# Aanwinsten - werking

Technische uitleg over de werking van de programmatuur.

# Achtergrondinformatie

## Marc4J

Voor het verwerken van de Marc21-records wordt library Marc4J gebruikt. Het `Record` interface in die library beschrijft een Marc21-record.

## New en Updates

Er zijn altijd twee MARC21-bestanden voor een collectie, een bestand met nieuwe records en een bestand met updates. De diectories waar deze bestanden op de FTP-server staan zijn configureerbaar. Zie daarvoor settings `aanwinsten.ftp.directory.new` en `aanwinsten.ftp.directory.updates`. Aanduiding 'new' of 'updates' komt ook nog terug in de bestandsnamen. Template-variabele `${set}` in setting `aanwinsten.ftp.file-template` wordt vervangen door 'new' of 'updates'.

Aanwinsten haalt altijd eerst het 'new'-bestand op en voegt er 'updates' aan toe. records in 'updates' overschrijven 'new' records.

## Titel-records en holding-records

Als een MARC21-record een 001-record bevat, dan is dat een titel-record, oftewel een record dat een boek beschrijft (titel, auteur, enzovoorts). Als een MARC21-record géén 001-record heeft, maar wél een 004-record, dan is dat een holding-record. Het beschrijft dan een exemplaar. Belangrijkste data in het exemplaar is de vindplaats (852$h). Veld 001 in het titel-record is het zogenaamde OCN (een key) en veld 004 in het holding-record is een referentie. Er kunnen meerdere holding-records zijn voor een titel-record.

Bij de scripties zijn er alleen titel-records.

## Map met titel-records

De titel-records worden bij het verwerken van een collectie opgeslagen in een hashMap<String, Record>, waarbij de OCN de key is.

Zie class `CollectieAanwinsten` member `titelRecords`.

## Tree met sets van holding-records

De rubrieken vormen een boomstructuur. Het hoofdstuk over rubrieken en de configuratie ervan beschrijft dat. Elke rubriek heeft een HashMap<String, Record> voor de holding-records die aan die rubriek zijn toegewezen.

Zie class `Section` member `holdingRecords`.

Voor de scripties is er daarnaast een Set<String> met alleen keys van records.

Zie class `Section` member `keys`.

## Map met tellingen holding-records

Wanneer van een aanwinst een exemplaar in een kast wordt gezet, maar ook een exemplaar in het magazijn, dan wil men niet dat die aanwinst ook voor het magazijn (rubriek CBM) vermeld wordt op de aanwinstenlijst. Is er van een aanwinst enkel een exemplaar voor het magazijn, dan moet het juist wél op de aanwinstenlijst in rubriek 'CBM'. Om te weten of een exemplaar uniek is, wordt het aantal holding-records per titel geteld in member `holdingsCounts` van class `ColectieAanwinsten`. Voor CBM is dan geregeld dat een record alleen wordt afgedrukt als het uniek is (aantal = 1).

## Merge van records

Class MarcRecordDecorator implementeert een Decorator-patroon (zie GOF). Het bevat een `List<Record>` (een lijst met Marc21-records dus) en implementeert het Record-interface van Marc4J. Aan zo'n decorator kunnen meerdere records worden toegevoegd. Typisch wordt daar een titel-record en een holding-record gestopt. Vraag je nu de waarde van een veld op aan de decorator, dan zoekt die de records in de lijst af tot het veld gevonden is en geeft dat terug. Zo vindt er een soort zachte merge plaats, zonder dat records echt worden aangepast.

Het is een decorator omdat het ook functionaliteit toevoegt. De class-definitie geeft aan: `extends StrLookup<String>`. Daarmee kan een object van dit type ook dienen als bron voor het vervangen van `${123-a}`-constructies in de record- en thesis-templates. Voor al die `${}`-vervangingen wordt StrSubstitutor van Apache Commons gebruikt. Deze kan een HashMap met key/value-paren gebruiken om in een template in te vullen. Dat gebeurt op plaatsen in class `RestructuredTextFormatter`. Maar de StrSubstitutor-class kan ook een object van type `StrLookup<String>` gebruiken voor het opzoeken van waarden. In dat geval wordt method `lookup(v)` van het object aangeroepen met dat wat tussen de accolades staat. Class RecordDecorator kan op die manier de waarden teruggeven van variabelen als `${100-a}`.

Bij het maken van de lijst, loopt de `RestructuredTextFormatter` recursief door de rubrieken (sections) heen en merget voor elk holding record het titel-record uit de HashMap met het holding-record en laat het vervolgens via het Record-template omzetten in een stukje reStructuredText. Voor keys van scripties wordt alleen het titel-record uit de HashMap gehaald en in een nieuw decorator-object gestopt en wordt het met het Thesis-template omgezet naar een stukje reStructuredtext.

## Transformatie van reStructuredText naar HTML en PDF

Aan de hand van de documentatie over de templates is goed na te gaan hoe een compleet reStructuredText-document tot stand komt. Die .rst-documenten staan ook naast de HTML- en PDF-documenten.

Het basiscommando om een reStructuredtext-document om te zetten naar HTML is:

```
rst2html --language=nl_NL <opties> <bron> <doel>
```

Dit is voor Nederlands. Voor Engels wordt `--language=en_UK` gebruikt. De --language-vlag beïnvloedt voor de huidige templates alleen de tekst boven de inhoudsopgave.

Hier is <bron> het pad naar het reStructuredText-bestand en <doel> het HTML-bestand dat gemaakt moet worden. Programma `rst2html` regelt het invullen van dat. Programma `rst2html` koppelt een eigen CSS aan de HTML, mkaar die is niet zo mooi en volgt zeker geen Tilburg University-vormgeving. Dat kan je aanpassen, en dat is wat in setting `aanwinsten.rst2html-options` ook gebeurt. Wat in die setting staat komt op de plaats van <opties>. Er staat nu: `aanwinsten.rst2html-options = --no-toc-backlinks --stylesheet-path=/etc/aanwinsten/basic.css`. Door optie 'stylesheet-path' wordt de inhoud van bestand `/etc/aanwinsten/basic.css` als stylesheet toegevoegd aan de HTML. De

`--no-toc-backlinks`-setting zorgt dat rubriekstitels niet een link worden naar de inhoudsopgave.

Het basiscommando om een reStructuredtext-document om te zetten naar PDF is:

```
rst2pdf --language=nl_NL <opties> --output=<doel> <bron>
```

Hier is <bron> het pad naar het reStructuredText-bestand en <doel> het PDF-bestand dat gemaakt moet worden. Programma `rst2pdf` heeft een eigen stylesheet voor de vormgeving. Dat kan je aanpassen, en dat is wat in setting `aanwinsten.rst2pdf-options` ook gebeurt. Wat in die setting staat komt op de plaats van <opties>. Er staat nu: `aanwinsten.rst2pdf-options = --stylesheets=/etc/aanwinsten/alignleft.pdfstyle,dejavu,tenpoint`. Het verandert een paar instellingen. In bestand `/etc/aanwinsten/alignleft.pdfstyle` is geregeld dat een aantal titels niet gecentreerd moet staan, maar links uitgelijnd. Stijl 'dejavu' is een ingebouwde stijl van rst2pdf en selecteert een ander lettertype (Dejavu) dat meer Unicode-tekens ondersteunt. Stijl 'tenpoint' is ook een ingebouwde stijl van rst2pdf en selecteert de gewenste lettergrootte.

# Ad hoc lijsten

Het programma biedt ook de mogelijkheid om ad hoc-lijsten te maken. Een ad hoc-lijst is een lijst gebaseerd op een of meer Marc-bestanden die door gebruikers in een speciale directory zijn gezet. Configuratie-instelling `aanwinsten.adhoc-lists.input.base-directory` geeft aan waar het programma de Marc-bestanden voor de ad hoc lijsten moet zoeken. Om een ad hoc-lijst te maken, moeten gebruikers een nieuwe subdirectory maken in die base-directory en daarin dus een of meer Marc-bestanden zetten. Er kan worden afgesproken dat automatisch elk uur het commando `maak-adhoc-lijsten` wordt aangeroepen. Wanneer dat commando wordt gebruikt, controleert `aanwinsten` alle subdirectories in de directory die is aangegeven met configuratie `aanwinsten.adhoc-lists.base-directory`. Per subdirectory controleert het of er Marc-bestanden in staan (*.mrc) en of er al een doelbestand is gemaakt in de output-directory. Er wordt gecontroleerd of één van de doelbestanden er is. Is zo'n bestand er niet, dan verwerkt het programma alle Marc-bestanden en maakt de hieronder opgesomde lijst van bestanden aan.

De output kan in een andere directory komen dan de input, maar dat hoeft niet. Hier een voorbeeld van gescheiden directories in de test-omgeving:

```
src/test/resources/ad-hoc
├── input
│   ├── experiment-1
│   │   └── testincidentelelijsten.mrc
│   └── experiment-2
│       └── MARCsubset.mrc
└── output
    └── experiment-1
        └── lijst.csv
```

Hier is de `input`-directory gezet als `aanwinsten.adhoc-lists.input.base-directory` en de `output`-directory als `aanwinsten.adhoc-lists.output.base-directory`. De output komt in een eenzelfde subdirectory te staan als de input-directory. Voor `experiment-1` is er een doelbestand, maar voor `experiment-2` nog niet. Bij de eerstvolgende run van dit commando. zal een directory `experiment-2` worden aangemaakt in de `input`-directory, en daarin zal bestand `lijst.csv` worden gemaakt op basis van bestand `MARCsubset.mrc`.

Wanneer de output- en input-basis-directories verschillen, loopt het programma na afloop van het maken van de lijsten nog even de output-directories na. Als er geen corresponderende input-directory is, dan wordt de output-directory geruimd. Als dus

directory `input/experiment-1` wordt geruimd, dan ruimt dit commando directory `output/experiment-1` op.

Op dit moment is er maar één doelbestand. Mochten er later ook andere doelbestanden moeten worden gemaakt door het aanwinsten-programma, bijvoorbeeld PDF of HTML, dan blijven subdirectories ongemoeid waar alleen maar een CSV staat. Om ook voor die oude lijsten dan de nieuwe doelbestanden te maken, zullen de gebruikers de CSV moeten verwijderen. De subdirectory lijkt dan nieuw en alle dan geldende doelbestanden worden gemaakt bij de eerstvolgende run van het programma met commando `maak-adhoc-lijsten`.

Het commando maakt de volgende bestanden aan wanneer de subdirectory nieuw lijkt:

**lijst.csv**

Data van alle bibliografische records (samengevoegd met vindplaatsrecords) in een niet gespecificeerde volgorde. Voor elk bibliografisch record is er minimaal 1 regel in het CSV-bestand; het aantal hangt af van het aantal holding-records. Voor elk holding-record is er 1 regel in het CSV-bestand, waarbij de data van het holding-record is gemerged met dat van het bibliografische record. Bij twee holding-records voor een OCN, zijn er dus twee regels CSV, waarbij bibliografische data wordt herhaald. Is er voor een OCN alleen maar een bibliografisch record en geen holding-record, dat wordt er toch een regel CSV geproduceerd, en wordt er een waarschuwing gelogd. De eerste regel is een header met kolomnamen die zo goed als gaat de velden beschrijven als het Marc-veldnummer en eventueel de selectie (EDT voor editors of AUT voor auteurs). Het gebruikte CSV-formaat is dat van Excel, maar met <TAB> als scheider. De geselecteerde velden en de kolomnamen zitten nu hard gecodeerd in het programma en kunnen niet worden geconfigureerd. Dat moet op termijn een keer verbeterd worden.

# JavaDoc en commentaar

De broncode van de klassen is grotendeels voorzien van JavaDoc. Wanneer `$ mvn site` wordt aangeroepen (zie tekst 'bouwen') dan wordt JavaDoc gemaakt.

De broncode bevat ook gewoon commentaar. Met name in CLI.java is dat commentaar uitgebreid en bedoeld om ter plekke het verhaal goed uit te leggen aan een programmeur.

Teksten zijn meest in het Nederlands, maar er is ook nog wat Engelstalig commentaar.