Cupcake-Fhir Cancer Reporting Server Manual

Project Name: Cancer Reporting from Physician EHRS to State Cancer Registries

Team Name: Cupcake

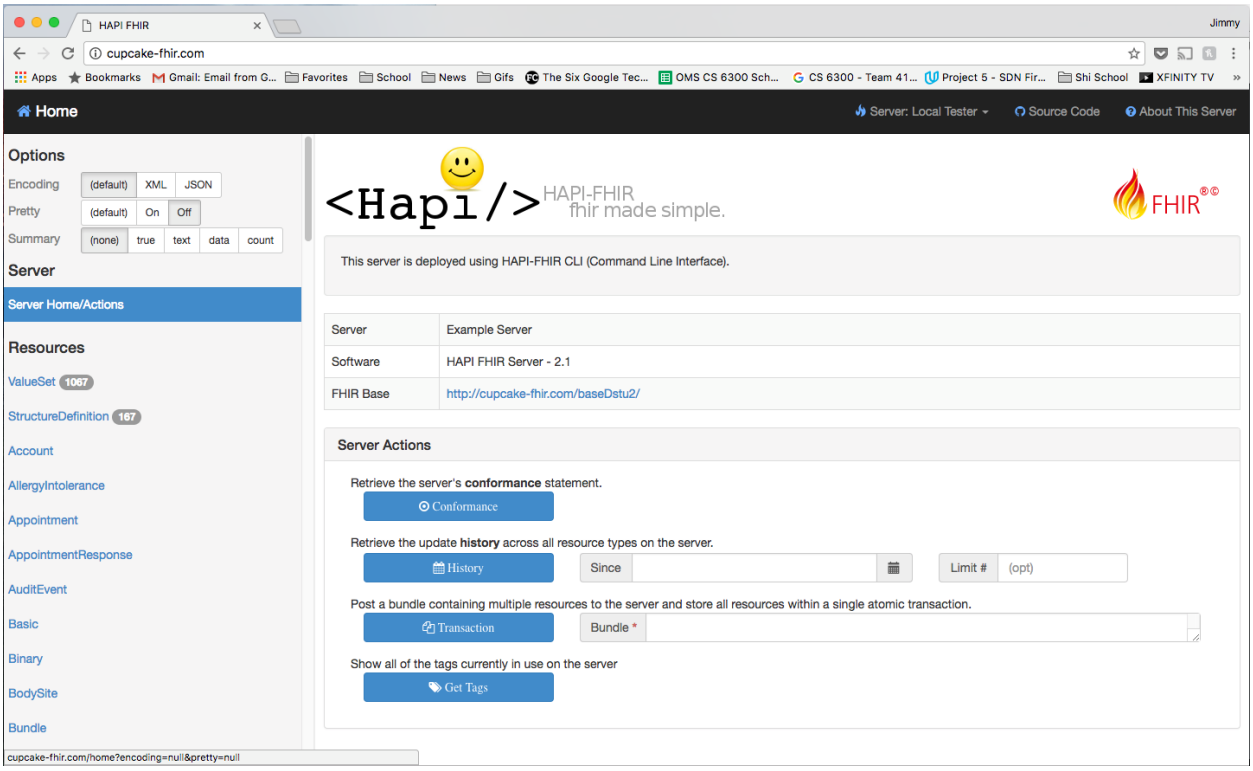
Team Members:

Name	Role
Subhadeep Nag	Project Manger
Jim Ecker	Software Developer
Andrew Frieze	Software Developer

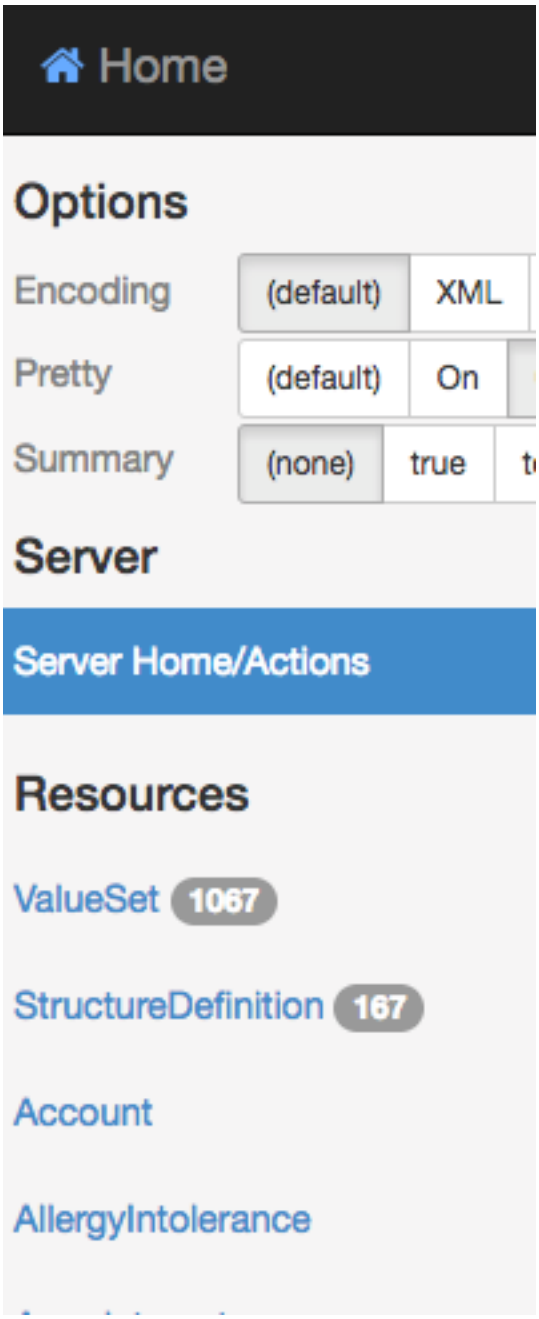
Github URL: https://github.gatech.edu/gt-hit-fall2016/Cancer-Laboratory-Reporting-Team-Cupcake

The Cupcake-Fhir Cancer Reporting Server is a proof of concept server to help facilitate efforts to move cancer diagnosis reporting from local practitioners to state cancer registries. As such, it contains custom profiles to bring CDA based cancer diagnosis transmissions into conformance with the FHIR standard.

The live demo server is located at cupcake-fhir.com. There, you can use the Hapi-fhir server front-end to explore the server's structure.



The server's primary role is for use as a conformance server since it is bringing cancer diagnosis transmission into conformance with FHIR standards. It also serves as a terminology server, providing the ValueSets needed.



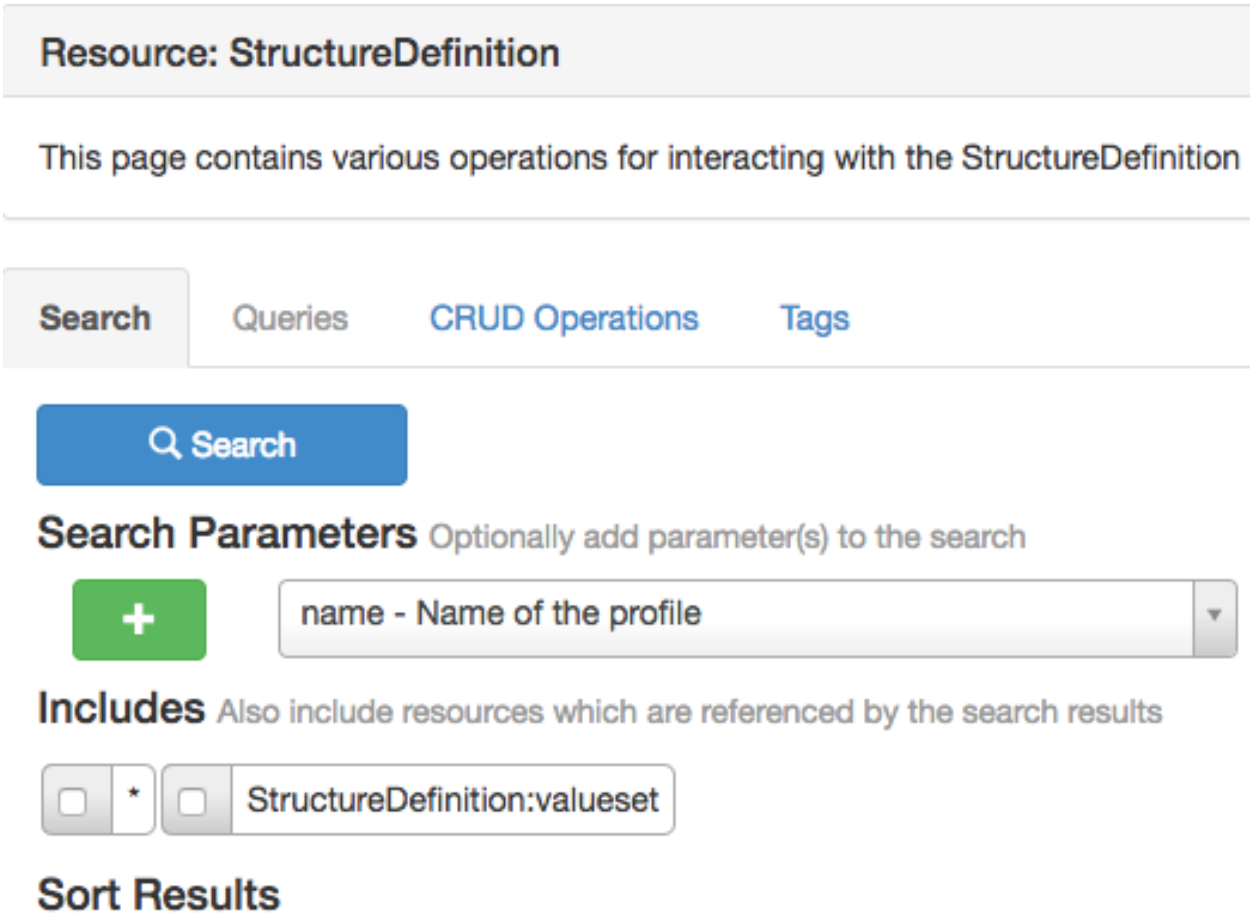
The server comes with custom profiles rolled out for cancer diagnosis transmission.

- CancerPatient
- CancerPractitioner
- CancerDiagnosis

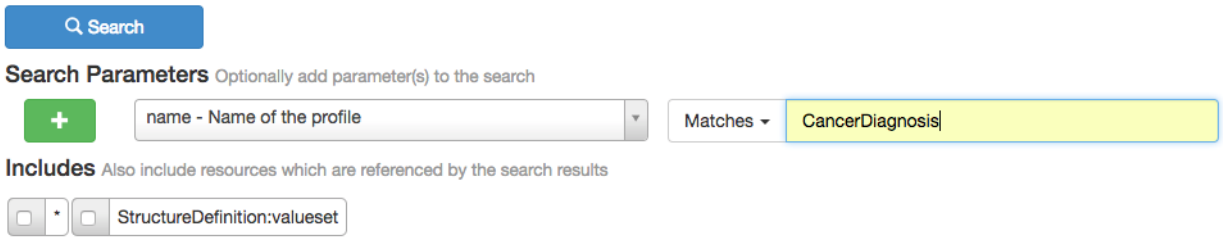
Each of these profiles provides an extension on a FHIR base class, constraining it to facilitate the information needed by the state cancer registries to which the diagnosis is being sent.

Starting from cupcake-fhir.com's Home page, you can view the structure of these profiles by clicking on StructureDefinition

and then selecting "name" under Search Parameters



and entering the name of the profile you wish to inspect in the “Matches” field to the right



Finally, clicking the “Search” button will bring up a JSON response from the server showing all the StructureDefinitions matching your query

Client Code - Use the following code snippet to execute this action in your own client.

```
// Create a client (only needed once)
FhirContext ctx = new FhirContext();
IGenericClient client = ctx.newRestfulGenericClient("http://cupcake-fhir.com

// Invoke the client
Bundle bundle = client.search().forResource(StructureDefinition.class)
   .where(new StringClientParam("name").matches().value("CancerDiagnosis"))
   .prettyPrint()
   .execute();
```

Request	GET http://cupcake-fhir.com/baseDstu2/StructureDefinition?r
Request Headers	Accept-Charset: utf-8 Accept: application/xml+fhir;q=1.0, application/json+fhir User-Agent: HAPI-FHIR/2.1 (FHIR Client; FHIR 1.0.2/DSTU2; Accept-Encoding: gzip

Response	✔ HTTP/1.1 200 OK
Response Headers	Date: Sun, 04 Dec 2016 20:46:28 GMT X-Powered-By: HAPI FHIR 2.1 REST Server (FHIR Server; FHIR 1. Last-Modified: Sun, 04 Dec 2016 20:46:28 GMT Content-Type: application/json+fhir;charset=utf-8 Transfer-Encoding: chunked Server: Jetty(9.3.z-SNAPSHOT)
Result Body JSON bundle (3224 bytes)	⊖ Bundle contains 1 / 1 entries ID Read Update StructureDefinition/CancerDiagnosis Raw Message


```
{
  "resourceType": "Bundle",
  "id": "ae055617-a80b-4019-9a20-68a1e15159d0",
  "meta": {
    "lastUpdated": "2016-12-04T20:46:28.141+00:00"
  },
  "type": "searchset",
  "total": 1,
  "link": [
    {
      "relation": "self",
      "url": "http://cupcake-fhir.com/baseDstu2/StructureDefi
```

This can be done for any of the other custom profiles (CancerPatient and CancerPractitioner).

Deploying your own copy of the server

We have included install scripts for users who would like to create their own local instance of the server. It is recommended that your server be running Linux with the Bash shell, have at least 2GB of ram and 40GB of storage, and running Java 8.

With a server not running anything on port 8080, go to the Cupcake-Fhir install directory

In Final Project/Final Application/Application:

```
bash install-server.sh
```

This script will download the Hapi-CLI server into your system's /opt directory and provide scripts for running the server and installing the profiles located in the ./Profiles directory

After running install-server.sh your install directory should look like this

- Profiles/
  - CancerPatient.structuredefinition.xml
  - CancerPractitioner.structuredefinition.xml
  - CancerDiagnosis.structuredefinition.xml
- install-server.sh
- run-server.sh
- install-profiles.sh

running

```
bash run-server.sh
```

at this point will load the server and run it using nohup as a background process. After a minute or two, you can point a browser at <<server ip>>:8080 and see the server is up and running.

Once you have verified that the server is up and running, you can run

```
bash install-profiles.sh
```

which will install the custom profiles to the server. **Note: This must be done while the server is running.**

The server uses an instance of the Derby database to hold data for storage. You will see a derby.log file in the install directory for the server after you run it. You should also see a target/

folder and a nohup.output file for the nohup log. This can be used to debug any problems you have with using the server.