Raw repo

Formålet med Raw repo (RR) er at gemme poster der kommer ind i brønden før posterne bliver ændret, således at vi internt altid har adgang til posterne i deres rå form. Herved har vi mulighed for f.eks. at genkøre brønden uden at skulle have poster gensendt fra biblioteker, og vi kan modtage og kvittere for poster uden hensyn til hvor lang tid efterbehandlingen tager.

RR ved ikke noget omkring indholdet af posterne, og kan derfor ikke foretage handlinger på baggrund af post indehold. Der er indført et type begreb - posterne kan have en mime type der dikterer hvilke aftager køer posten havner på, men de skal sættes udefra, inden posten havner i repositoriet. Udover mime type kan posterne have relationer til hinanden. Relationer bliver brugt ved sammenskrivning og udlevering af poster. Hvordan sammenskrivning og validering af relationer foregår er bestemt af den specifikke posts mime type.

Viden omkring en posttype ligger i kode udenfor RR, og det er her datamodellen for den specifikke posttype skal implementeres. RR er generisk, og tilbyder generelle primitiver der kan bruges til implementere forskellige datamodeller for posttyper.

Poster

Poster er unikt identificeret ved hjælp af to felter: *Bibliographic Record ID* og *Agency ID*. Navnene stammer fra markpost verdenen, men deres funktion i RR er mere generelle. *Bibliographic Record ID* og *Agency ID*, bliver brugt til at finde og udlevere de relevante poster, samt eventuelt at sammenskrive poster inden udlevering. Dette bliver nærmere beskrevet i afsnittet om relationer.

Mime type

Poster i RR har en mime type defineret, der bliver angivet ved indsendelse. Denne mime type definerer semantikken omkring posttypen, nærmere bestemt hvorledes relationer mellem poster af denne type kan udformes, og hvordan sammenskrivninger foretages. På nuværende tidspunkt håndterer RR kun marcXchange og autoritetsposter (mime typer: `text/marcxchange` og `text/authority+marcxchange`), men RR er designet til at kunne håndtere andre post typer, f.eks. pdf, eller jpg, der ikke har samme sammenskrivningsregler.

Relationer

RR understøtter to relationstyper; søskenderelationer (horisontal), og generationsrelationer (vertikale)

Søskenderelationer

søskenderelationer kan oprettes mellem poster med samme *Bibliographic Record ID*, og søskende poster sammenskrives ved udlevering.

Poster kan have en udgående relation af denne type og flere indgående, og danner dermed et træ, med 1 rod knude. RR er ikke designet til at håndtere cykliske relationer, men der er ikke noget i øjeblikket til at detektere disse, og det er brugerens ansvar ikke at introducere dem.

I marcpost sammenhæng bliver søskenderelationer brugt til at sammenskrive poster inden udlevering. Nedenstående figur indeholder 4 bind poster, og derfor fire poster der kan udleveres.

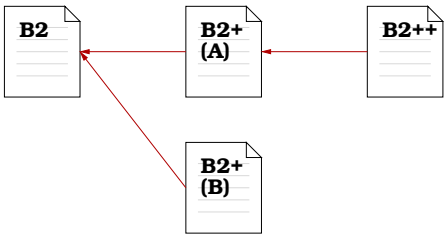

Figure 1: Søskenderelationer

1. **B2**: B2 har ingen udgående søskenderelationer, og bliver der udleveret som den er.
2. **B2+(A)**: **B2+(A)** har en udgående relation til **B2**, og derfor bliver **B2+(A)**'s indhold indsat i **B2**, og den sammenskrevne post bliver udleveret.
3. **B2+(B)**: **B2+(B)** har en udgående relation til **B2**, og derfor bliver **B2+(B)**'s indhold indsat i **B2**, og den sammenskrevne post bliver udleveret.
4. **B2++**: **B2++** har en udgående relation til **B2+(A)**, der igen har en udgående relation til **B2**, så her foretages 2 sammenskrivninger inden udlevering først sammenskrives **B2** og **B2+(A)** som beskrevet i punkt. 2, hvorefter indholdet af **B2++** bliver indsat i denne post og resultatet bliver udleveret.

Metoder Følgende metoder kan bruges til at hente informationer om en posts søskenderelationer:

- `GetRelationsSiblingsToMe`: Indgående søskenderelationer.
- `GetRelationsSiblingsFromMe`: Udgående søskenderelationer.

Sammenskrivningen af poster foretages ved hjælp af en merger. I øjeblikket er der kun understøttelse for sammenskrivning af marc poster, vha. `MarcXMerger`, men en passende merger kan bruges til andre scenarier.

Metoderne findes i RawRepoDAO

Generationsrelationer

Generationsrelationerne bruges ved udlevering af poster. Poster kan have flere indgående og flere udgående generationsrelationer, men der er begrænsninger når poster både har søskende- og generationsrelationer (se afsnittet om begrænsninger).

generationsrelationerne danner grundlag for hvilke poster der bliver udleveret sammen med den rekvirerede post. Nærmere bestemt bliver alle poster "over" den ønskede post udleveret sammen med posten, dvs. poster man finder ved at følge udgående generationsrelationer.

Nedenstående figur viser en række poster der har generations relationer mellem dem, og et par beskrevne eksempler på udhentning..

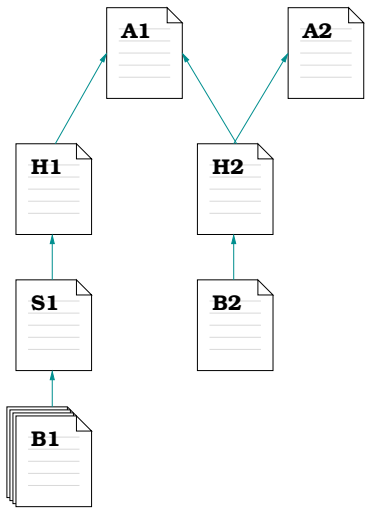

Figure 2: generationsrelationer

1. **S1**: Ved udhentning af **S1** bliver generations relationerne fulgt, og følgende poster bliver udleveret: **S1**, **H1**, **A1**.
2. **B2**: Ved udhentning af **B2**, bliver følgende poster udleveret: **B2**, **H2**, **A1**, **A2** (bemærk at begge autoritets poster bliver udleveret).

Metoder Følgende metoder kan bruges til at hente informationer om en posts generationsrelationer:

- `GetRelationsChildren`: Udgående generationsrelationer.
- `GetRelationsParents`: Indgående generationsrelationer.

Metoderne findes i RawRepoDAO

Begrænsninger af relationsdannelse

Der begrænsninger for brug af relationstyper når både søskende- og generationsrelationer bliver brugt:

Poster kan ikke både have udgående søskenderelationer og udgående generationsrelationer, til poster med mimetypen `text/marcxchange`. Man kan godt have udgående søskenderelationer og udgående generationsrelationer til autoritestsposter (mimetype `text/authority+marcxchange`)

Denne begrænsning er indført da det ellers vil blive svært at identificere hvilke poster der skal udleveres, og hvordan sammenskrivning foretages, fordi den post man har en søskenderelation til, kan have en anden generationsrelation.

Sletning af poster

Man kan slette poster fra RR. Sletningen foregår ved hjælp af en markering, så RR indeholder de slettede poster. Det betyder at man kan hente slettede poster fra RR.

På DAOen er følgende metoder implementeret til at undersøge eksistens af poster:

- `recordExists` Returnere true hvis posten eksisterer, og ikke er slettemarkeret.
- `recordExistsMaybeDeleted` Returnere true hvis posten eksisterer, selvom den er slettemarkeret.

Versionering

RR Implementere et versionering så det er muligt at udhente ældre udgaver af posten.

RR gemmer alle udgaver af en post der er yngre en den definerede skæringsdato, og den yngste udgave der er ældre end skæringsdatoen.

Scenarier Skæringsdato er 42 dage.

1. Post X er inddateret 1 gang. RR indeholder derfor kun 1 udgave.
2. Post X er inddateret for et år siden, og rettet 4 gange i går. RR indeholder alle 5 udgaver.
3. Post X er inddateret for et år siden, rettet for 9 måneder siden, og rettet i går. RR indeholder 3 udgaver: posten som den så ud for 9 måneder siden, posten som den så ud i går, og den aktuelle udgave.

Eksempler

3 Udhentnings eksempler:

- Udhentning af **S1**: Ved udhentning af **S1**, foretages ingen sammenskrivninger, og følgende poster vil blive udleveret: **S1**, **H1** og **A1**.
- Udhentning af **H2+**: Først bliver **H2** og **H2+** sammenskrevet, og følgende poster bliver udleveret: Den sammenskrevne **H** post, **A1** og **A2**.
- Udhentning af **B2++**: Først bliver **B2**, **B2+** og **B2++** sammenskrevet, og følgende poster bliver udleveret: Den sammenskrevne **B** post, **H2**, og **A1**, og sammenskrivningen af **H2**, **H2+**, og **H2++**, hvis **H2++** har samme *Agency id* som **B2++**

Dokumentation af API

http://is.dbc.dk/job/rawrepo/javadoc/

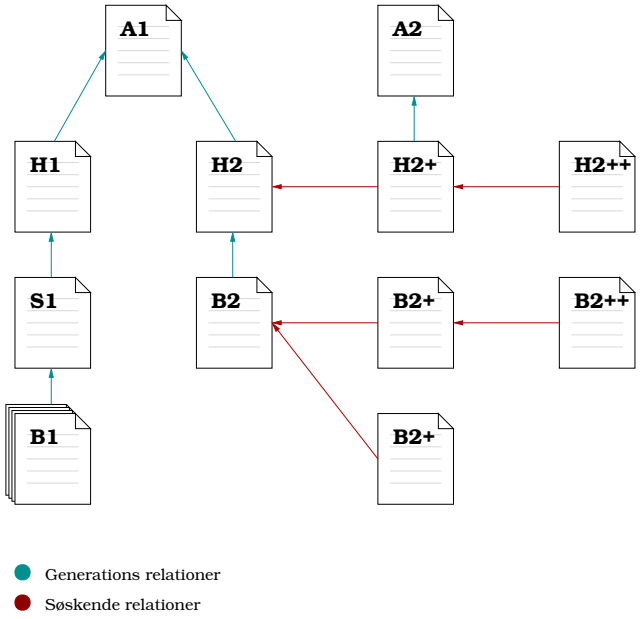

Figure 3: Poster i RR