Stamdataservice på NSP

Guide til udviklere

Indholdsfortegnelse

1 Formål 3

1.1 Metode og rapportens opbygning 3

1.2 Målgruppe og læsevejledning 3

2 System design 4

2.1 Properties 4

3 Opsætning af udviklingsmiljø 5

3.1 Kildekode 5

3.1.1 DoDi 5

3.2 Byggemiljø 5

3.3 Database setup 6

3.3.1 DoDi 6

3.4 Test 6

3.5 IDE 7

3.5.1 Eclipse 7

3.6 IntelliJ Idea IDE 8

3.7 Distribution 9

4 Tips og tricks 10

4.1 JBoss out of memory 10

4.1.1 Beskrivelse 10

4.1.2 Løsning 10

5 Ændringslog 11

6 Referencer og kilder 12

# Formål

Nærværende dokument er en guide til nye udviklere af stamdataservicen på DoDi’en. Guiden gennemgår på overordnet plan de aktiviteter, der er nødvendige for at kunne videreudvikle stamdataservicen.

## Metode og rapportens opbygning

Efter nærværende introduktion vil dokumentet gennemgå de væsentligste dele af opsætningen af et lokalt udviklingsmiljø og afvikling af test/performancetest.

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

I dokumentet benyttes følgende notationer:

Markering af **scripts** og **kommandoer**.

Markering af **advarsler**

Markering af **referencer til filer**.

## Målgruppe og læsevejledning

Den primære målgruppe for dokumentet er systemudviklere.

# System design

Stamdata importerne består af en kerne komponent (sdm4-core), og en række importer specifikke komponenter – alle deployes på Dodi’en og har udelukkende til formål at populere data i en database (kan godt være forskellige databaser for hver komponent), se [DESIGN] for yderligere information vedrørende DoDi.

**Komponenter på DoDi:**

DoDi applikationerne er en række selvstændigt kørende applikationer, til overvågning af indkomne data filer, der kan være i forskellige formater (XML, fastlængde, csv, m.v). Indkomne filer indlæses, valideres og gemmes herefter i DoDi’ens database.

Hvert komponent indeholder en implementation af et Parser interface der fungerer som entry-point til applikationen. Det anbefales, at man som ny udvikler på projektet kigger koden igennem med denne fil som udgangspunkt.

## Properties

*Stamdata* importerne styres af java properties filer. Hver stamdata importer har en default konfigurations fil (default-config.properties) som er deployet sammen med war filen, denne kan overstyres med en properties fil lagt i jboss uden for war filen (config.properties) – se installations guiden for detaljer

# Opsætning af udviklingsmiljø

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

* Java Developer Kit 6.0\_x
* Et passende udviklingsmiljø
* Maven 3.x
* Virtualbox version 4.1.18
* Vagrant version 1.0.3
* MySQL database 5.1.x
* JBoss AS7 (bliver sat op via Puppet scripts)

## Kildekode

Kildekoden er placeret i to forskellige github-repositories:

### DoDi

Kildekoden er placeret her:

Core modul: <https://github.com/trifork/sdm4-core/>

Maven-plugin: <https://github.com/trifork/sdm4-vagrant-maven-plugin/>

Test-utils: <https://github.com/trifork/sdm4-testutils>

Parent-modul inkl. Test-environment: <https://github.com/trifork/sdm4-parent/>

**Importer moduler**

<https://github.com/trifork/sdm4-autorisationimporter/>

<https://github.com/trifork/sdm4-bemyndigelseimporter/>

<https://github.com/trifork/sdm4-cprimporter/>

<https://github.com/trifork/sdm4-sikredeimporter/>

<https://github.com/trifork/sdm4-sksimporter/>

<https://github.com/trifork/sdm4-sorimporter/>

<https://github.com/trifork/sdm4-sorrelationimporter/>

<https://github.com/trifork/sdm4-takstimporter/>

<https://github.com/trifork/sdm4-yderimporter/>

<https://github.com/trifork/sdm4-doseringimporter/>

<https://github.com/trifork/sdm4-vitaminimporter/>

Koden checkes ud på følgende måde:

% git clone [<kode-url>](https://github.com/trifork/sdm2/)

## Byggemiljø

Stamdataprojektet anvender Maven som byggesystem [MAVEN]. Strukturen følger de generelle anbefalinger for Maven projekter, og er struktureret med en parent pom.xml og en projekt pom.xml fil for hvert underprojekt.

Subprojekterne er opbygget efter Maven layout konventionen.

For at bygge en importer, skal man gøre følgende:

Check Core-modul ud og kør mvn install

Check Maven-plugin ud og kør mvn install

CheckTest-utils ud og kør mvn install

Check parent-modul ud og kør mvn install

Herefter er du klar til at bygge et importer-modul.

Importmodulets integrationstest er afhængig af at ligge som et subdirectory til parent-modulet, så du skal checke det ud som sådan et.

Hvis du ønsker at checke alle importermoduler ud, ligger der i roden af parent-modulet et script, checkout-importermodules.sh, som udfører checkout af alle importermoduler samt maven-plugin og test-utils.

## Database setup

### DoDi

DoDi-projekterne anvender Vagrant+Puppet til at opsætte udviklingsmiljøet.

For at få bootet og opsat en udviklingsmaskine, køres følgende

% vagrant up

Fra roden af et importermodul eller fra parent-modulet.

Når vagrant har bygget den virtuelle maskine, er den sat op med den korrekte version af JBoss 7 samt en MySQL der er konfigureret til at have en root-bruger med password papkasse og en sdm4-bruger med password sdm4.

Der er også automatisk oprettet en tom database med standard-navnet for databasen, (sdm\_warehouse) og en datasource i JBoss med det JNDI-navn, som applikationen som default forventer at kunne slå datasource op.

Applikationen kører ved opstart automatisk databaseskema på databasen.

Der er forwardet følgende porte ind i den virtuelle maskine, så man kan tilgå jboss og mysql fra sin lokale maskine

8080 -> 8080 (dvs. at man kan tilgå <http://localhost:8080/>)

3307 -> 3306 (dvs. at man kan have en urelateret mysql kørende lokalt og tilgå SDM-mysql’en på den virtuelle maskine på url localhost:3307)

## Test

Installationen kan verificeres ved at eksekvere stamdataservicens test suite.

Stamdataservicen benytter JUnit og Mockito til test.

Testkoden er for hvert modul lokaliseret i:

src/test/java

Test suiten afvikles ved at udføre følgende kommando i projektroden:

% mvn test

Kommandoen kan også udføres under de individuelle moduler, hvorved kun undermodulets test udføres.

Installationen kan yderligere verificeres ved at udføre kommandoen:

% mvn verify

Denne kommando validerer code coverage og kode konventionerne for projektet.

Kode konventionerne følger reglerne defineret i filen:

config/checkstyle.xml

## IDE

Stamdataservicen 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 stamdataservicen hentes ind i Eclipse:

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 roden og hvert undermodul. 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 stamdataservicen kan derfor direkte importeres. Projektet importeres i IntelliJ ved under ”Create new project” at vælge ”Import project from external model”. Herefter udvælges roden af stamdataservicen, 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

Stamdataservicen kan bygges til distribution eller lokal test ved at udføre:

% mvn package

Dette generer en række WAR filer, der efterfølgende kan deployeres lokalt eller i produktion.

# Tips og tricks

I de følgende beskrives problemer og deres løsninger:

## JBoss out of memory

### Beskrivelse

I JBoss’s boot.log:

$JBOSS\_HOME/server/default/log/boot.log

Logger JBoss noget i stil med ”out of memory” og nævner “permgenspace”

### Løsning

Forøg JBoss permgen space ved at ændre linien indeholdende:

JAVA\_OPTS="-Xms2048m -Xmx2048m -XX:MaxPermSize=**256**m -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"

i filen

$JBOSS\_HOME/bin/run.conf

til

JAVA\_OPTS="-Xms2048m -Xmx2048m -XX:MaxPermSize=**512**m -Dorg.jboss.resolver.warning=true -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"

# Ændringslog

Nyeste udgave af dette dokument kan erhverves ved henvendelse til NSP-operatøren.

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Dato** | **Ændring** | **Ansvarlig** |
| 1.0 | 28/4-2011 | Initielt Dokument | Trifork |
| 1.1 | 6/10-2011 | Opdateret med CPR tjenester | Trifork |
| 1.2 | 8/12-2011 | Kvalitetssikret af Lakeside | Lakeside |
| 1.3 | 22/12-2011 | Opdateret bla. med performance test af autorisationsservicen og kopi-register-servicen | Trifork |
| 1.4 | 20/8-2012 | Tilføjet DoDi-specifikke afsnit der beskriver SDM4 | Trifork  jrf@trifork.com |
| 1.5 | 24/8 2012 | Fjernet al dokumentation der ikke var SDM4 importer specifik | Trifork |

# Referencer og kilder

|  |  |  |
| --- | --- | --- |
| Reference-id | Indhold / Overskrift | Henvisning |
| [MAVEN] | Welcome to Apache Maven | <http://maven.apache.org/> |
| [NIAB] | NSP in a box | Kan rekvireres ved henvendelse til operatøren inklusiv vejledning i anvendelse og konfiguration. |
| [BRS-guide til anvendere] | Guide til anvendere | Ligger i doc bibliotek i en BRS release |
| [BRS-driftvejledning] | Driftvejledning | Ligger i doc bibliotek i en BRS release |