HAIBA

LPR-kontakter, data behandler

Guide til udviklere

version 1.0

Indholdsfortegnelse

1 Formål 3

1.1 Dokumentets opbygning 3

2 System design 4

2.1 Properties 4

3 Opsætning af udviklingsmiljø 5

3.1 Kildekode 5

3.2 Byggemiljø 5

3.2.1 Dependencies 5

3.3 Database setup 6

3.3.1 Indlæggelsesdatabase 6

3.3.2 LPR database 6

3.4 Test 6

3.5 IDE 6

3.5.1 Eclipse 6

3.6 IntelliJ Idea IDE 7

3.7 Distribution 8

4 Tips og tricks 9

4.1 Tomcat out of memory 9

4.1.1 Beskrivelse 9

4.1.2 Løsning 9

5 Ændringslog 10

6 Referencer og kilder 11

# Formål

Dette dokument er en guide til nye udviklere af LPR Databehandleren. Guiden gennemgår på overordnet plan de aktiviteter, der er nødvendige for at kunne videreudvikle på servicen.

## Dokumentets opbygning

Dokumentet vil gennemgå de væsentligste dele af opsætningen af et lokalt udviklingsmiljø og afvikling af tests

Dokumentet forudsætter, at læseren har grundig kendskab til Java udvikling, Spring 3 og Maven. Kendskab til Tomcat applikationsserver 7 vil yderligere hjælpe læseren, men er ikke en forudsætning.

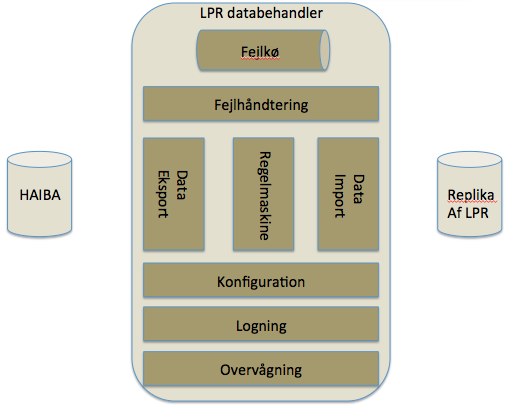
Den primære målgruppe for dokumentet er systemudviklere.

# System design

LPR databehandleren sørger for at importere data fra Landspatientregisteret (LPR), for så via nogle forretningsregler igen at eksportere data til HAIBA indlæggelsesdata.

LPR databehandleren er bygget op via Spring version 3.x (se <http://www.springsource.org/> ) som en standard JEE webapplikation. Kendskab til følgende Spring undermoduler vil være en fordel (MVC, batch og Integration).

Byggemiljøet er baseret på Maven 3, mere om dette senere.

Modulerne er strukturerede på følgende måde:

<TODO, opdater alle punkter med pakkestruktur og vigtige klasser>

* Data importeren har til ansvar at hente data ind fra LPR databasen, via batches af konfigurerbar størrelse, og placere det i en intern LPR model.

Dataimporteren ligger i pakken: dk.nsi.haiba.lprimporter.importer, klassen ImportExecutor er det skedulerede job som sørger for at importere data når noget nyt dukker op

* Regelmaskinen sørger for udførelse af de beskrevne forretningsregler i den rækkefølge som står i dette dokument, data bliver i dette step konverteret til en intern HAIBA model.

Regelmaskinen ligger i pakken: dk.nsi.haiba.lprimporter.rules, hvor klassen RulesEngine sørger for processeringen, de enkelte forretningsregler ligger også i denne pakke og implementerer alle LPRRule interfacet

* Data eksporteren sørger for at skrive det behandlede data ned i HAIBA databasen ud fra den interne HAIBA model
* Konfigurationsmodulet sørger for opsætning af LPR databehandleren ud fra den eksterne konfiguration (eksempelvis batchstørrelser, tidsintervaller m.m.)
* Logningsmodulet sørger for applikationslog til debug og applikationsfejlhåndtering - ikke fejl i forbindelse med databehandling, dette sørger fejlhåndteringsmodulet for.
* Overvågningsmodulet sørger for at udstille en html side, hvor man kan se, eller maskinelt overvåge, LPR databehandlerens tilstand.

Overvågningsmodulet ligger i pakken: : dk.nsi.haiba.lprimporter.status, hvor StatusReporter klassen, er den der bliver ramt når der spørges på status URL'en via en browser

* Fejlhåndteringsmodulet har til ansvar at rapportere datafejl, enten fordi input format ikke kan bruges, eller hvis noget data falder uden for gældende regelsæt og derved ikke kan processeres. Fejlhåndteringsmodulet skriver til en fejlkø (Fil), hvor statistikere eller klinisk personale kan se hvad det er for noget data der er fejl, og hvad fejlen lød på.

## Properties

*LPR Databehandleren* styres af java properties filer. Hvor der er en default konfigurations fil (default-config.properties) som er deployet sammen med war filen, denne kan overstyres med en properties fil lagt i Tomcat uden for war filen (config.properties) – se installations guiden for nærmere detaljer

# Opsætning af udviklingsmiljø

Opsætningen af udviklingsmiljøet for LPR Databehandleren forudsætter, at følgende elementer allerede er installeret på udviklerens maskine:

* Java Developer Kit 6.0\_x
* Et passende udviklingsmiljø (Eclipse og IntelliJ eksempler er givet i dette dokument)
* Maven 3.x
* MySQL database 5.5.x
* Tomcat 7 (Udviklet og testet på version 7.0.34)

Installationsvejledningen indeholder detaljer omkring opsætning af ovenstående komponenter.

## Kildekode

Kildekoden er placeret i et github-repositorie og kan checkes ud på følgende måde:

git clone git@github.com:trifork/HAIBA-LPRimporter.git

## Byggemiljø

LPR Databehandleren anvender Maven som byggesystem [MAVEN]. Strukturen følger de generelle anbefalinger for Maven projekter, og er struktureret

efter Maven layout konventionen.

For at bygge en LPR Databehandleren, skal man gøre følgende:

mvn install

### Dependencies

For at kunne hente NSI-specifikke afhængigheder (bl.a. nsp-util) i binær form i stedet for at skulle bygge alle afhængigheder selv på det lokale udviklingsmiljø , indeholder pom'en en reference til nexus.trifork.com, som er et artefaktrepository der er placeret hos Trifork. Binære releases af LPR Databehandleren findes også i nexus.trifork.com.

Repository’et bør, når det er muligt, udskiftes med et artefaktrepository der er driftet hos NSI. Når et sådant er etableret

.

## Database setup

### Indlæggelsesdatabase

Databaseskema for indlæggelser ligger i database/HAIBA-Indlaeggelser.sql filen, den indeholder de tabeller, views m.m. hvortil data fra LPR skal transformeres og eksporteres

### LPR database

Databaseskema for LPR ligger i database/HAIBA-LPR.sql filen, den indeholder de tabeller som LPR data vil ligge i. Data fra LPR er det grunddata som skal transformeres og eksporteres til indlæggelsesdatabasen.

<TODO, Beskriv behov for tilretning af LPR databasen med tidspunkter for migrering etc. >

<TODO, beskriv hvilke test data der findes>

## Test

Installationen kan verificeres ved at eksekvere LPR Databehandlerens test suite.

Testsuiten benytter JUnit og Mockito til test.

Testkoden er lokaliseret i:

src/test/java

Test suiten afvikles ved at udføre følgende kommando i projektroden:

mvn test

Installationen kan yderligere verificeres (Code coverage, kode konventioner o.l.) ved at udføre kommandoen:

mvn verify

## IDE

LPR Databehandleren kan principielt udvikles i enhver Java IDE, der forstår Maven projekters opbygning.

I dette dokument beskrives kort opsætning for to af de pt. mest udbredte Java IDE’er: Eclipse og IntelliJ.

### Eclipse

Eclipse er ikke født med Maven support, og det anbefales derfor, at man installerer m2eclipse inden LPR Databehandleren hentes ind i Eclipse, pluginet kan hentes her:

http://www.eclipse.org/m2e/

Herefter importeres projekterne i Eclipse via ”import”:

Alternativt kan man importere projektet ved at udføre følgende kommando:

mvn eclipse:eclipse

Og herefter importere projektet på normal vis i Eclipse.

Kommandoen genererer Eclipse projektfilerne (.project og .classpath) for LPR Databehandleren. Denne metode kræver dog, at kommandoen udføres hver gang man ændrer i pom filerne.

## IntelliJ Idea IDE

IntelliJ Idea er født med Maven support, og LPR Databehandleren kan derfor direkte importeres. Projektet importeres i IntelliJ ved under ”Create new project” at vælge ”Import project from external model”. Herefter udvælges roden af LPR Databehandleren, hvorefter projektet importeres.

Det anbefales i den sammenhæng, at man krydser af i ”Import Maven projects automatically”, hvorefter IntelliJ selv detekterer nye moduler i projektet.

Alternativt kan man importere projektet ved at udføre følgende kommando:

mvn idea:idea

Herefter kan projektet importeres på normal vis i IntelliJ.

*Obs! Denne metode kræver dog, at kommandoen udføres hver gang man ændrer i pom filerne.*

## Distribution

LPR Databehandleren kan bygges til distribution eller lokal test ved at udføre:

mvn package

Dette generer en WAR fil, der efterfølgende kan deployeres lokalt eller på et testmiljø.

Til produktion bør Maven release plugin bruges, da det får tag'et bygget og automatisk får opdateret versionsnumre m.v.

Først køres:

mvn release:prepare

Går det godt køres:

mvn release:perform

# Tips og tricks

I det følgende beskrives problemer man som udvikler kan støde på, og forslag til løsning af samme.

## Tomcat out of memory

### Beskrivelse

I Tomcats log, logger den noget i stil med ”out of memory” og nævner “permgenspace”, dette kan ske ved at der hot-deployes ofte, som man typisk gør under udvikling

### Løsning

Forøg Tomcat permgen space ved at sætte følgende miljø variabel:

JAVA\_OPTS="-Xms2048m -Xmx2048m -XX:MaxPermSize=512m"

# Ændringslog

Kilden til dette dokument kan findes på:

[https://github.com/trifork/HAIBA-LPRimporter/blob/master/doc/Guide til Udviklere.docx](https://github.com/trifork/HAIBA-LPRimporter/blob/master/doc/Guide%20til%20Udviklere.docx)

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Dato** | **Ændring** | **Ansvarlig** |
| 1.0 | 1/3-2013 | Initielt Dokument | Trifork  Kjeld Froberg |
|  |  |  |  |

# Referencer og kilder

|  |  |  |
| --- | --- | --- |
| Reference-id | Indhold / Overskrift | Henvisning |
| [MAVEN] | Welcome to Apache Maven | <http://maven.apache.org/> |