Guide til FHIR-validering, ændringshåndtering og semantisk søgning med FKI

Dette er en guide til at implementere de forskellige værktøjer, som anvendes i KL’s webinar ” Demo af FHIR-validering, ændringshåndtering og semantisk søgning med FKI”. Guiden henvender sig især til personer med kendskab til softwareudvikling, men som ikke til daglig anvender HL7 FHIR og SNOMED CT. Personer der i forvejen kender til FHIR implementering vil sandsynligvis allerede kende de teknikker og ressourcer der præsenteres her fra den almindelige tekniske dokumentation, men vil kunne se dem appliceret på det kommunale genstandsfelt her.

# FHIR-validering

**Formål:** Her demonstreres, hvordan man kan tjekke at borgerdata udtrykt med FHIR, faktisk lever op til FKI.

Download den seneste version af FHIR validator. Den kan fx findes her, og der er også link til den relevante dokumentation: <http://hl7.org/fhir/validator/>. Læg den et passende sted. I dette eksempel har jeg lagt den i C:/.

Opret en instans du gerne vil validere. Din instans skal overholde FKI implementation guiden(IG). Fx kan du bruge nedenstående FSIII-indsats der vedrører sårbehandling:

<http://build.fhir.org/ig/hl7dk/KL-dk/CarePlan-PressureUlcerIntervention.json>

Gem også json-objektet et passende sted.

FHIR validator er et eksekverbar java-program. Det betyder, at du bare kan køre det fra en kommandopromt, eller powerShell. (hvis du har installeret java). Nu kan instansen valideres ved at køre validatoren med følgende parametre:

* Stien til den instans der skal valideres
* Den implementation guide, der skal valideres op imod evt. inklusiv version. Der skal refereres til den med package-id eller canonical-url. (ikke web base).
* Den profil, der skal valideres op imod repræsenteret med sin canonical-url (som nogen gange, hvis du læser en IG står opført som defining URL)

For sårbehandlings-eksemplet ser det sådan ud i kommandopromten:

C:\>java -jar C:\validator\_cli.jar C:\CarePlan-PressureUlcerIntervention.json -ig dk.fhir.ig.kl.common.caresocial#current -profile <http://kl.dk/fhir/common/caresocial/StructureDefinition/KLCommonCareSocialPlannedIntervention>

Hvilket validerer som det skal. I outputtet skrives blandt andet:

Success: 0 errors, 0 warnings, 1 notes

Information: All OK

Nu kan vi prøve at ændre instansen til noget vi forventer ikke validere. Åbn json-objektet i en almindelig editor udskift "code": "G1.38" med "code": "G1.35". Gem, og kør igen i kommandopromten. Observer at det er en warning der kommer, fordi G1.35 stadig er en valid FSIII indsats-kode, men at validatoren opdager at display-teksten ikke er rigtig:

Display Name for http://kl.dk/fhir/common/caresocial/CodeSystem/FSIII#G1.35 should be one of 'Stomipleje' instead of 'Sårbehandling' for 'http://kl.dk/fhir/common/caresocial/CodeSystem/FSIII#G1.35'

Udskift nu til ”code”: ”I4.4” og "display": "Problemer med tryksår"

Når du kører validatoren igen, fanger den, at man ikke må putte en tilstandskode i en indsatskode-attribut. Koden I4.4 er med andre ord ikke i det ValueSet, der er tilladt for denne attribut.

# Semantisk søgning med FKI

**Formål:** Her demonstreres, hvordan man kan søge i et datasæt bestående af FHIR tilstande, og kun få dem frem, der svarer til et bestemt søgekriterium. Dette kræver at vi generere et stort sæt (ca. 400) aktuelle tilstande med detaljerede problematikker tilknyttet specifikke test-borgere (ca. 30). Disse uploades til en FHIR-server, og vi bruger SNOMED CT’s expression constraint language (ECL), til at udtrække data på enkelte borgere og hele gruppen.

Det næste skridt kræver et java-udviklingsmiljø (IDE og JDK), og et projekt, hvor man kan bruge hapi-fhir. (se evt start-guide i bilag 1). Desuden en kørende SNOMED CT query service, som implementeres som beskrevet her: <https://github.com/IHTSDO/snomed-query-service> (læg mærke til at servicen kræver at du har adgang til SNOMED CT release files, som du kan få adgang til her: <https://mlds.ihtsdotools.org/#/landing/DK>)

**Upload data:**

Download MedComs test-patienter: <https://svn.medcom.dk/svn/drafts/Klassifikationer/TestPatienter.xlsx>, og konverter excel-arket til CSV utf8 i excel. (Hvilket typisk gør ; til separator, derfor antager resten af demoen netop dette). Åbn i en tekst-editor, og slet alle rækker, der ikke er data (fx kolonnenavne og tomme rækker), og tjek evt. at excel har utf8-kodet rigtigt.

Hent de fire filer 119SCT.csv, 140SCTcsv, hjemmeplejeSCT.csv, sygeplejeSCT.csv, fra GitHub og læg dem et passende sted. Det er mapninger mellem FSIII koder og SNOMED CT koder for fire fagområder. Mapningerne kan hentes på kommunernes terminologiserver, men for simplicitetens skyld, vedlægges de her.

Læg java-klassen UploadPatientsAndCondition ind i dit hapi-fhir projekt, ret evt. fhir-serverbase, hvis du vil bruge en anden testserver end hapi-fhir testserveren.

* Ret filstien i main-metoden, til det sted hvor du har lagt testpatient-filen (linje 38)
* Ret filstierne i metoden GivepatientsConditions(), til det sted du har lagt de fire mapningsfiler ( linje 111 og efterfølgende)
* Ret filstien i metoden saveConditionsToFile(), til et sted tilstandene må gemmes (linje 233), og ret det også til i main-metoden (linje 42)

Kør programmet. Programmet skriver links til de patienter, som er uploadet på serveren, hvis det virker rigtigt. Derefter udskriver programmet en (grim) beskrivelse af de tilstande der er genereret. Programmet bruger SNOMED CT til at ”gætte”, hvad nogle borgeres forskellige problemer kunne være, derfor er der lidt blandet dansk og engelsk, og alle tilstandene er ikke nødvendigvis realistiske. Til slut udskrives link til alle tilstande på serveren

**Søg i data:**

Først gennemføres almindelige FHIR søgninger.

Fx kan vi søge efter tilstande, der vedrører §140 træningstilstanden: *0c126894-60e1-4781-96b7-9b227677bfb6 (Vaske sig)*

GET [http://hapi.fhir.org/baseR4/Condition?code=0c126894-60e1-4781-96b7-9b227677bfb6](http://hapi.fhir.org/baseR4/Condition?code=129043005&_include=Condition%3Apatient&_pretty=true)

Eller vi kan søge på hjemmeplejetilstanden *J1.1 (Vaske sig)*

GET [http://hapi.fhir.org/baseR4/Condition?code= J1.1](http://hapi.fhir.org/baseR4/Condition?code=129043005&_include=Condition%3Apatient&_pretty=true)

Vi kan også bruge SNOMED CT, og søge efter alle tilstande, der vedrører borgere der er 129043005 |Afhængig af hjælp ved badning| (her søger vi altså mere detaljeret end tilstanden, hvilket kun er muligt pga SNOMED-kodningen)

GET  [http://hapi.fhir.org/baseR4/Condition?code=129043005](http://hapi.fhir.org/baseR4/Condition?code=129043005%20)

Men hvis vi gerne vil vide, hvor mange borgere der har en tilstand (altså en nedsat funktionsevne) ift. det at vaske sig, så skal vi jo gerne kunne søge på alle FSIII (og FFB) tilstande der vedrører dette på tværs af fagområde. Der er defineret FHIR filterfunktioner, der skulle muliggøre dette, men referenceimplementeringerne implementerer dem ikke altid fx kører følgende ikke:

GET <https://hapi.fhir.org/baseR4/Condition?code:below=365180007>

GET <https://hapi.fhir.org/baseR4/Condition?code:below=http://snomed.info/sct|365180007>

Skal søgningerne foretages server-side kræver det desuden at tilstandene er SNOMED CT kodet – eller at der foreligger et FHIR ConceptMap. I nedenstående benytter vi i stedet FSIII koderne og en lokal mapningstabel, som er trukket fra kommunernes terminologiserver, og SNOMED CT ECL. Eksemplet er FSIII baseret, men kunne lige så godt vedrøre FFB eller blandet FFB og FSIII data.

Åbn og kør ConditionSCTSearch.java. Forudsætningerne er:

* At der er uploadet borgere og tilstande til FHIR serveren
* At du kender ID’et for en af borgerne (i filen er lige nu antaget at de borgerspecifikke søgninger vedrører Patient/2785556 (Bruno), men referencen kan udskiftes)
* At du har placeret mapningsfilen alleSCT.csv et passende sted, og udskiftet filstien i linje 167
* At der er en kørende SNOMED CT query service

Filen kører søgningerne:

* Alle Brunos tilstande
* Alle kognitive problemer (Baseret på: http://localhost:8080/concepts?ecQuery=<<373930000)
* Alle Brunos kognitive problemer
* Alle hukommelsesproblemer (Baseret på: http://localhost:8080/concepts?ecQuery=<<106136008)
* Brunos ADL tilstande (Baseret på: http://localhost:8080/concepts?ecQuery=<<118233009)
* Alle gang- og bevægelsestilstande (Baseret på: [http://localhost:8080/concepts?ecQuery=<<364832000](http://localhost:8080/concepts?ecQuery=%3c%3c364832000))

Helt overordnet virker søgningerne ved:

* Søg alle SNOMED CT koder frem, der opfylder et bestemt ECL søgekriterium
* Find ud af om nogle af SNOMED CT koderne har en korresponderende FSIII kode (findes i mapningsfilen)
* Søg på FSIII koderne én for én, med eller uden en bestemt borger som ekstra søgekriterium.

Det er oplagt at bruge denne typer søgninger i sammenhæng med tilstandsrelationerne, som er KL’s vejledning til søgning indenfor faglige områder

# Ændringshåndtering med FKI

I dette eksempel bruges eksemplet Ejnar.

Hele ændringshåndteringseksemplet ligger i FKI IG’en. Sæt dig fx ind i det ved at lave en tekstsøgning på ”Ejnar” her <http://build.fhir.org/ig/hl7dk/KL-dk/artifacts.html>. Problemet her er at history-delen ikke virker når man ser på eksempler i implementation guiden.

Ejnar-eksemplet omhandler kort fortalt at Ejnar er begyndt at drikke efter sin kones død. Der udredes på et område og registreres en tilstand (der har registreret årsagen ”sorg”). Senere opdateres tilstanden med et fagligt notat, hvor det fremgår at det går bedre. I sidste version af tilstanden afsluttes den. Der hører også en planlagt opfølgning, der opdateres til en udført opfølgning, samt et opfølgningsresultat til eksemplet.

Hele Ejnar eksemplet (som i virkeligheden er om Einer), kan også uploades til en FHIR-server. Så det nemmere at se, hvordan provenance-resourcen i virkeligheden pakker det historiske data. Læg dog mærke til at hapi-fhir baserede testservere ofte ikke bevarer den versionerede reference til de ressourcer den omhandler. Det kan være forvirrende ift. referencerne, men der ud over ligger data på den rigtige måde.

I Einer-eksemplet bruges java-filen PatientConditions.java

Læg java-klassen PatientConditions ind i dit hapi-fhir projekt, ret evt. fhir-serverbase, hvis du vil bruge en anden testserver end hapi-fhir testserveren. Læg i øvrigt mappen ConditionChange et passende sted.

* Ret filstierne i linje 47 og 49 til det sted hvor ConditionChange mappen ligger.

Kør programmet. Programmet udskriver ID-et på Einer.

Indsæt Einers id i nedenstående GET statement, og kør på serveren.

GET [http://hapi.fhir.org/baseR4/Patient?\_id=\*\*\*\*\*\*\*&\_revinclude=\*&\_pretty=true](http://hapi.fhir.org/baseR4/Patient?_id=*******&_revinclude=*&_pretty=true)

Navigér rundt i eksemplet på serveren.

# Bilag 1: Hapi-FHIR startguide

1. Prerequisites:

An IDE. For example Netbeans or Intellij. For netbeans: Make sure that “Gradle support” is working. Java version: I have a JDK13, and it works fine. At some point we had problems with JDK 9, and recommended JDK8

1. Start new single gradle project, go to the build.gradle and add the following Gradle dependencies:

// https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-base

implementation group: 'ca.uhn.hapi.fhir', name: 'hapi-fhir-base', version: '5.1.0'

// https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-structures-r4

implementation group: 'ca.uhn.hapi.fhir', name: 'hapi-fhir-structures-r4', version: '5.1.0'

// https://mvnrepository.com/artifact/ca.uhn.hapi.fhir/hapi-fhir-client

implementation group: 'ca.uhn.hapi.fhir', name: 'hapi-fhir-client', version: '5.1.0'

// https://mvnrepository.com/artifact/org.slf4j/slf4j-simple

implementation group: 'org.slf4j', name: 'slf4j-simple', version: '2.0.0-alpha1'

More repositories and possibly newer version can be found at: <https://mvnrepository.com>

1. Now you can make your first hapi FHIR application. Make a new java-class and run the following main class. It makes a search for patients named Chuck

import ca.uhn.fhir.context.FhirContext;

import ca.uhn.fhir.rest.client.api.IGenericClient;

import java.io.IOException;

import org.hl7.fhir.r4.model.Bundle;

import org.hl7.fhir.r4.model.Patient;

public class SearchForChuck {

public static void main(String[] args) throws IOException {

// We're connecting to a R4 compliant server in this example

FhirContext ctx = FhirContext.forR4();

String serverBase = "http://hapi.fhir.org/baseR4";

IGenericClient client = ctx.newRestfulGenericClient(serverBase);

// Perform a search

Bundle results = client

.search()

.forResource(Patient.class)

.where(Patient.FAMILY.matches().value("Chuck"))

.returnBundle(org.hl7.fhir.r4.model.Bundle.class)

.execute();

System.out.println("Found " + results.getEntry().size() + " patients named 'Chuck'");

}

}

More examples and explanations: <https://hapifhir.io/hapi-fhir/docs/client/generic_client.html>