

Fhir On Pi - 6



In de [vorige aflevering](#) hebben we Maven geïnstalleerd en ben ik vrij lang ingegaan op de IHE organisatie en de moeilijkheden die je tegenkomt als je computersystemen niet alleen met elkaar wilt **verbinden**, maar er ook voor wil zorgen dat informatie met behoud van hun betekenis van het ene systeem naar het andere kan worden overgedragen.

Juist in de zorg is dat extra complex omdat de verschillende sectoren in de zorg en de verschillende specialismen allemaal een eigen “vaktaal” hebben, waardoor een bepaald gegeven bij de ene zorgverlener nét iets anders kan betekenen dan bij de andere. Het alleen maar doorgeven van “data” is daarom onvoldoende; het maakt nogal uit of je iemands hartslag opneemt in rust of na een fietsproef (de *context* van de data), “taaislijmziekte” wordt in de zorg ook aangeduid met CF of Cystic Fibrosis (*synoniemen*), de “ziekte van Paget” kan betrekking hebben op een kwaadaardige tumor in de borst, maar het is ook de naam van een chronische botziekte (*eponiemen*). Zolang informatie wordt uitgewisseld tussen mensen in dezelfde omgeving (hetzelfde ziekenhuis, dezelfde afdeling van de GGZ, hetzelfde specialisme) en de uitwisseling tussen mensen plaatsvindt (die bijvoorbeeld op non-verbale reacties kunnen reageren) zal de verwarring beperkt blijven (al is gebleken dat -bijvoorbeeld bij overdrachtssituaties veel informatie verloren gaat). Maar als er informatie tussen verschillende zorginstellingen (of zelfs met “buiten”) wordt uitgewisseld tussen IT-systemen dan mag er over de betekenis van de gegevens geen enkele onduidelijkheid bestaan.

“**Semantische interoperabiliteit**” is dan een vereiste, zeker als we zorggerelateerde informatie tussen ICT-systemen gaan uitwisselen. Standaarden als HL7, HL7 Fhir en SNOMED daarbij een belangrijke rol. De “word cloud” in de afbeelding is gemaakt door deelnemers aan het webinar “[Eenheid van Taal en SNOMED, een introductie](#)”, in juni 2020 gegeven door Nictiz en geeft aan waaraan zij dachten bij “eenheid van taal”.



Voor nu genoeg daarover. We gaan nu (eindelijk) de **HAPI Fhirservers** installeren. In de vorige aflevering hebben we als laatste git geïnstalleerd en we laten nu git op onze Raspberry het materiaal ophalen uit github.

We maken eerst een aparte directory om de download in neer te zetten. Zorg dat je in de “home” directory van de gebruiker pi staat:

```
cd /home/pi (als gebruiker pi kun je ook cd ~ gebruiken (tilde))
```

Maak de directory “hapi” aan onder /home/pi

```
mkdir hapi (als gebruiker pi is “sudo” hier niet nodig)
```

Verplaats je naar de nieuw aangemaakte subdirectory:

```
cd hapi
```

Als je wilt weten of het gelukt is, dan geef je de opdracht:

```
pwd
```

en als het goed is reageert de Pi dan met: /home/pi/hapi

Dan nu het ophalen (“klonen”) zelf:

```
git clone https://github.com/hapifhir/hapi-fhir-jpaserver-starter
```

Er gebeurt nu van alles op je Pi. Je ziet een zin “cloning into hapi-fhir-jpastarter...” en er worden een boel “objects” opgehaald. Als de Pi daarmee klaar is moet je deze code nog even uitchecken:

```
cd hapi-fhir-jpaserver-starter
```

```
git checkout
```

Goed. Je hebt nu alles op de Pi staan wat nodig is om de HAPI Fhir-server op te bouwen.

Nightly builds en Continuous Integration

Voordat we de HAPI-Fhirserver gaan “bouwen” moet ik even iets uitleggen, want in aflevering 5 heb ik wat uitgelegd over “Make”, “Apache ANT” en “Apache Maven”. Ik legde uit dat Maven niet alleen de bouw van een applicatie stuurt, maar ook het testen ervan. Het voordeel is dat je op die manier aan “nightly builds” en “continuous integration” kunt doen: ontwikkelaars schrijven overdag code, slaan die op in een repository (bijvoorbeeld Git) en via Maven worden alle delen (gemaakt door verschillende ontwikkelaars, vaak in verschillende organisaties) samengevoegd en tot een toepassing gemaakt. Maven kan daarbij ook unit-tests (tests die één component testen) en integration tests (tests die kijken of verschillende componenten goed samenwerken) aansturen en uiteindelijk zelfs systeemtests (tests van het geheel).

Dit is ook zo bij de HAPI Fhir-server die je zonet naar je Pi hebt gedownload. Nu zijn er een paar probleempjes bij al die tests, als we ze gaan uitvoeren op de Pi, namelijk:

1. Ze duren erg lang
2. Ze hebben een heleboel aanvullende programmatuur nodig (want de tools die het testen uitvoeren zijn zélf ook weer java-programma's)
3. Eén of meer van die testtools werken niet goed op de Pi. Bij mij "faalt" onder meer een test die met Elasticsearch wordt uitgevoerd en dan breekt ook de compilatie af.

We bouwen daarom de HAPI Fhir-server op zónder al die testen, dat blijkt in de praktijk wél te werken op de Pi.

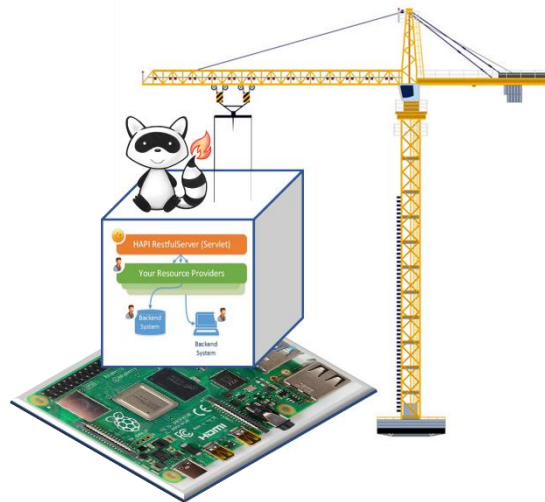
We gaan (eindelijk) bouwen!

De opdracht hiervoor is:

mvn clean install -DskipTests

(Vanuit de directory /home/pi/hapi-fhir-jpaserver-starter)

Vanaf het moment dat je hierna op <Enter> drukt wordt het druk in je scherm. Maven haalt van alles en nog wat op, compileert het, haalt weer meer op, enzovoorts....



Je zult af en toe een "Warning" zien dat het javac-pad niet gevonden kan worden, dat geeft niet, Maven vindt heus waar bij jou de Java-omgeving staat. Ook andere "warnings" (bijvoorbeeld voor "duplicate names") leverden bij mij geen grote problemen op.

```

p@Pi3bplus: ~/hapi-fhir-jpaserver-starter
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 19 source files to /home/pi/hapi/hapi-fhir-jpaserver-starter/target/classes
[WARNING] Unable to autodetect 'javac' path, using 'javac' from the environment.
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ hapi-fhir-jpaserver-starter ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ hapi-fhir-jpaserver-starter ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 8 source files to /home/pi/hapi/hapi-fhir-jpaserver-starter/target/test-classes
[WARNING] Unable to autodetect 'javac' path, using 'javac' from the environment.
[INFO] --- maven-surefire-plugin:3.0.0-M3:test (default-test) @ hapi-fhir-jpaserver-starter ---
[INFO] Tests are skipped.
[INFO] --- maven-war-plugin:3.2.3:war (default-war) @ hapi-fhir-jpaserver-starter ---
[INFO] Packaging webapp
[INFO] Assembling webapp [hapi-fhir-jpaserver-starter] in [/home/pi/hapi/hapi-fhir-jpaserver-starter/target/ROOT]
[INFO] Processing war project
[INFO] Copying webapp resources [/home/pi/hapi/hapi-fhir-jpaserver-starter/src/main/webapp]

```

Verder geeft Maven veel "Information" meldingen, onder meer dat tests worden overgeslagen. Je ziet ook dat er een ongelofelijke hoeveelheid "source files" wordt opgehaald. Het is een groot project! (Maar volgens mij worden ook de "test classes" opgehaald, al gebruiken we die niet).

Bij mij duurde het bouwen tussen de twee en drie minuten (als je vaker "bouwt" maakt Maven gebruik van een repository van componenten die de eerste keer al goed waren).

Ondertussen kan ik mooi iets uitleggen over de HAPI Fhir-server waar de Pi het nu zo druk mee heeft. Er zijn twee onderwerpen die wel even toegelicht moeten worden:

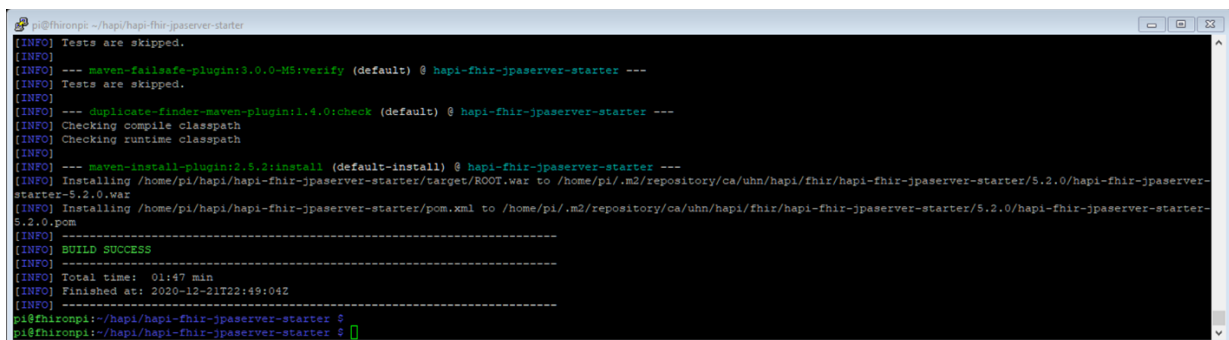
- de gegevensopslag (“database”);
- de testclient

Gegevensopslag. De Fhir-server moet natuurlijk ergens zijn gegevens in opslaan, zodat bijvoorbeeld een patiënt die je vandaag toevoegt, morgen weer kan worden opgevraagd. In meer professionele omgevingen zal je voor die opslag een “echte” database gebruiken, bijvoorbeeld MySQL, Postgresql of een ander. Dat kan met HAPI, maar vraagt wat extra configuratie en -natuurlijk- de installatie van óók die database. Voor beperktere toepassing beschikt de HAPI Fhirserver over een “H2 database engine”. Dit is een relationele database die geheel in Java geschreven is en nu, tegelijk met HAPI, wordt opgebouwd. Tegen déze database vinden namelijk de tests plaats (die we nu niet uitvoeren). Die H2 database vereist geen extra installaties, al moeten we wel even iets voorbereiden omdat we het systeem anders niet draaiend krijgen.

De Testclient. Strikt genomen accepteert de HAPI Fhir-server alleen Fhir-requests (opdrachten geschreven in JSON formaat volgens de FHIR-structuren) en antwoordt de server alleen in Fhir-responses. Als mens is dat allemaal wat lastig te interpreteren en daarom zit bij de HAPI Fhir-server ook een “testclient”, een vrij simpele webpagina waarmee je de Fhir-server kunt benaderen. Die “testclient” is voor ons veel beter te begrijpen, al zit de Fhir-communicatie daar vrij dicht onder.

De “testclient” is géén EPD of Labsysteem, maar helpt wel om bijvoorbeeld te zien hoeveel patienten er in de database zitten en om de (in Fhir-structuur weergegeven) informatie per patient te bekijken.

BUILD SUCCES. Goed, als de Pi klaar is met het bouwen van het geheel dan zie die verlossende opmerking en een indicatie van de tijd die ervoor nodig was.



```

pi@thronpi: ~/hapi/hapi-fhir-jpaserver-starter
[INFO] Tests are skipped.
[INFO] --- maven-failsafe-plugin:3.0.0-M5:verify (default) @ hapi-fhir-jpaserver-starter ---
[INFO] Tests are skipped.
[INFO] --- duplicate-finder-maven-plugin:1.4.0:check (default) @ hapi-fhir-jpaserver-starter ---
[INFO] Checking compile classpath
[INFO] Checking runtime classpath
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ hapi-fhir-jpaserver-starter ---
[INFO] Installing /home/pi/hapi/hapi-fhir-jpaserver-starter/target/ROOT.war to /home/pi/.m2/repository/ca/uhn/hapi/fhir/hapi-fhir-jpaserver-starter/5.2.0/hapi-fhir-jpaserver-starter-5.2.0.war
[INFO] Installing /home/pi/hapi/hapi-fhir-jpaserver-starter/pom.xml to /home/pi/.m2/repository/ca/uhn/hapi/fhir/hapi-fhir-jpaserver-starter/5.2.0/hapi-fhir-jpaserver-starter-5.2.0.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:47 min
[INFO] Finished at: 2020-12-21T22:49:04Z
[INFO] -----
pi@thronpi:~/hapi/hapi-fhir-jpaserver-starter $
pi@thronpi:~/hapi/hapi-fhir-jpaserver-starter $

```

Nu moeten we nog een paar kleine dingen doen.

User Tomcat. Bij het aanmaken van de user “tomcat” in aflevering 3 heb ik een fout gemaakt. Die fout leidde ertoe dat “tomcat” niet kon inloggen, en daarmee ook geen directories kon aanmaken voor de H2 database. Dat gaan we nu rechtzetten. We stoppen eerst het tomcat-proces:

sudo systemctl stop tomcat

We verwijderen daarna de user tomcat:

```
sudo userdel tomcat
```

dan verwijderen we ook de groep die we in die aflevering hebben aangemaakt:

```
sudo groupdel tomcat
```

Nu maken we de user tomcat opnieuw aan, maar nu wel op de goede manier:

```
sudo adduser tomcat
```

Het adduser-script zet nu een aantal dingen klaar voor tomcat en vraagt dan om het wachtwoord bij dit account.

New password: <vul hier iets in en onthoudt het wachtwoord!>

Retype new password: < vul hier hetzelfde in>

Het script vraagt nog een paar dingen, als voor- en achternaam en kamernummer maar die mag je ook leeg laten.

Ruimte voor de database. Deze versie van de HAPI Fhir-server wil ruimte hebben voor de H2-database direct onder de “root” directory (de topdirectory waar alle andere directories onder hangen) en wel in de map /target . Dat is bijzonder (of misschien wel een foutje) want normale gebruikers mogen in Unix-systemen geen mappen onder root aanmaken, dat moeten wij dus doen voor tomcat:

```
sudo mkdir /target
```

```
sudo mkdir /target/database
```

Nu zorgen dat tomcat hier bij kan:

```
sudo chown tomcat:tomcat /target
```

Okee. Nu kunnen we het resultaat van het bouwen van daarnet kopieëren naar de tomcat webapp omgeving:

```
sudo cp /home/pi/hapi/hapi-fhir-jpaserver-starter/target/ROOT.war /opt/tomcat/webapps/ROOT.war
```

De file ROOT.war is een “web-archive” waarin de hele HAPI server gecomprimeerd opgeslagen is, het is een groot bestand, dus de kopie-slag duurt even. Nu nog zorgen dat Tomcat iets met dit bestand kan doen:

```
sudo chown tomcat:tomcat /opt/tomcat/webapps/ROOT.war
```

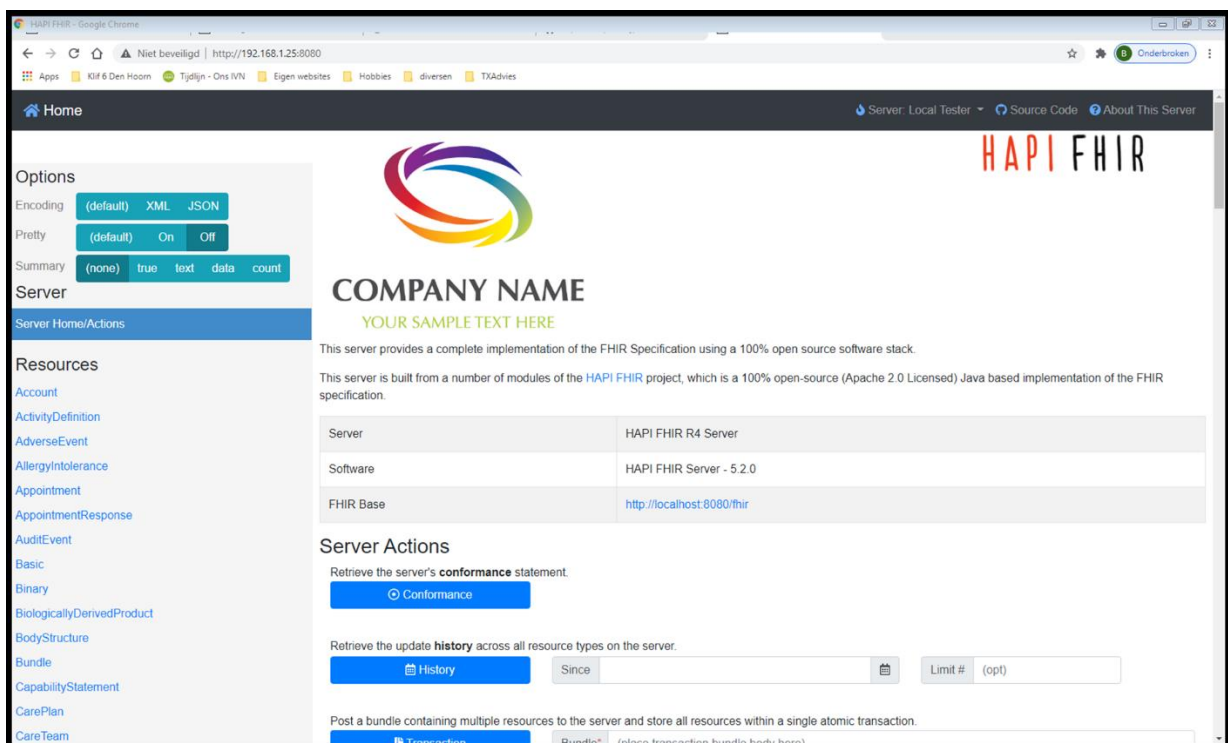
Tomcat wil straks dit “archive” uitpakken in de directory /opt/tomcat/webapps/ROOT maar die directory is er al (aangemaakt bij de Tomcat-installatie). De bestaande directory hernoemen we daarom even:

```
sudo mv /opt/tomcat/webapps/ROOT /opt/tomcat/webapps/ROOT_OLD
```

Nou, dat was echt alles. We starten Tomcat weer op; het opstarten kan nu iets langer duren want Tomcat gaat ook de HAPI-server uitpakken en een database(je) opbouwen.

```
sudo systemctl start tomcat
```

Start nu je webbrowser weer en ga naar <IP-adres van je Raspberry>:8080. Op deze plek stond tot nu toe welkomspagina van tomcat, maar die heb je nu verplaatst (naar ROOT_OLD) en in plaats daarvan zie je de testpagina van de HAPI Fhir-server.



Aan de linkerkant van de testpagina zie je de Fhir-“resources” waarvoor plek is in je database. Kijk gerust rond op de pagina, druk op buttons, je kunt niks stukmaken.

Kijk ook even naar de rechterbovenkant in de grijze balk zie je onder andere de tekst “Server: Local Tester”. Je kunt daar ook kiezen voor “Server: Global Tester” en dan koppel je de testclient aan de open (test-)server <http://hapi.fhir.org/baseR4> . Opeens heb je duizenden patienten, “encounters” en “observations” ter beschikking.

In de volgende (en voorlopig laatste) aflevering ga ik in op deze testpagina en een klein beetje dieper of Fhir. En we gaan ook zonder de testpagina tegen onze fonkelnieuwe HAPI Fhirserver aanpraten.

Tot de volgende!