KIH Database

Guide til videreudvikling

1. maj 2013

Indholdsfortegnelse

Indledning

Dette dokument er en vejledning til interesserede, som ønsker at komme i gang med udvikling på KIH Database projektet.

Målgruppen for vejledningen er udviklere med kendskab til Java, Groovy og Grails og til en vis grad GORM/Hibernate. Vejledningen beskriver hvad der er nødvendigt at vide for at komme i gang med (videre-)udvikling af KIH Databasen.

Overordnet består KIH Databasen af en server applikation udviklet i Grails, som udstiller en web applikation og et antal SOAP baserede service endpoints.

Forudsætning til værktøjer

Denne guide antager at følgende værktøjer er installeret i dit udviklingsmiljø:

- Grails 2.x.x (http://grails.org/doc/latest/guide/gettingStarted.html#requirements)
- Git (http://git-scm.com/)
- Apache CXF, matchende den version af cxf plugin, som er installeret.

NB: Der bliver pt. (april 2013) anvendt Grails 2.1.0 i projektet

Download af kode

Tjek kode ud fra GIT-repository'et:

```
$ git clone ssh://git@git.XXXXXXXXX.dk/kih_database.git
```

Nu kan både tests og selve serveren afvikles:

```
$ cd kih_database
```

Kør alle test cases (unit, integration og funktionel):

```
$ grails test-app
```

Eller start serveren:

```
$ grails run-app
```

Introduktion til kode

For at komme i gang med at udvikle på KIH Database-serveren er der visse dele af koden det er en fordel at have kendskab til.

Kodens struktur (Grails MVC pattern)

KIH Database-kodebasen følger Grails' MVC-struktur hvorfor følgende typer af filer, samt specifikke filer, findes som beskrevet i nedenstående tabel.

Tabel 1

Sti	File(r)
`kih_database/grails-app/domain/dk/silverbullet/kihdb/model/`	Domæneklasser anvendt i KIH DB , f.eks. Citizen, LaboratoryReport etc.
`kih_database/grails-app/controllers/`	Login- og logout-controllers
`kih_database/grails-app/controllers/dk/silverbullet/kihdb/`	HomeController og MetaController
`kih_database/grails-app/controllers/dk/silverbullet/kihdb/model/`	Controllerklasser anvendt i KIH DB f.eks. CitizenController, LaboratoryReportController etc.
`kih_database/grails-app/view/`	Views til div. controller-actions.
`kih_database/grails-app/conf/`	Herunder std. Grails-konfigurationsfiler. I KIH Database er specielt følgende filer værd at fremhæve: Bootstrap.groovy Config.groovy Datasource.groovy
`kih_database/grails-app/services/dk/silverbullet/kihdb/services`	Services anvendt af div. controllers
`kih_database/grails-app/endpoints/dk/silverbullet/kihdb/service`	Udstillede SOAP baserede endpoints.
`kih_database/src/java`	Java kildekode. Primært til adskille autogeneret kode fra wsdl2java
`kih_database/src/groovy`	Groovy kildekode, til understøttelse af funktionaliteten.

`kih_database/test/unit`	Placering af unittests
`kih_database/test/integration`	Placering af integrationstests
`kih_database/test/functional`	Placering af funktionelle tests (Geb/Spock)
`kih_database/test/data`	Placering af testdata brugt under test afvikling.

Herudover findes diverse settings i `kih_database/application.properties`.

Forløb gennem koden

Ved visning af indekssiden (umiddelbart efter login) er det i første omgang `HomeController'ens` `index()`-action der afvikles, denne henviser til Borger søgningssiden, som inkluderer `main.gsp`, og herved bliver `main.gsp` også renderet; `main.gsp` genererer header, footer og topmenu.

Grails' håndtering af requests følger dette mønster:

```
request -> controller action -> view GSP -> response
```

En controller kan kalde metoder i services, og et view kan inkludere templates og benytte sig af tag-biblioteker. Controlleren kan desuden foretage redirects, hvilket kortslutter forløbet i ovenstående mønster. Det er værd at bemærke at controllers som standard *ikke* er transaktionelle og handlinger på domæne objekter derfor bør foretages i services.

Udstilling af SOAP baserede services

En af KIH Databasens primære formål er at stille webservice snitflader til rådighed. Pt. er følgende endpoints understøttet:

- Den Gode Kroniker Service v. 1.0 – segment 15.
  - ChronicDatasetService – til at hente borger og telemedicinske målinger
  - MonitoringService – Til at indberette og fjerne målinger
- MonitoringDataset Service v.1.0.1
  - MonitoringDatasetService – til at hente, sende og slette målinger

Services er udstilles igennem CXF plugin version 1.1.0 (1. maj 2013), som wrapper Apache CXF komponenter.

Services er placeret under:

```
kih_database/grails-app/endpoints/dk/silverbullet/kihdb/service
```

Følgende trin gennemføres, når der skal tilføjes et service endpoint:

1. Kode generes fra wsdl vha. `wsdl2java`, som passer til Apache CXF versionen i Grails' CXF plugin:
   a. `$ wsdl2java -fe jaxws21 -encoding UTF-8 -server -impl -p <pakkenavn> MonitoringDatasetService.wsdl`
2. Den genererede kode flyttes til `src/java`.
3. Interface, som er generet fra WSDL, flyttes til `grails-app/endpoints/<pakke>`
4. Implementer WSDL interface
5. Opret og implementer funktionel test i `test/functional`

Sikkerhed for services

De udstillede services bliver sikret vha. af SEAL.java bibliotekerne. Dette gøres vha. Apache CXF interceptor mekanisme. Der er implementeret 2 interceptorer:

- `dk.silverbullet.kihdb.sosi.SosiInInterceptor`
  - Håndterer request i inbound chain
- `dk.silverbullet.kihdb.sosi.SosiOutInterceptor`
  - Håndterer request i outbound chain

De to interceptors er defineret som Spring Beans i `grails-app/conf/spring/resources.groovy`:

```
import dk.silverbullet.kihdb.sosi.SosiInInterceptor
import dk.silverbullet.kihdb.sosi.SosiOutInterceptor

// Place your Spring DSL code here
beans = {
       …
       sosiInInterceptor(SosiInInterceptor)
       sosiOutInterceptor(SosiOutInterceptor)
       …
}
```

Disse interceptors tilføjes til services programmatisk i `BootStrap.groovy` ved:

```
monitoringEndpointFactory.getInInterceptors().add(sosiInInterceptor)
monitoringEndpointFactory.getOutInterceptors().add(sosiOutInterceptor)

chronicDatasetEndpointFactory.getInInterceptors().add(sosiInInterceptor)
chronicDatasetEndpointFactory.getOutInterceptors().add(sosiOutInterceptor)

monitoringDatasetEndpointFactory.getInInterceptors().add(sosiInInterceptor)
monitoringDatasetEndpointFactory.getOutInterceptors().add(sosiOutInterceptor)
```

Opsætning af lokal udviklingsdatabase

DataSource.groovy indeholder beskrivelse af hvilken database der skal anvendes til det pågældende runtime-miljø (test, development, etc.). Vær opmærksom på at konfiguration skrevet i DataSource.groovy (kan) blive overskrevet af konfiguration skrevet i en konfigurationsfil, som er navngivet/placeret i følgende sti:

`C:/kihdatamon/settings/kihdb-config.properties` (Windows) eller
`${userHome}/.kih/kihdb-config.properties` (Linux, OS X)

Et eksempel på en datamon-config.properties-fil ses her:

```
# Common settings
#dataSource.dbCreate = update
#dataSource.pooled = true
##dataSource.logSql = true

# SQL Server DB
#dataSource.driverClassName = net.sourceforge.jtds.jdbc.Driver
#dataSource.username = kihdb
#dataSource.password = kihdb
#dataSource.url = jdbc:jtds:sqlserver://localhost:1433:kihdb
#dataSource.dialect = dk.silverbullet.kihdb.util.SQLServerDialect
#hibernate.default_schema = dbo
#hibernate.default_catalog = kihdb

# H2 based Db
dataSource.driverClassName = org.h2.Driver
dataSource.url=jdbc:h2:devDb;MVCC=TRUE

## MYSQL DB
#dataSource.dialect = dk.silverbullet.kihdb.util.MySQLInnoDBDialect
#dataSource.driverClassName = com.mysql.jdbc.Driver
#dataSource.username = kihdb
#dataSource.password = kihdb
#dataSource.url = jdbc:mysql://localhost:3306/kihdb
```

Stien til konfigurationsfilen er angivet i `Config.groovy`. Findes der ikke en `kihdb-config.properties`-fil, gælder `DataSource.groovy`.

Umiddelbart kan/bør der arbejdes med en af to forskellige databaser til udvikling: Enten en in-memory - eller en filbaseret H2 database (se nedenstående afsnit).

Der er primært én konfigurationsfil som er relevant mht. til opsætning af database: `DataSource.groovy` (antaget der ikke findes en `kihdb-config.properties` fil).

`DataSource.groovy` indeholder den konkrete database-konfiguration for hvert miljø, såsom database-URL etc.

AuditLog plugin

Der henvises til KIH AuditLog plugin dokumentation for detajler. AuditLog plugin inkluderes i løsningen ved at tilføje følgende til filen `BuildConfig.groovy` under `plugins` sektionen:

```
runtime ":kih-auditlog:0.31"
```

For at oversætte tekniske betegnelser for Grails controller og -action til nogle mere læsbare tekster, skal der implementeres en lookup struktur. Auditlog plugin'en forventer at finde en "`auditLogLookupBean`", som implementerer interfacet: `dk.silverbullet.kih.api.auditlog.AuditLogLookup`. "Lookup" klassen indeholder mapningen for den anvendende applikation og konfigureres i projektets Spring opsætning, f.eks. i `resources.groovy`, som:

```
auditLogLookupBean(<egen lookup klasse>)
```

Nedenfor er er vist et eksempel på indholdet i en implementation af `AuditLogLookup`:

```
class KihDbAuditLogLookup implements AuditLogLookup {

    def log = LoggerFactory.getLogger(KihDbAuditLogLookup.class)
    Map lookup = [:]

    def addCitizenText() {
        return [
                "search": "Søg patient"
        ]
    }

    KihDbAuditLogLookup() {
        // Setup controllers ....

        lookup["home"] = [
            "index": "Vis patientsøgning"
        ]

        lookup["logout"] = [
            "logout": "Log ud",
            "index" : "Vis log ind",
            "list" : "Log ud list",
        ]
        lookup["login"] = [
            "auth": "Log ind",
            "authFail": "Fejlet login",
            "denied": "Log ind afvist",

        ]

        lookup["citizen"] = addStandardTexts("patient", "patienter")
```

```
        lookup["citizen"] << addCitizenText()

        lookup["measurementBase"] = addStandardTexts("måling", "målinger")
        lookup["selfMonitoringSample"] = addStandardTexts("måling", "målinger")
        lookup["laboratoryReport"] = addStandardTexts("målepunkt", "målepunkter")
        lookup["labProducer"] = addStandardTexts("måleproducent", "måleproducenter")

        //lookup["iupaccode"] = null

    }

    def addStandardTexts(String singular, String plural) {
        return [
            "create":"Opret ${singular}",
            "delete": "Slet ${singular}",
            "edit": "Rediger ${singular}",
            "index": "Liste af ${plural}",
            "list": "Liste af ${plural}",
            "save": "Gem ${singular}",
            "show": "Vis ${singular}",
            "update": "Opdater ${singular}"
        ]
    }


    @Override
    Map retrieve() {
        return lookup
    }
}
```

Dokumenthistorik

Version	Dato	Initialer	Ændring
1.0	01-05-2013		Initiel version