionsMethod extends WebdavMethod {

    private static Category logCat = LogServices.getCategory(OptionsMethod.class);

    public OptionsMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }

    protected void parseRequest() throws WebdavException {
    }

    protected void executeRequest() throws WebdavException {

        logCat.debug("executing OPTIONS");

        // todo: dav 2 requires locks+++
        resp.addHeader("DAV", "1");


        String resourceUri = requestUri;
        if (resourceUri == null) {
            resourceUri = "/";
        }

        String methodsAllowed = "OPTIONS, GET, HEAD, POST, PUT, MKCOL, PROPFIND,
DELETE, COPY, MOVE";

        resp.addHeader("Allow", methodsAllowed);

        // used by MS Web Folder
        resp.addHeader("MS-Author-Via", "DAV");
    }
```

```
}
```

### PostMethod

```
package no.inspera.applications.webdav.method;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * POST method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 * @version $Id: PostMethod.java,v 1.1 2003/05/01 12:06:36 johnarne Exp $
 */
public class PostMethod extends PutMethod {

    public PostMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }
}
```

### PropFindMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.webdav.datamodel.ResourceProperty;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.db.WebdavDataAccess;
import no.inspera.services.log.LogServices;
import no.inspera.utils.XMLTools;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.ParserConfigurationException;
import java.io.IOException;
import java.util.*;

/**
 * PROPFIND method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: PropFindMethod.java,v 1.7 2003/05/20 14:26:52 johnarne Exp $
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 */
public class PropFindMethod extends WebdavMethod {

    private static org.apache.log4j.Category logCat =
LogServices.getCategory(PropFindMethod.class);

    /**
     * Root node URI.
     */
    protected static final String DIRECTORY = "/";


    /**
     * Default depth is infite.
     */
    protected static final int INFINITY = Integer.MAX_VALUE;


    /**
     * Specify a property mask.
     */
    protected static final int FIND_BY_PROPERTY = 0;


    /**
     * Display all properties.
```

```
 */
protected static final int FIND_ALL_PROP = 1;


/**
 * Return property names.
 */
protected static final int FIND_PROPERTY_NAMES = 2;

/**
 * Depth.
 */
protected int depth;


/**
 * Type of the PROPFIND method.
 */
protected int propFindType;


/**
 * Properties to display
 */
protected Vector propertyVector;


/**
 * Current namespace number used to generate namespace abbreviations.
 */
protected int namespaceNumber = 0;


/**
 * Namespaces list. Keyed by namespace names.
 */
protected HashMap namespaces;


/**
 * PROPFIND method constructor.
 *
 * @param req HTTP request
 * @param resp HTTP response
 */
public PropFindMethod(HttpServletRequest req, HttpServletResponse resp) {
    super(req, resp);
    readRequestContent();

    namespaces = new HashMap();
    namespaces.put(INSPERA_NAMESPACE, INSPERA_NAMESPACE_ABBREV);
    depth = INFINITY;
    propFindType = FIND_ALL_PROP;
}

/**
 * Parse the request.
 *
 * @exception WebdavException Bad request
 */
protected void parseRequest() throws WebdavException {

    // parse Depth header
    String depthStr = req.getHeader("Depth");
    if (depthStr == null) {
        depth = INFINITY;
    } else if (depthStr.equals("0")) {
        depth = 0;
    } else if (depthStr.equals("1")) {
        depth = 1;
    } else if (depthStr.equalsIgnoreCase("infinity")) {
        depth = INFINITY;
    } else {
        try {
            depth = Integer.parseInt(depthStr);
            if (depth < 0) {
                depth = 0;
```

```
                }
            } catch (NumberFormatException ex) {
                depth = INFINITY;
            }
        }

        // limit tree browsing a bit
        if (depth > DEPTH_LIMIT) {
            depth = DEPTH_LIMIT;
        }

        if (req.getContentLength() != 0) {

            Node propNode = null;

            try {
                Document document = parseRequestContent();

                // Get the root element of the document
                Element rootElement = document.getDocumentElement();
                NodeList childList = rootElement.getChildNodes();

                for (int i=0; i < childList.getLength(); i++) {
                    Node currentNode = childList.item(i);
                    switch (currentNode.getNodeType()) {
                        case Node.TEXT_NODE:
                            break;
                        case Node.ELEMENT_NODE:
                            if (currentNode.getNodeName().endsWith("prop")) {
                                propFindType = FIND_BY_PROPERTY;
                                propNode = currentNode;
                            }
                            if (currentNode.getNodeName().endsWith("propname")) {
                                propFindType = FIND_PROPERTY_NAMES;
                            }
                            if (currentNode.getNodeName().endsWith("allprop")) {
                                propFindType = FIND_ALL_PROP;
                            }
                            break;
                    }
                }
            } catch (SAXException e) {
                resp.setStatus(WebdavStatus.SC_BAD_REQUEST);
                throw new WebdavException("WEBDAV_SC_"+WebdavStatus.SC_BAD_REQUEST,
Locale.getDefault(), PropFindMethod.class);
            } catch (ParserConfigurationException e) {
                System.err.println(e.getMessage());

resp.setStatus(no.inspera.applications.webdav.WebdavStatus.SC_INTERNAL_SERVER_ERROR);
                throw new WebdavException
                    ("WEBDAV_SC_"+WebdavStatus.SC_INTERNAL_SERVER_ERROR,
Locale.getDefault(), PropFindMethod.class);
            } catch (IOException e) {
                System.err.println(e.getMessage());
                e.printStackTrace();
                resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
                throw new WebdavException
                    ("WEBDAV_SC_"+WebdavStatus.SC_INTERNAL_SERVER_ERROR,
Locale.getDefault(), PropFindMethod.class);
            }

            if (propFindType == FIND_BY_PROPERTY) {
                propertyVector = new Vector();
                NodeList childList = propNode.getChildNodes();

                for (int i=0; i < childList.getLength(); i++) {
                    Node currentNode = childList.item(i);
                    switch (currentNode.getNodeType()) {
                        case Node.TEXT_NODE:
                            break;
                        case Node.ELEMENT_NODE:
                            ResourceProperty property = getProperty(currentNode);
                            propertyVector.addElement(property);
                            break;
                    }
                }
```

```
            }

        }

    }


    /**
     * Parse the namespace info of a node name.
     *
     * @param node The DOM node to parse
     * @return The corresponding ResourceProperty object
     */
    protected ResourceProperty getProperty(Node node) {

        ResourceProperty property = new ResourceProperty(node.getLocalName(),
null,node.getNamespaceURI());
        generateNamespaceAbbreviation(node.getNamespaceURI());

        return property;

    }


    /**
     * Execute the request.
     *
     * @exception WebdavException
     */
    protected void executeRequest()
        throws IOException, WebdavException {

        logCat.debug("executing propfind");

        resp.setStatus(WebdavStatus.SC_MULTI_STATUS);

        // Loads the associated object from the store.
        String resourceUri = requestUri;

        resp.setContentType("text/xml; charset=ISO-8859-1");

        StringBuffer xml = new StringBuffer();

        // Create multistatus object
        xml.append("<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>");

        xml.append("<multistatus" + generateNamespaceDeclarations()+ ">");

        WebdavDataAccess wda = WebdavDataAccess.getInstance();
        WebdavResource resource = wda.getResource(resourceUri,
params.getMarketplaceId());
        logCat.debug("resource null? "+(resource==null));
        if (resource != null) {
            logCat.debug("found a resource!");
            if (depth == 0) {
                logCat.debug("depth is 0");
                xml.append(parsePropertiesOfObject(resource));
            } else {
                logCat.debug("depth is not 0");
                // The stack always contains the object of the current level
                Stack stack = new Stack();
                stack.push(resource);

                // Stack of the objects one level below
                Stack stackBelow = new Stack();

                while ((!stack.isEmpty()) && (depth >= 0)) {

                    WebdavResource cur = (WebdavResource) stack.pop();
                    xml.append(parsePropertiesOfObject(cur));

                    if (depth > 0) {

                        Iterator children = null;

                        try {
```

```
                            children =
wda.getChildren(cur.getContentItem().getObjectId(),cur.getUri());
                        } catch (Exception e) {
                            //resp.setStatus(getErrorCode(e));  // no special handling
needed
                            //throw new WebdavException(WebdavStatus.SC_ACCEPTED,
false); // abort the TA
                            logCat.debug("Exception getting children: ", e);
                        }

                        while (children.hasNext()){
                            stackBelow.push(children.next());
                        }
                    }

                    if (stack.isEmpty()) {
                        depth--;
                        stack = stackBelow;
                        stackBelow = new Stack();
                    }

                }
            }
        }

        xml.append("</multistatus>");

        logCat.debug("xml:" + new String(xml));

        resp.getOutputStream().println(new String(xml));
        resp.getOutputStream().flush();

    }


    /**
     * Parse properties of an object.
     *
     * @param object
     * @exception WebdavException
     */
    protected StringBuffer parsePropertiesOfObject(WebdavResource object) throws
WebdavException {
        logCat.debug("parsing properties.");

        StringBuffer xml = new StringBuffer();
        xml.append("<response>");

        String status = new String("HTTP/1.1 " + WebdavStatus.SC_OK + " " +
WebdavStatus.getStatusText(WebdavStatus.SC_OK));

        try {
            xml.append("<href>");
            try {
                String path = object.getUri();
                logCat.debug("appending path: "+path);
                xml.append(path);

            } catch (Exception e){// RevisionDescriptorNotFoundException e) {
                logCat.debug("exception getting resource url: ",e);
            }

            xml.append("</href>");
        } catch (Exception e) {
            logCat.debug("Exception writing href tag:", e);
        }

        switch (propFindType) {
            case FIND_ALL_PROP :
                logCat.debug("find all properties");
                // Show properties / values for current object.
                xml.append("<propstat>");
                xml.append("<prop>");

                status = new String("HTTP/1.1 " + WebdavStatus.SC_OK
                                    + " " + WebdavStatus.getStatusText
                                    (WebdavStatus.SC_OK));
```

```
                Iterator properties =
object.getProperties(null,params.getLanguageId()).values().iterator();

                while (properties.hasNext()){
                    ResourceProperty currentProperty = (ResourceProperty)
properties.next();
                    if (currentProperty != null) {
                        String namespace = currentProperty.getNamespace();
                        generateNamespaceAbbreviation(namespace);
                        String namespaceAbbrev = (String) namespaces.get(namespace);
                        String currentPropertyValue = currentProperty.getValue();
                        if ((currentPropertyValue == null) ||
(currentPropertyValue.toString().equals(""))) {
                            xml.append(XMLTools.tag(currentProperty.getName(),
namespaceAbbrev,  null));
                        } else {
                            xml.append(XMLTools.tag(currentProperty.getName(),
namespaceAbbrev, currentPropertyValue));
                        }

                    }
                }
                xml.append("</prop>");
                xml.append("<status>"+status+"</status>");
                xml.append("</propstat>");
                break;
            case FIND_PROPERTY_NAMES :
                logCat.debug("find property names");
                // TODO: not used by Web Folder
                break;
            case FIND_BY_PROPERTY :
                logCat.debug("find specific property");
                // Show requested properties value.

                Enumeration propertyList = propertyVector.elements();
                HashMap foundProperties = object.getProperties(new
HashSet(propertyVector), params.getLanguageId());

                xml.append("<propstat>");
                xml.append("<prop>");
                status = new String("HTTP/1.1 " + WebdavStatus.SC_OK
                                    + " " + WebdavStatus.getStatusText
                                    (WebdavStatus.SC_OK));

                Vector propertiesNotFoundVector = new Vector();

                while (propertyList.hasMoreElements()) {
                    ResourceProperty property = (ResourceProperty)
propertyList.nextElement();
                    String propertyName = property.getName();
                    String propertyNamespace = property.getNamespace();

                    // Search in the custom properties
                    logCat.debug("looking for property "+propertyName+", namespace
"+propertyNamespace); //todo: namespace
                    ResourceProperty currentProperty =
(ResourceProperty)foundProperties.get(property);
                    if (currentProperty != null) {

                        String namespace = currentProperty.getNamespace();
                        String namespaceAbbrev =(String) namespaces.get(namespace);
                        String currentPropertyValue = currentProperty.getValue();
                        if ((currentPropertyValue == null) ||
(currentPropertyValue.equals(""))) {
                            xml.append(XMLTools.tag(currentProperty.getName(),
namespaceAbbrev, null));
                        } else {
                            xml.append(XMLTools.tag(currentProperty.getName(),
namespaceAbbrev, currentPropertyValue));
                        }
                    } else {
                        propertiesNotFoundVector.addElement(property);
                    }

                }
```

```
            xml.append("</prop>");
            xml.append(XMLTools.tag("status", status));
            xml.append("</propstat>");

            Enumeration propertiesNotFoundList =
                propertiesNotFoundVector.elements();

            if (propertiesNotFoundList.hasMoreElements()) {

                status = new String("HTTP/1.1 " + WebdavStatus.SC_NOT_FOUND
                                    + " " + WebdavStatus.getStatusText
                                    (WebdavStatus.SC_NOT_FOUND));

                xml.append("<propstat>");
                xml.append("<prop>");

                while (propertiesNotFoundList.hasMoreElements()) {
                    ResourceProperty propertyNotFound =
                        (ResourceProperty) propertiesNotFoundList.nextElement();
                    String name = propertyNotFound.getName();
                    String namespace = propertyNotFound.getNamespace();
                    String namespaceAbbrev =
                        (String) namespaces.get(namespace);
                    xml.append(XMLTools.tag(name, namespaceAbbrev,null));
                }
                xml.append("</prop>");
                xml.append(XMLTools.tag("status", status));
                xml.append("</propstat>");
            }
            break;
    }

    xml.append("</response>");

    return xml;
}


/**
 * Generate a namespace abbreviation for the given namespace.
 *
 * @param namespaceName Name of the namespace
 */
protected void generateNamespaceAbbreviation(String namespaceName) {

    if (namespaces.get(namespaceName) != null)
        return;

    if (namespaceName.equals("DAV:"))
        return;

    String namespaceAbbrev = "ns" + namespaceNumber++;
    namespaces.put(namespaceName, namespaceAbbrev);

}


/**
 * Generate namespace declaration attributes.
 *
 * @return String namespace attributes
 */
protected String generateNamespaceDeclarations() {

    StringBuffer result = new StringBuffer();

    result.append(" xmlns=\"").append("DAV:")
        .append("\" ");

    Iterator namespaceList = namespaces.keySet().iterator();
    while (namespaceList.hasNext()) {

        String namespace = (String) namespaceList.next();
        String abbrev = (String) namespaces.get(namespace);
        result.append("xmlns:").append(abbrev).append("=\"")
            .append(namespace).append("\" ");
```

```
        }

        return result.toString();
    }
}
```

### PutMethod

```
package no.inspera.applications.webdav.method;

import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.applications.publish.util.PublishConstants;
import no.inspera.applications.publish.util.ImportFileWriter;
import no.inspera.applications.webdav.WebdavException;
import no.inspera.applications.webdav.datamodel.WebdavResource;
import no.inspera.applications.webdav.WebdavStatus;
import no.inspera.db.ContentAdminDataAccess;
import no.inspera.db.WebdavDataAccess;
import no.inspera.services.importer.Importer;
import no.inspera.services.log.LogServices;
import org.apache.log4j.Category;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * PUT method. Based on corresponding class in Jakarta Slide
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @author <a href="mailto:remm@apache.org">Remy Maucherat</a>
 * @version $Id: PutMethod.java,v 1.3 2003/05/20 14:27:07 johnarne Exp $
 */
public class PutMethod extends WebdavMethod {

    private static Category logCat = LogServices.getCategory(PutMethod.class);

    /**
     * Resource to be written.
     */
    private String resourcePath;



    /**
     * PUT Method constructor.
     *
     * @param req HTTP request
     * @param resp HTTP response
     */
    public PutMethod(HttpServletRequest req, HttpServletResponse resp) {
        super(req, resp);
    }

    /**
     * Parse XML request.
     *
     * @exception WebdavException Does not happen
     */
    protected void parseRequest() throws WebdavException {
        resourcePath = requestUri;
        if (resourcePath == null) {
            resourcePath = "/";
        }
    }

    /**
     * Execute request.
     *
     * @exception WebdavException Bad request
     */
    protected void executeRequest() throws WebdavException {

        if (!(Long.parseLong(req.getHeader("content-length"))>0)){
            logCat.debug("content-length = 0!");
            resp.setStatus(WebdavStatus.SC_NO_CONTENT);
            return;
        }
```

```
        String parentUri = resourcePath.substring(0,resourcePath.lastIndexOf("/"));
        WebdavDataAccess wdda = WebdavDataAccess.getInstance();
        logCat.debug("getting destination folder, "+parentUri);
        WebdavResource destFolder = wdda.getResource(parentUri,
params.getMarketplaceId());
        if (destFolder == null){
            // returning 409. correct?
            logCat.debug("destination folder not found.");
            resp.setStatus(WebdavStatus.SC_CONFLICT);
            return;
        }
        logCat.debug("got destination folder.");
        String destName = resourcePath.substring(resourcePath.lastIndexOf("/")+1);

        byte[] content = null;
        try {
            content = readFromStream(req.getInputStream());
        } catch (IOException e) {
            logCat.warn("could not read content body. ",e);
            resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
            return;
        }
        String extension = destName.substring(destName.lastIndexOf(".")+1);
        String name = destName.substring(0,destName.length()-(extension.length()+1));
        logCat.debug("destination name: "+name+", extension: "+extension);


        String mimeType = req.getContentType();
        if (mimeType == null || mimeType.equals("")){
            //try to deduce mimetype by extension
            mimeType = ImportFileWriter.getMimeType(extension);
        }
        Importer imp = Importer.getImporter(mimeType,params.getMarketplaceId(),
params.getLanguageId());
        ContentRevision rev = imp.doImport(content);
        if (rev == null){
            logCat.info("could not import object");
            resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
            return;
        }
        logCat.debug("languageId in rev is "+rev.getLanguageId());
        if (rev.getTitle() == null || rev.getTitle().equals(""))
            rev.setTitle(name);
        if (rev.getObjectType().equals(PublishConstants.OBJECTTYPE_CONTENTURL))
            rev.setTitle(name+"."+extension);
        //rev.getContentItem().setName(name);
        rev.getContentItem().setParentFolder(destFolder.getContentItem());
        logCat.debug("imported revision, ciid: "+rev.getContentItem().getObjectId()+",
contenttype: "+rev.getContentItem().getContentType());


        logCat.debug("getting destination resource, "+destName);
        WebdavResource destResource = wdda.getResource(resourcePath,
params.getMarketplaceId());
        ContentAdminDataAccess cada = ContentAdminDataAccess.getInstance();

        if (destResource != null) {
            logCat.debug("destination resource found, checking for folder.");
            if
(destResource.getContentItem().getContentType().equals(PublishConstants.OBJECTTYPE_CON
TENTFOLDER)){
                logCat.debug("destination is collection, returning conflict.");
                resp.setStatus(WebdavStatus.SC_CONFLICT);
                return;
            }

            // we do not overwrite (as the protocol specifies) if the existing
resource is same contenttype.
            logCat.debug("contenttype of existing revision:
"+destResource.getContentItem().getContentType());
            if
(rev.getContentItem().getContentType().equals(destResource.getContentItem().getContent
Type())){
                rev.setContentItem(destResource.getContentItem());
            }else{
                // if contenttype differs, we delete existing ci.
```

```
                try {
                    logCat.debug("deleting existing object...");

cada.deleteContentItem(destResource.getContentItem().getObjectId(),
params.getMarketplaceId());
                } catch (Exception e) {
                    logCat.warn("could not delete existing object.", e);
                    resp.setStatus(WebdavStatus.SC_CONFLICT);
                    return;
                }
            }
        }

        try {
            rev.getContentItem().setName(name);
            logCat.debug("storing revision...");
            cada.newContentRevision(rev, params.getUserId(), params.getLanguageId(),
params.getMarketplaceId(), params.getLocale());
        } catch (Exception e) {
            logCat.warn("Could not store new revision of ci
"+rev.getContentItem().getObjectId(), e);
            resp.setStatus(WebdavStatus.SC_INTERNAL_SERVER_ERROR);
            return;
        }
        resp.setStatus(WebdavStatus.SC_CREATED);
    }
}
```

## A-5 no.inspera.services.importer

### POIPropertyReader

```
package no.inspera.services.importer;

import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.kernel.Person;
import no.inspera.services.exception.BaseException;
import no.inspera.services.log.LogServices;
import no.inspera.services.metadataservice.Metadata;
import no.inspera.services.relation.PartyObjectRoleType;
import no.inspera.services.relation.RoleType;
import org.apache.log4j.Category;
import org.apache.poi.hpsf.PropertySetFactory;
import org.apache.poi.hpsf.SummaryInformation;
import org.apache.poi.poifs.eventfilesystem.POIFSReader;
import org.apache.poi.poifs.eventfilesystem.POIFSReaderEvent;
import org.apache.poi.poifs.eventfilesystem.POIFSReaderListener;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.StringTokenizer;

/**
 * Metadata-extractor for MS Office files.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: POIPropertyReader.java,v 1.1 2003/05/20 14:33:55 johnarne Exp $
 */
public class POIPropertyReader {

    private static Category logCat = LogServices.getCategory(POIPropertyReader.class);

    /**
     * Reads properties from the file and stores them on the revision.
     * @param data Office file to read from.
     * @param rev Revision to store metadata on.
     */
    public static void readProperties(byte[] data, ContentRevision rev){
        POIFSReader r = new POIFSReader();
        r.registerListener(new MyPOIFSReaderListener(rev), "\005SummaryInformation");
        try {
            r.read(new ByteArrayInputStream(data));
        } catch (IOException e) {
            logCat.warn("could not read from file.", e);
```

```
        }
    }

    /**
     * Private class handling the read-events.
     */
    private static class MyPOIFSReaderListener implements POIFSReaderListener{

        ContentRevision rev = null;

        /**
         * Default constructor
         * @param rev The revision metadata is to be stored on.
         */
        public MyPOIFSReaderListener(ContentRevision rev){
            this.rev = rev;
        }

        /**
         * Handles metadata extraction and storing.
         * @param event The read-event.
         */
        public void processPOIFSReaderEvent(POIFSReaderEvent event){

            SummaryInformation si = null;
            try {
                si = (SummaryInformation)
PropertySetFactory.create(event.getStream());
            }
            catch (Exception ex){
                logCat.warn("could not get summaryinfo from event "+event.getName(),
ex);
            }

            String title = si.getTitle();
            logCat.debug("title: "+title);
            if (title != null && !title.equals(""))
                rev.setTitle(title);

            String desc = si.getSubject();
            logCat.debug("desc: "+desc);
            if (desc != null && !desc.equals(""))
                rev.setDescription(desc);

            String author = si.getAuthor();
            logCat.debug("author: "+author);
            if (author != null && !author.equals("")){
                String firstName, lastName = null;
                if (author.indexOf(",")>0){
                    lastName = author.substring(0,author.indexOf(","));
                    firstName = author.substring(author.indexOf(","));
                }else if (author.indexOf(" ")>0){
                    lastName = author.substring(0,author.lastIndexOf(" "));
                    firstName = author.substring(author.lastIndexOf(" "));
                }else{
                    lastName = author;
                    firstName = "";
                }
                logCat.debug("firstname: "+firstName+", lastname: "+lastName);
                try {
                    Person p = new Person();
                    p.setFirstName(firstName);
                    p.setLastName(lastName);
                    p.addRoletype(RoleType.getRoleType("author_rel"));
                    PartyObjectRoleType port = new PartyObjectRoleType();
                    port.setRoleType("author_rel");
                    port.setParty(p);
                    ArrayList contributors = new ArrayList();
                    contributors.add(port);
                    rev.addClassification("contributors", contributors);
                } catch (BaseException e) {
                    logCat.warn("exception adding author", e);
                }
            }

            String keywords = si.getKeywords();
            logCat.debug("keywords: "+keywords);
```

```
            if (keywords != null && !keywords.equals("")){
                ArrayList keys = new ArrayList();
                StringTokenizer st = new StringTokenizer(keywords, ";,", false);
                while (st.hasMoreElements()){
                    String keyword = (String)st.nextElement();
                    logCat.debug("keyword found: "+keyword);
                    keys.add(keyword);
                }
                if (keys.size()>0)
                    rev.addClassification("keyword", keys);
            }

            String comments = si.getComments();
            logCat.debug("comments: "+comments);
            if (comments != null && !comments.equals("")){
                rev.setComments(comments);
            }

            Date creationDate = si.getCreateDateTime();
            logCat.debug("created: "+creationDate);
            if (creationDate != null){
                Calendar c = Calendar.getInstance();
                c.setTime(creationDate);
                rev.setCreationDate(c);
            }

            Date lastModified = si.getLastSaveDateTime();
            logCat.debug("lastModified: "+creationDate);
            if (lastModified != null){
                Calendar c = Calendar.getInstance();
                c.setTime(lastModified);
                rev.setLastModified(c);
            }

            String application = si.getApplicationName();
            logCat.debug("application: "+application);
            if (application != null && !application.equals("")){
                Metadata m = rev.getMetadata();
                m.addMetadata("sourceApplicationName", application);
            }

            String revnum = si.getRevNumber();
            logCat.debug("revnum: "+revnum);
            if (revnum != null && !revnum.equals("")){
                Metadata m = rev.getMetadata();
                m.addMetadata("sourceRevisionNumber", revnum);
            }
        }
    }
}
```

### XMPPropertyReader

```
package no.inspera.services.importer;

import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.kernel.Person;
import no.inspera.services.exception.BaseException;
import no.inspera.services.log.LogServices;
import no.inspera.services.metadataservice.Metadata;
import no.inspera.services.relation.PartyObjectRoleType;
import no.inspera.services.relation.RoleType;
import no.inspera.utils.XMLTools;
import org.apache.log4j.Category;
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.Reader;
import java.io.StringReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
```

```
/**
 * Metadata extractor for files with embedded XMP packet.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: XMPPropertyReader.java,v 1.1 2003/05/20 14:33:55 johnarne Exp $
 */
public class XMPPropertyReader {

    private static Category logCat = LogServices.getCategory(POIPropertyReader.class);

    /**
     * Searches the supplied file for XMP packet and stores metadata on the supplied
revision.
     * @param data File to read from.
     * @param rev The revision to store metadata on.
     */
    public static void readProperties(byte[] data, ContentRevision rev){
        XmpParser parser = new XmpParser();
        StringBuffer strBuff = new StringBuffer();
        Reader meta = null;
        logCat.debug("looking for xmp-packet...");
        try {
             meta = parser.getXpacket(new ByteArrayInputStream(data));
        } catch (IOException e) {
            logCat.warn("exception parsing for xmp.", e);
        }
        int offset = parser.getOffset();
        if (offset > 0) {
            logCat.debug("xmp-packet found!");
            try {
                char[] buf = new char[1024];
                int count = 0;
                while ((count = meta.read(buf)) >= 0) {
                    strBuff.append(buf,0,count);
                }
                logCat.debug("metadata: "+new String(strBuff));
            } catch (IOException e) {
                e.printStackTrace();
            }

            Document doc = null;
            try {
                doc = XMLTools.getDocument(new InputSource(new StringReader(new
String(strBuff))), false);
            } catch (Exception e) {
                logCat.warn("could not parse xmp.", e);
            }
            if (doc != null){
                NodeList nodes =
doc.getElementsByTagNameNS("http://ns.adobe.com/xap/1.0/", "*");
                logCat.debug(nodes.getLength()+" xap-nodes found.");
                String nodeName = null;
                SimpleDateFormat dateFormatter = new SimpleDateFormat();
                for (int i = 0; i < nodes.getLength(); i++) {
                    nodeName = nodes.item(i).getLocalName();
                    logCat.debug("processing "+nodeName);

                    if (nodeName.equals("Title")){
                        // Title/Alt/li/text
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        String title =
nodes.item(i).getFirstChild().getFirstChild().getFirstChild().getNodeValue();
                        logCat.debug("title: "+title);
                        if (title != null && !title.equals(""))
                            rev.setTitle(title);
                    }else if (nodeName.equals("Description")){
                        // Description/Alt/li/text
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        String desc =
nodes.item(i).getFirstChild().getFirstChild().getFirstChild().getNodeValue();
                        logCat.debug("desc: "+desc);
                        if (desc != null && !desc.equals(""))
                            rev.setDescription(desc);
                    }else if (nodeName.equals("Author")){
                        // Author/text
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        String author = nodes.item(i).getFirstChild().getNodeValue();
```

```
                        logCat.debug("author: "+author);
                        if (author != null && !author.equals("")){
                            String firstName, lastName = null;
                            if (author.indexOf(",")>0){
                                lastName = author.substring(0,author.indexOf(","));
                                firstName = author.substring(author.indexOf(","));
                            }else if (author.indexOf(" ")>0){
                                lastName = author.substring(0,author.lastIndexOf("
"));
                                firstName = author.substring(author.lastIndexOf(" "));
                            }else{
                                lastName = author;
                                firstName = "";
                            }
                            logCat.debug("firstname: "+firstName+", lastname:
"+lastName);
                            try {
                    Person p = new Person();
                    p.setFirstName(firstName);
                    p.setLastName(lastName);
                    p.addRoletype(RoleType.getRoleType("author_rel"));
                    PartyObjectRoleType port = new PartyObjectRoleType();
                    port.setRoleType("author_rel");
                    port.setParty(p);
                    ArrayList contributors = new ArrayList();
                    contributors.add(port);
                    rev.addClassification("contributors", contributors);
                } catch (BaseException e) {
                    logCat.warn("exception adding author", e);
                }
                        }
                    }else if (nodeName.equals("Keywords")){
                        // Keywords/Bag/li/text
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        NodeList bagItems =
nodes.item(i).getFirstChild().getChildNodes();
                        ArrayList keys = new ArrayList();
                        for (int j = 0; j < bagItems.getLength(); j++) {
                            String keyword =
bagItems.item(j).getFirstChild().getNodeValue();
                            logCat.debug("keyword found: "+keyword);
                            keys.add(keyword);
                        }
                        if (keys.size()>0)
                            rev.addClassification("keyword", keys);
                    }else if (nodeName.equals("CreateDate")){
                        // CreateDate/text
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        String creationDate =
nodes.item(i).getFirstChild().getNodeValue();
                        logCat.debug("created: "+creationDate);
                        if (creationDate != null){
                            Calendar c = Calendar.getInstance();
                            try {
                                c.setTime(dateFormatter.parse(creationDate));
                                rev.setCreationDate(c);
                            } catch (ParseException e) {
                                logCat.warn("could not parse creation date", e);
                            }
                        }
                    }else if (nodeName.equals("ModifyDate")){
                        // ModifyDate/text
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        String lastModified =
nodes.item(i).getFirstChild().getNodeValue();
                        logCat.debug("lastModified: "+lastModified);
                        if (lastModified != null){
                            Calendar c = Calendar.getInstance();
                            try {
                                c.setTime(dateFormatter.parse(lastModified));
                                rev.setCreationDate(c);
                            } catch (ParseException e) {
                                logCat.warn("could not parse last modified", e);
                            }
                        }
                    }else if (nodeName.equals("CreatorTool")){
                        // CreatorTool/text
```

```
                        XMLTools.removeEmptyTextNodesFromNode(nodes.item(i));
                        String application =
nodes.item(i).getFirstChild().getNodeValue();
                        logCat.debug("application: "+application);
                        if (application != null && !application.equals("")){
                            Metadata m = rev.getMetadata();
                            m.addMetadata("sourceApplicationName", application);
                        }
                    }
                }
            }
        }
    }
}
```

## Importer

```
package no.inspera.services.importer;

import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.services.log.LogServices;
import org.apache.log4j.Category;

/**
 * Abstract superclass for all importers.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: Importer.java,v 1.5 2003/05/26 13:10:34 jorgen Exp $
 */
public abstract class Importer {

    // instance variables.
    protected long marketplaceId;
    protected long languageId;
    protected String mimeType;

    private static Category logCat = LogServices.getCategory(Importer.class);

    /**
     * Gets an appropriate importer for the mimetype.
     * @param mimeType mimetype to get importer for
     * @param marketplaceId
     * @param languageId
     * @return subclass for importing files of this mimetype
     */
    public static Importer getImporter(String mimeType, long marketplaceId, long
languageId) {
        Importer i = null;
        if (mimeType == null) mimeType = "";
        logCat.debug("mimetype: "+mimeType);
        // TODO: get this from db instead
        if (mimeType.equals("text/xml") || mimeType.equals("text/plain"))
            i =  new XMLImporter();
        else if (mimeType.equals("application/msword"))
            i = new WordImporter();
        else if (mimeType.equals("application/rtf") ||
mimeType.equals("text/richtext"))
            i = new RTFImporter();
        else if (mimeType.equals("image/jpeg") || mimeType.equals("image/gif") ||
mimeType.equals("image/bmp"))
            i = new ImageImporter();
        else if (!mimeType.equals(""))
            i = new FileImporter();
        else
            i = new EmptyImporter();

        i.marketplaceId = marketplaceId;
        i.languageId = languageId;
        i.mimeType = mimeType;
        return i;
    }

    /**
     * Import-method that all subclasses has to implement.
     * @param data File to import.
     * @return Imported ContentRevision subclass.
     */
```

```
    public abstract ContentRevision doImport(byte[] data);

}
```

### RTFImporter

```
package no.inspera.services.importer;

import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.kernel.Stylesheet;
import no.inspera.kernel.dispatcher.TemplateEngine;
import no.inspera.db.StylesheetDataAccess;
import no.inspera.services.log.LogServices;

import javax.xml.transform.Transformer;
import java.io.*;

import org.xml.sax.InputSource;
import org.apache.log4j.Category;
import com.tetrasix.majix.rtf.RtfReader;
import com.tetrasix.majix.rtf.RtfAnalyser;
import com.tetrasix.majix.rtf.RtfDocument;
import com.tetrasix.majix.rtf.RtfInfo;
import com.tetrasix.majix.xml.XmlWriter;
import com.tetrasix.majix.xml.ConversionTemplate;
import com.tetrasix.majix.xml.XmlGenerator;
import com.tetrasix.majix.templates.ConversionTemplateFactory;

/**
 * Importer for RTF files.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: RTFImporter.java,v 1.3 2003/05/26 13:10:34 jorgen Exp $
 */
public class RTFImporter extends Importer{

    private static Category logCat = LogServices.getCategory(RTFImporter.class);

    /**
     * Imports the file.
     * @param data File to import.
     * @return ContentDocument.
     */
    public ContentRevision doImport(byte[] data) {
        String docbook = convertFromRTFToDocBook(new String(data));
        String xml = convertFromDocbookToInsperaXML(docbook);
        logCat.debug("After transformation: "+xml);
        Importer xmlImporter = getImporter("text/xml", marketplaceId, languageId);
        return xmlImporter.doImport(xml.getBytes());
    }

    /**
     * Handles the conversion from RTF to DocBook XML.
     * @param toConvert RTF file
     * @return DocBook XML
     */
    private String convertFromRTFToDocBook(String toConvert){
        StringBuffer retval  = new StringBuffer();
               try {
                       RtfReader reader;
                       try {
                               reader = new RtfReader(toConvert);
                       }catch (FileNotFoundException e) {
                               System.out.println("Error occurred when initializing
RtfReader");
                               return null;
                       }
                       ByteArrayOutputStream bos = new ByteArrayOutputStream();
                       XmlWriter xw = new XmlWriter(bos);
                       RtfAnalyser ana = new RtfAnalyser(reader, xw);
                       RtfDocument doc = ana.parse();
            ConversionTemplateFactory factory = new ConversionTemplateFactory();
            ConversionTemplate template = factory.getConversionTemplate("inspera");
            logCat.info("template: "+template.getName());

                       XmlGenerator gen = new XmlGenerator(template);
                       gen.generate(doc, xw, null);
```

```
                    xw.flush();
                    xw.close();
                    reader.close();
            logCat.info("Convertet from RTF:\n "+bos.toString());
            //first append metadata
            RtfInfo metadata = doc.getInfo();
            retval.append("<article>");
            if (metadata != null){
                retval.append("<metaData>");

retval.append("<title>"+metadata.getProperty(RtfInfo.TITLE)+"</title>");

retval.append("<description>"+metadata.getProperty(RtfInfo.SUBJECT)+"</description>");

retval.append("<company>"+metadata.getProperty(RtfInfo.COMPANY)+"</company>");

retval.append("<comments>"+metadata.getProperty(RtfInfo.COMMENTS)+"</comments>");

retval.append("<creationDate>"+metadata.getProperty(RtfInfo.CREATIONTIME)+"</creationD
ate>");
                logCat.info("creationDate:
'"+metadata.getProperty(RtfInfo.CREATIONTIME)+"' , version:
'"+metadata.getProperty(RtfInfo.VERSION)+"' ,application:
'"+metadata.getProperty(10)+"', keywords: '"+metadata.getProperty(8)+"'");

retval.append("<keywords>"+metadata.getProperty(RtfInfo.KEYWORDS)+"</keywords>");

retval.append("<version>"+metadata.getProperty(RtfInfo.VERSION)+"</version>");

retval.append("<application>"+metadata.getProperty(RtfInfo.APPLICATION)+"</application
>");
                retval.append("</metaData>");
                String author =metadata.getProperty(RtfInfo.AUTHOR);
                retval.append("<contributors>");
                if (author!=null) {
                    String firstName ="";
                    String lastName ="";
                    try {
                        firstName=author.indexOf(" ")!=-
1?author.substring(0,author.indexOf(" ")+1):author.substring(0,author.indexOf(",")+1);
                        lastName =author.indexOf(" ")!=-
1?author.substring(author.indexOf("
")+1,author.length()):author.substring(author.indexOf(",")+1,author.length());
                    } catch (Exception e) {
                        logCat.error("error:", e);
                    }
                    retval.append("<contributor>");

retval.append("<contributorFirstName>"+firstName+"</contributorFirstName>");

retval.append("<contributorLastName>"+lastName+"</contributorLastName>");
                    retval.append("<contributorTitle/>");
                    retval.append("<roleType>author_rel</roleType>");
                    retval.append("</contributor>");
                }
                retval.append("</contributors>");
            }

            String s=bos.toString();
            retval.append(s.substring(s.indexOf("<article>")+9,s.length()));

               }

               catch(IOException e){
                       logCat.error("error: ",e);
               }
        return retval.toString();
    }

    /**
     * Handles the conversion from DocBook XML to Inspera XML, done by XSLT.
     * @param toConvert DocBook XML
     * @return Inspera XML
     */
    private String convertFromDocbookToInsperaXML(String toConvert){
        String retval=null;
```

```
        Stylesheet sh =
StylesheetDataAccess.getInstance().getStylesheet("DOC_BOOKXML2XML", marketplaceId);
        Transformer transformer = TemplateEngine.newFileTransformer(sh,
languageId,marketplaceId, true);
        StringWriter str = new StringWriter();
        PrintWriter output = new PrintWriter(str);
        try {
            TemplateEngine.transformXSLT(new InputSource(new StringReader(toConvert)),
transformer, new javax.xml.transform.stream.StreamResult(output));
            retval=str.toString();
        } catch (Exception e) {
            logCat.error("error:", e);
        }
        return retval;
    }
}
```

### WordImporter

```
package no.inspera.services.importer;

import no.inspera.applications.publish.datamodel.ContentRevision;
import no.inspera.applications.publish.datamodel.ContentDocument;
import no.inspera.applications.publish.util.converters.Converter;
import no.inspera.services.log.LogServices;

import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

import org.apache.log4j.Category;

/**
 * Imports MS Word .doc files.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id: WordImporter.java,v 1.2 2003/05/20 14:33:05 johnarne Exp $
 */
public class WordImporter extends Importer{

    private static Category logCat = LogServices.getCategory(WordImporter.class);

    /**
     * Converts the supplied .doc file to a ContentDocument
     * @param data File to import
     * @return ContentDocument
     */
    public ContentRevision doImport(byte[] data) {

        String rtf = null;
        //String host = depbundle.getString("CONVERTER_SERVER_IP");
        String host = "rmi://localhost:1099/Converter";
        logCat.info("got converter host... "+host);
        try {
            Converter c = (Converter)Naming.lookup(host);
            logCat.info("looked up and converter class name is
"+c.getClass().getName());
            rtf = (String)c.convert(new String(data));
        } catch (NotBoundException e) {
            logCat.error("NotBoundException:", e);
        } catch (java.net.MalformedURLException e) {
            logCat.error("MalformedURLException:", e);
        } catch (RemoteException e) {
            logCat.error("RemoteException:", e);
        }
        if (rtf == null) return null;
        Importer rtfImporter = getImporter("text/richtext", marketplaceId,
languageId);

        return rtfImporter.doImport(rtf.getBytes());
    }
}
```

### FMImporter

```
package no.inspera.services.importer;

import no.inspera.applications.publish.datamodel.ContentRevision;
```

```
import no.inspera.kernel.Stylesheet;
import no.inspera.kernel.dispatcher.TemplateEngine;
import no.inspera.db.StylesheetDataAccess;
import no.inspera.services.log.LogServices;

import javax.xml.transform.Transformer;
import java.io.StringWriter;
import java.io.PrintWriter;
import java.io.StringReader;

import org.xml.sax.InputSource;
import org.apache.log4j.Category;

/**
 * Inports FraneMaker XML files.
 *
 * @author <a href="mailto:johna@inspera.no">John Arne S. Pedersen</a>
 * @version $Id$
 */
public class FMImporter extends Importer{

    private static Category logCat = LogServices.getCategory(FMImporter.class);

    /**
     * Converts a FrameMaker XML file to ContentDocument by XSLT to Inspera XNL.
     * @param data File to import.
     * @return ContentDocument
     */
    public ContentRevision doImport(byte[] data) {
        String xml=null;
        Stylesheet sh =
StylesheetDataAccess.getInstance().getStylesheet("FRAMEXML2XML", marketplaceId);
        Transformer transformer = TemplateEngine.newFileTransformer(sh,
languageId,marketplaceId, true);
        StringWriter str = new StringWriter();
        PrintWriter output = new PrintWriter(str);
        try {
            TemplateEngine.transformXSLT(new InputSource(new StringReader(new
String(data))), transformer, new javax.xml.transform.stream.StreamResult(output));
            xml=str.toString();
        } catch (Exception e) {
            logCat.error("error:", e);
        }
        logCat.debug("After transformation: "+xml);
        Importer xmlImporter = getImporter("text/xml", marketplaceId, languageId);
        return xmlImporter.doImport(xml.getBytes());
    }
}
```

# *Vedlegg B: Stylesheets for konvertering*

## B-1 DocBook XML til Inspera XML

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:template match="/article">
    <document>
        <xsl:apply-templates select="metaData"/>
        <xsl:call-template  name="content"/>
    <xsl:apply-templates select="contributors"/>
</document>
</xsl:template>

<xsl:template name="content">
    <contentSection>
        <xsl:apply-templates select="*[local-name() != 'metaData' and local-name() !=
'contributors']|text()"/>
    </contentSection>
</xsl:template>

<xsl:template match="section">
    <headline level="{@level}"><xsl:value-of select="title"/></headline>
    <para>
        <xsl:apply-templates select="*|text()"/>
    </para>
</xsl:template>

<xsl:template match="metaData">
    <metaData>
        <title><xsl:value-of select="title"/></title>
        <description><xsl:value-of select="description"/></description>
        <sourcecreationdate><xsl:value-of select="creationDate"/></sourcecreationdate>
        <sourcerevisionnumber><xsl:value-of select="version"/></sourcerevisionnumber>
        <comments><xsl:value-of select="comments"/></comments>
        <keywords><xsl:value-of select="keywords"/></keywords>
    </metaData>
</xsl:template>

<xsl:template match="contributors">
    <contributors>
        <xsl:apply-templates select="*|text()"/>
    </contributors>
</xsl:template>

<xsl:template match="contributor">
    <contributor>
        <xsl:apply-templates select="*|text()"/>
    </contributor>
</xsl:template>

'<xsl:template match="contributorFirstName">
    <contributorFirstName>
        <xsl:apply-templates select="*|text()"/>
    </contributorFirstName>
</xsl:template>

<xsl:template match="contributorLastName">
    <contributorLastName>
        <xsl:apply-templates select="*|text()"/>
    </contributorLastName>
</xsl:template>

<xsl:template match="roleType">
    <roleType>
        <xsl:apply-templates select="*|text()"/>
    </roleType>
</xsl:template>

<xsl:template match="contributorTitle">
    <contributorTitle>
```

```
        <xsl:apply-templates select="*|text()"/>
    </contributorTitle>
</xsl:template>

<xsl:template match="pageInfo">
    <!-- just used to avoid output from default templates -->
</xsl:template>

<xsl:template match="pageCaptions">
    <!-- just used to avoid output from default templates -->
</xsl:template>

<xsl:template match="pagebreak">
    <img src="{$admingfx}blackLine.gif" xmlElement="pagebreak" xmlAttribute-
page="{@page}"/>
</xsl:template>

<xsl:template match="title">
    <strong><xsl:value-of select="."/></strong>
</xsl:template>

<xsl:template match="para[.='']">
    <linebreak/>
</xsl:template>

<xsl:template match="resourceMetaData">
    <span xmlElement="resourceMetaData" author-lastName="{author/lastName}" author-
firstName="{author/firstName}">[Forfatter gjort søkbar, redigeres ikke her]</span>
</xsl:template>

<xsl:template match="para">
        <xsl:apply-templates select="*|text()"/><linebreak/>
</xsl:template>

<xsl:template match="emphasis">
    <emphasis><xsl:apply-templates select="*|text()"/></emphasis>
</xsl:template>

<xsl:template match="strong">
    <strong><xsl:apply-templates select="*|text()"/></strong>
</xsl:template>

<xsl:template match="underline">
    <underline><xsl:apply-templates select="*|text()"/></underline>
</xsl:template>

<xsl:template match="alfabetizedList">
    <alfabetizedList><xsl:apply-templates select="*|text()"/></alfabetizedList>
</xsl:template>

<xsl:template match="itemizedList">
    <itemizedList><xsl:apply-templates select="*|text()"/></itemizedList>
</xsl:template>

<xsl:template match="orderedList">
    <orderedList><xsl:apply-templates select="*|text()"/></orderedList>
</xsl:template>

<xsl:template match="listItem">
    <listItem><xsl:apply-templates select="*|text()"/></listItem>
</xsl:template>

<xsl:template match="alfaItem">
    <alfaItem><xsl:apply-templates select="*|text()"/></alfaItem>
</xsl:template>

<xsl:template match="link">
    <a xmlElement="link" xmlAttribute-type="{@type}" xmlAttribute-ref="{@ref}"
xmlAttribute-style="{@style}" xmlAttribute-windowStyle="{@windowStyle}" xmlAttribute-
windowWidth="{@windowWidth}" xmlAttribute-windowHeight="{@windowHeight}"><xsl:apply-
templates select="*|text()"/></a>
</xsl:template>

<xsl:template match="siteLink">
    <span xmlElement="siteLink" xmlElementContent="{text()}"
logicalTitle="{@logicalTitle}" contentSectionId="{@contentSectionId}" style="color:
```

```
#306699;">[Site-lenke (ikke redigerbar her):<xsl:apply-templates
select="*|text()"/>]</span>
</xsl:template>

<xsl:template match="extLink">
    <a href="{@url}" target="{@windowStyle}"><xsl:apply-templates
select="*|text()"/></a>
</xsl:template>

<xsl:template match="linebreak">
    <span xmlElement="linebreak"><br/></span>
</xsl:template>

<xsl:template match="headline[@level = '1']">
    <h1><xsl:value-of select="."/></h1>
</xsl:template>

<xsl:template match="headline[@level = '2']">
    <h2><xsl:value-of select="."/></h2>
</xsl:template>

<xsl:template match="headline[@level = '3']">
    <h3><xsl:value-of select="."/></h3>
</xsl:template>

<xsl:template match="headline[@level = '4']">
    <h4><xsl:value-of select="."/></h4>
</xsl:template>

<xsl:template match="headline">
    <h3><xsl:value-of select="."/></h3>
</xsl:template>

<xsl:template match="table[@xmlElement='table']|TABLE[@xmlElement='table']">
    <table border="{@xmlAttribute-border}" padding="{@xmlAttribute-padding}"
style="{@xmlAttribute-style}"><xsl:apply-templates select="*"/></table>
</xsl:template>

<xsl:template match="table[@xmlElement='table']|TABLE[@xmlElement='table']">
    <table><xsl:apply-templates select="*"/></table>
</xsl:template>

<xsl:template match="table">
    <table border="1"><xsl:apply-templates select="*"/></table>
</xsl:template>

<xsl:template match="row">
    <row><xsl:apply-templates select="*"/></row>
</xsl:template>

<xsl:template match="cell">
    <cell><xsl:apply-templates select="*"/></cell>
</xsl:template>

<xsl:template match="td[@xmlElement = 'cell']|TD[@xmlElement = 'cell']">
    <cell align="{@xmlAttribute-align}" style="{@xmlAttribute-style}"><xsl:apply-
templates select="*|text()"/></cell>
</xsl:template>

<xsl:template match="tbody|TBODY">
    <xsl:apply-templates select="*"/>
</xsl:template>

<xsl:template match="tr[@xmlElement = 'row']|TR[@xmlElement = 'row']">
    <row height="{@xmlAttribute-height}"><xsl:apply-templates select="*"/></row>
</xsl:template>

<xsl:template match="td[@xmlElement = 'cell']|TD[@xmlElement = 'cell']">
    <cell align="{@xmlAttribute-align}" style="{@xmlAttribute-style}"><xsl:apply-
templates select="*|text()"/></cell>
</xsl:template>

</xsl:stylesheet>
```

## B-2 FrameMaker XML til Inspera XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml" indent="yes"/>

<!-- Denne xsl viser proof-of-consept for import av xml generert av framemaker, og er
langt fra komplett. -->

<xsl:template match="XML">&lt;document&gt;<xsl:apply-templates
select="*|text()"/>&lt;/document&gt;</xsl:template>

<xsl:template match="Heading1">&lt;heading level="1"&gt;<xsl:value-of
select="."/>&lt;/heading&gt;</xsl:template>

<xsl:template match="Heading2">&lt;heading level="2"&gt;<xsl:value-of
select="."/>&lt;/heading&gt;</xsl:template>

<xsl:template match="TITLE"/>

<xsl:template match="DIV"><xsl:apply-templates select="*|text()"/></xsl:template>

<xsl:template match="Body">&lt;para&gt;<xsl:value-of
select="."/>&lt;/para&gt;</xsl:template>

</xsl:stylesheet>
```