Working with FHIR Terminology Services – From a Coder's Perspective

Joshua Wiedekopf, M.Sc.

IT Center for Clinical Research | Institute of Medical Informatics

Universität zu Lübeck | Universitätsklinikum Schleswig-Holstein

Workshop FHIR Terminology Services 2022-05-10

What I'm Going To Talk About Today

- Interactive creation with Snapper is not always sufficient, and resources want to be consumed

- Accessing FHIR APIs is easy and powerful

- We will do the following today:
  - List resources on the server
  - Validate that a code is contained by a CodeSystem/ValueSet
  - List all codes in a CodeSystem/ValueSet for selection by the user
  - (Expansion of SNOMED CT Expression Constraint Language expressions)
  - Conversion of a table in a relational DB to FHIR CodeSystem & Mapping to LOINC

- Demonstration of common patterns implemented in Python
  - Some implementations also in Java/Spring Boot, for demonstrating usage patterns

Convention

- https://github.com/itcr-uni-luebeck/fhir-term-samples

- Python Dependencies (via `python -m venv`):
  - `fhir.resources`
  - `requests`
  - `rich, questionary`

- Java Dependencies (via Gradle):
  - `ca.uhn.hapi.fhir:{hapi-fhir-structures-r4, hapi-fhir-client}`
    - `org.apache.httpcomponents:httpcomponents-client`
  - `org.projectlombok:lombok`
  - Spring Boot

- **Cave: You may need to URL-encode request parameters!**

Authentication

- Provide DFN certificate for every request (unless on IP allow list)

- "Workaround" – Use your reverse proxy to handle authentication transparently!
  - also great for adding a cache like Varnish in front of the SSL termination for speed and reliability…

```
location /onto/ {
    proxy_pass               https://terminology-highmed.medic.medfak.uni-koeln.de:443/;
    proxy_ssl_certificate        /etc/pki/ontoserver/certs/cert-with-chain.pem;
    proxy_ssl_certificate_key    /etc/pki/ontoserver/certs/private.pem;
    proxy_ssl_protocols          TLSv1.2 TLSv1.3;
    proxy_ssl_ciphers            HIGH:!aNULL:!MD5;
    proxy_ssl_trusted_certificate  /etc/pki/ontoserver/chain/chain.pem;
    proxy_ssl_verify             on;
    proxy_ssl_verify_depth       5;
}
```

Authentication in Python

```python
import requests
from fhir.resources.capabilitystatement import CapabilityStatement

endpoint = "https://terminology-highmed.medic.medfak.uni-koeln.de/fhir/metadata"
sess = requests.Session() # create a persistent session/connection
cert_file = "../joshua_dfn.pem" # contains both certificate and private key
#cert_file = ("../joshua_dfn.crt", "../joshua_dfn.key") # two files for public/private key
sess.cert = cert_file # applies to all request initiated from the session
response = sess.get(endpoint) # issue the actual request (without preparing it)
conformance = CapabilityStatement(**response.json()) # parse as FHIR
software = conformance.software # access structure of FHIR resource
print(f"{software.name} version {software.version}")

############
# Ontoserver® version 6.5.0
############
```
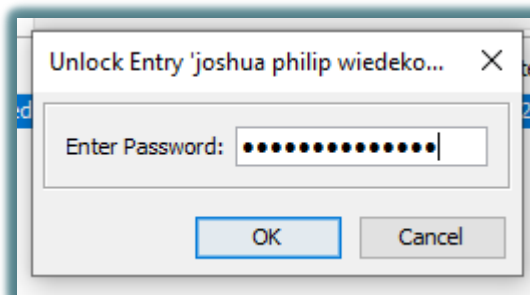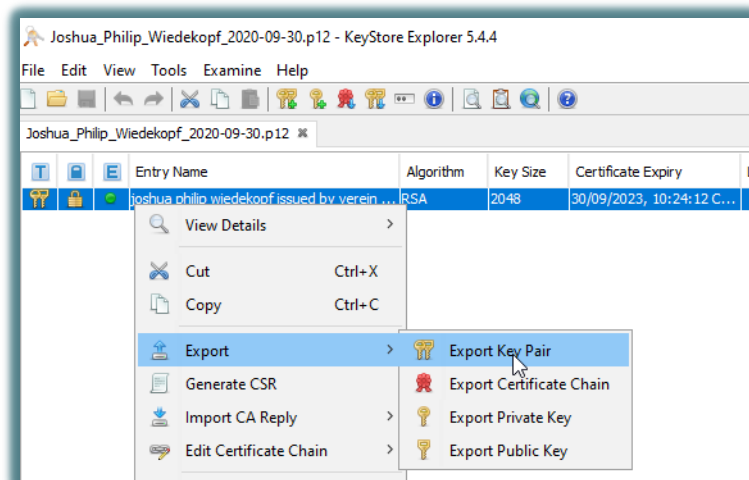
Python: Key Material

- Python expects OpenSSL/PEM format for key material

- Convert from PKCS#12 using OpenSSL

```
openssl pkcs12 -in path.p12 -out cert_with_key.pem -nodes
```
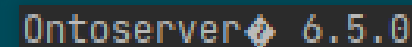
- Or use Keystore Explorer
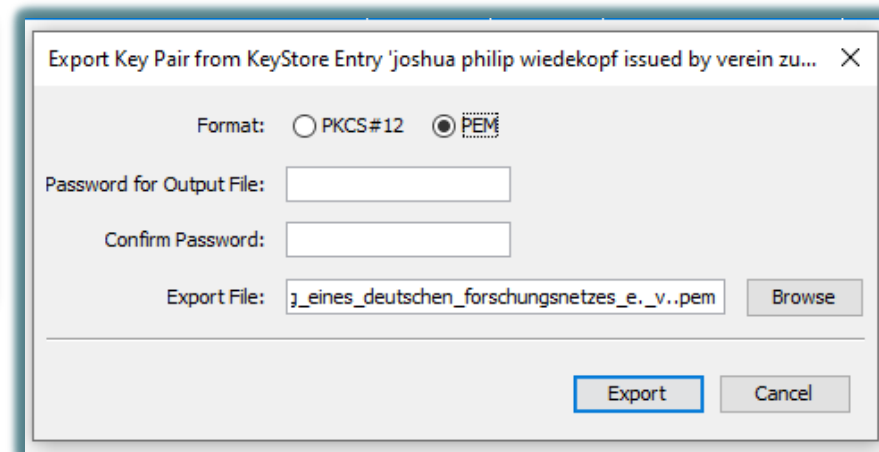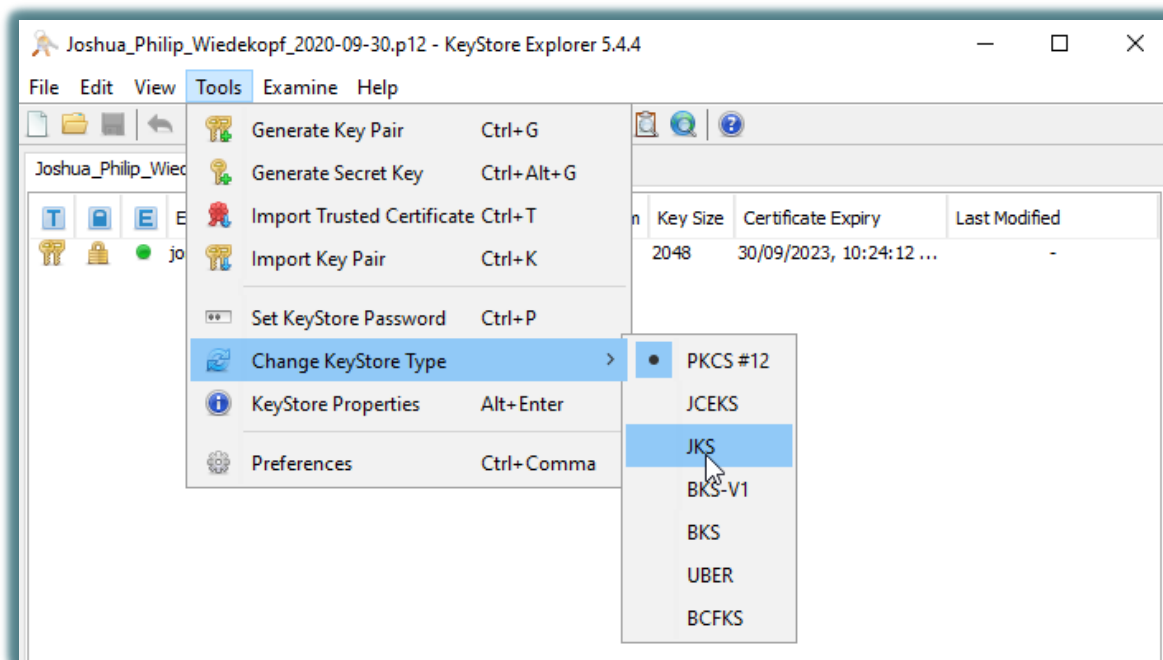
Authentication in Java

```java
URL keystorePath = ClassLoader.getSystemResource("keystore.jks"); //get keystore resource
char[] password = "pw".toCharArray(); //password of keystore and key (could as well be different)
try {
    SSLContext sslContext = SSLContexts.custom() //load key material, trust comes from system default
        .loadKeyMaterial(keystorePath, password, password).build();
    CloseableHttpClient httpClient = HttpClients.custom().setSSLContext(sslContext).build();
    final FhirContext fhirContext = FhirContext.forR4(); //do this exactly once during app lifecycle
    final IRestfulClientFactory restfulClientFactory = fhirContext.getRestfulClientFactory();
    restfulClientFactory.setHttpClient(httpClient); //all new clients will use custom HTTP Client

    final String serverBase = "https://terminology-highmed.medic.medfak.uni-koeln.de/fhir";
    IGenericClient fhirClient = fhirContext.newRestfulGenericClient(serverBase);
    CapabilityStatement capabilities = fhirClient.capabilities().ofType(CapabilityStatement.class)
        .execute(); //query the metadata and parse as CapabilityStatement (could be subclass)
    System.out.printf("%s %s%n", capabilities.getSoftware().getName(),
        capabilities.getSoftware().getVersion()); //access the nodes of the parsed capability statement
} catch (Exception e) { //… }
```

Ontoserver� 6.5.0

BUILD SUCCESSFUL in 13s

Java: Key Material

- Java works using *keystores & truststores*

- Using PKCS#12 is possible, or convert to JKS
  - Use [Keystore Explorer](#)

Spring Boot: Authentication – REST Template Bean

```java
@SpringBootApplication public class Application {
  @Bean @SneakyThrows public RestTemplate fhirRestTemplate(FhirSslProps fhirSslProps,
  @Value("classpath:keystore.jks") Resource keystoreResource) { //inject resource/settings
    SSLContext sslContext = SSLContexts.custom().loadKeyMaterial(keystoreResource.getURL(),
      fhirSslProps.getKeystorePassword().toCharArray(),
      fhirSslProps.getKeyPassword().toCharArray()).build(); //load material from resource
    CloseableHttpClient httpClient = HttpClients.custom().setSSLContext(sslContext).build();
    ClientHttpRequestFactory clientRequestFactory =
      new HttpComponentsClientHttpRequestFactory(httpClient); //create a new request factory
    return new RestTemplate(clientRequestFactory); //and wrap it as a template
  }
  @Bean public FhirContext fhirContext() { return FhirContext.forR4(); } //create once, use often
  public static void main(String[] args) { SpringApplication.run(Application.class, args); }
}


@Data @ConfigurationProperties(prefix = "fhir.ssl") class FhirSslProps { //holder for SSL settings
  private char[] keystorePassword, keyPassword; //injected from application.yml
}
```

Spring Boot: Authentication – FHIR Service Bean

```java
@Service public class FhirService {
  private final FhirContext fhirContext;
  private final RestTemplate fhirRestTemplate;
  private final URI endpoint = URI.create(
    "https://terminology-highmed.medic.medfak.uni-koeln.de/fhir".replaceAll("/$", ""));


  @SneakyThrows public <T extends Resource> T getResourceFromPath(String path, Class<T> clazz,
Object... variables) throws HttpClientErrorException {
    String resolved = String.format("%s/%s", endpoint, path);
    ResponseEntity<String> response = fhirRestTemplate.getForEntity(resolved,
      String.class, variables);
    if (response.getStatusCode().is2xxSuccessful())
      return fhirContext.newJsonParser().parseResource(clazz, response.getBody());
    throw new HttpClientErrorException(response.getStatusCode(),
      String.format("Error requesting %s", resolved.toString()));
  }
  //…Constructor…
}
```

Spring Boot: Authentication – FHIR Service Bean

```java
@Controller @RequestMapping("/metadata")
public class MetadataDemoController {
  private final FhirService fhirService; //inject FHIR service
  private final CapabilityStatement metadata; //request the metadata once, it won't change often

  @SneakyThrows public MetadataDemoController(FhirService fhirService) {
    this.fhirService = fhirService;
    this.metadata = fhirService.getResourceFromPath("metadata", CapabilityStatement.class);
  }

  @SneakyThrows @GetMapping("/version")
  public ModelAndView queryVersion(Map<String, Object> model) {
    model.put("data", String.format("%s %s",
      metadata.getSoftware().getName(), metadata.getSoftware().getVersion()));
      //write data to view model
    return new ModelAndView("simpleData", model);
  }
}
```

File: spring/src/main/java/de/uzl/itcr/fhir_term_samples/spring/metadata/
MetadataDemoController.java

11

List Resources

- https://endpoint.de/fhir/{CodeSystem, ValueSet, ConceptMap} 🌐
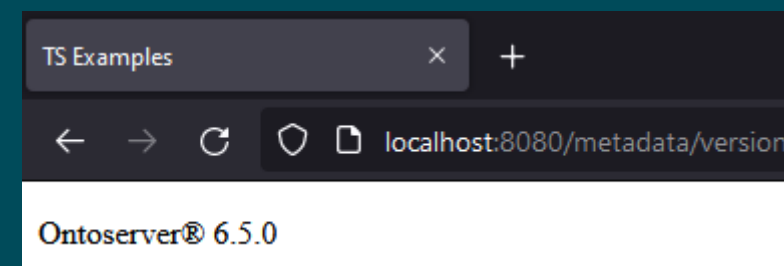
Search Results (Bundle)

Bundle entries: 34

Type	Id	Title	URL	Version	
CodeSystem	900000000000207008-20200309	SNOMED CT core	http://snomed.info/sct	http://snomed.info/sct/900000000000207008/version/20200309	→
CodeSystem	gecco-codesystem-ecrf-parameter-codes	Parameter Codes eCRF	https://www.netzwerk-universitaetsmedizin.de/fhir/CodeSystem/ecrf-parameter-codes	1.0.4	→
CodeSystem	atc	Anatomisch-therapeutisch chemische Klassifikation (ATC) Amtliche deutsche Fassung 2020	http://fhir.de/CodeSystem/dimdi/atc	2020	→
CodeSystem	ops2015	OPS	http://fhir.de/CodeSystem/dimdi/ops	2015	→
CodeSystem	ops2019	OPS	http://fhir.de/CodeSystem/dimdi/ops	2019	→
CodeSystem	LOINC-2.68	LOINC	http://loinc.org	2.68	→
CodeSystem	icd10gm2018	ICD-10	http://fhir.de/CodeSystem/dimdi/icd-10-gm	2018	→
CodeSystem	gecco-codesystem-frailty-score	Frailty Score	https://www.netzwerk-universitaetsmedizin.de/fhir/CodeSystem/frailty-score	1.0	→
CodeSystem	ops2021	OPS	http://fhir.de/CodeSystem/dimdi/ops	2021	→

Helper API

```python
from requests import Session


class FhirApi:
    def __init__(self, cert_file: str = "dfn.pem", endpoint: str = "https://terminology-…"):
        self.session = Session(); self.cert_file = cert_file; self.session.cert = self.cert_file
        self.endpoint = endpoint.rstrip("/") # remove slash at end to make sure that joining the path works


    def build_url(self, path: str) -> str:
        return self.endpoint + "/" + path.lstrip("/") # remove slash at beginning also


    def request_and_parse_fhir(self, path: str, resource):
        request_url = self.build_url(path)
        response = self.session.get(request_url)
        if response.status_code >= 200 and response.status_code < 300:
            try: # very simplistic error handling
                return resource(**response.json()) # parse with given class
            except Exception as e:
                raise ValueError(f"Parsing the response was not possible") from e
        else:
            raise SystemError(f"Error requesting from {request_url}, status code {response.status_code}")
```

List resources

```python
from get_session import FhirApi
from fhir.resources.bundle import Bundle


fhir_api = FhirApi() # use default settings
bundle : Bundle = fhir_api.request_and_parse_fhir("CodeSystem", Bundle)
# request a bundle of Code Systems
resources = [r.resource for r in bundle.entry] # we only care about the entries
resources.sort(key=lambda r: (r.name if r.title is None else r.title, r.version))
# order by name/title and version
for r in resources:
    name = r.name if r.title is None else r.title
    print(f" – '{name}' ({r.url}, version {r.version})")


# – 'AdministrativeGender' (http://hl7.org/fhir/administrative-gender, version 4.0.1)
# – 'Anatomisch-therapeutisch chemische Klassifikation (ATC) Amtliche deutsche Fassung
#    2020'(http://fhir.de/CodeSystem/dimdi/atc, version 2020)
```

Code Validation within CodeSystems/ValueSets

- `https://endpoint.de/fhir/CodeSystem/`$validate-code`
  `?code=A01`
  `&url=http://fhir.de/CodeSystem/dimdi/atc` 🌐

- `https://endpoint.de/fhir/ValueSet/`$validate-code`
  `?url=https://www.netzwerk-`
  `universitaetsmedizin.de/fhir/ValueSet/known-exposure`
  `&system=http://snomed.info/sct`
  `&code=840546002` 🌐

- **Remember**: Concepts are identified by tuples of *(Canonical URL, Code)!*
  - *Also, consider versions.*

- Try your best to not query instances by ID, like
  https://endpoint.de/fhir/ValueSet/1.2.276.0.76.11.520--20200608123231

Code Validation within CodeSystems

```python
# imports...
# request the list of code systems from API
# prompt user for CodeSystem and code interactively
request_path = f"CodeSystem/$validate-code?code={code}&url={url}&version={version}"
parameters: Parameters = fhir_api.request_and_parse_fhir(request_path, Parameters)
# retrieve the parameters as applicable from the Parameters class
result = next(p for p in parameters.parameter if p.name == 'result').valueBoolean
if result:
    display = next(p for p in parameters.parameter if p.name == "display").valueString
    print(f"The code '{code}' ('{display}') belongs to the CodeSystem.")
else:
    message = next(p for p in parameters.parameter if p.name == "message").valueString
    print(message)
```

Code Validation within CodeSystems

...and within ValueSets

```
? ValueSet to use? AceInhibitorsATC (https://www.netzwerk-universitaetsmediz
in.de/fhir/ValueSet/ace-inhibitors-atc v1.0)
? Code System of the code? http://fhir.de/CodeSystem/dimdi/atc version None
? Code: C09A
The code 'C09A' ('ACE-HEMMER, REIN') belongs to the ValueSet.
```

```
? ValueSet to use? BirthSex (https://www.netzwerk-universitaetsmedizin.de/fh
ir/ValueSet/birth-sex v1.0)
? Code System of the code? (Use arrow keys)
 » http://hl7.org/fhir/administrative-gender version None
   http://fhir.de/CodeSystem/gender-amtlich-de version None
```

```
? ValueSet to use? BirthSex (https://www.netzwerk-universitaetsmedizin.de/fh
ir/ValueSet/birth-sex v1.0)
? Code System of the code? http://hl7.org/fhir/administrative-gender version
 None
? Code: QUZ
The specified code 'QUZ' is not known to belong to the specified code system
'http://hl7.org/fhir/administrative-gender'
```

Query Codes from ValueSets with Filter

- Querying the entire CodeSystem is generally not advised

- Use the expansion of the relevant ValueSet instead

- `http://endpoint.de/fhir/ValueSet/$expand`
  `?url=http://hl7.org/fhir/ValueSet/dicom-dcim`
  `&version=01`
  `&filter=frame` 🌐

- CodeSystems (often) have canonical URL assigned to implicit ValueSet with all concepts that can be used for $expand

Query Codes from ValueSets with Filter

```python
# request the list of code systems with valueSet set, and ValueSets, from API
# prompt user for CS/VS, and for optional filter
request_path = f"ValueSet/$expand?version={version}&url={url}"
if filter.strip():
    request_path += f"&filter={filter}"
valueset: ValueSet = fhir_api.request_and_parse_fhir(request_path, ValueSet)
# prompt user for selection of a code from (filtered) list
lookup_path = f"CodeSystem/$lookup?code={sel_code}&system={sel_url}"
# lookup details of the selected concept
lookup: Parameters = fhir_api.request_and_parse_fhir(request_path, Parameters)
def print_parameter_value(param_name: str, parameters: Parameters, print_name = None):
    # rather easy print routine
print_parameter_value("name", lookup, "CodeSystem Name")
print_parameter_value("version", lookup, "CodeSystem Version")
print_parameter_value("display", lookup, "Code Display")
print_parameter_value("designation", lookup, "Code Designation")
```

Query Codes from ValueSets with Filter

Expand SNOMED CT ECL

- Consider availability of multiple SNOMED CT editions/versions

- `https://endpoint.de/fhir/ValueSet/$expand?url=http://snomed.info/sct/32506021000036107/version/20210630%3Ffhir_vs=ecl/%3C%3C%20305060011000036107%20%7CAustralian%20product%7C%3A%20700000101000036108%20%7ChasTP%7C%20%3D%2017311000168105%20%7CPanadol%7C` 🌐

  AU edition

  '?'

- `<< 30506011000036107 |Australian product|: 700000101000036108 |hasTP| = 17311000168105 |Panadol|`

- Make sure to URL-encode your query parameter!

Expand SNOMED CT ECL

```python
# request the list of all SNOMED CT version on this server
request_path = "CodeSystem?url=http://snomed.info/sct"
refset_re = r"sct\/(\d*)\/"
version_re = r"(\d*)$"
snomed_bundle = fhir_api.request_bundle(request_path)
# prompt user for selection of one edition, and version of that edition
print(f"Using version {version} for refset {edition}")
encoded_ecl = urllib.parse.quote(ecl) # ask for, and url-encode ECL string
expansion_url = f"ValueSet/$expand?url=http://snomed.info/sct/{edition}/version/{version}?fhir_vs=ecl/{encoded_ecl}"
vs: ValueSet = fhir_api.request_and_parse_fhir(expansion_url, ValueSet)
print(f"There are {vs.expansion.total} concepts in the expression:")
if vs.expansion.total > 0:
    for concept in vs.expansion.contains:
        print(f" - {concept.code} |{concept.display}|")
```

Expand SNOMED CT ECL

```
(venv) → python git:(main) x python expand_ecl.py
? Which edition of SNOMED CT to use? (Use arrow keys)
 » 'NA' (refset 999000031000000106)
   'SNOMED CT United Kingdom Edition reference set module' (refset 999000031000000106)
   'SNOMED CT United Kingdom composition module' (refset 83821000000107)
   'SNOMED Clinical Terms Australian extension' (refset 32506021000036107)
   'module van Nederlandse editie' (refset 11000146104)
```

```
                                                                      [15/2912]
    20190831
    20210131
? Which version of the selected refset to use? 20210630
Using version 20210630 for refset 32506021000036107
? ECL Expression? << 30506011000036107 |Australian product|: 700000101000036108 |hasTP| = 173110001681
05 |Panadol|
Encoded ECL expression: %3C%3C%2030506011000036107%20%7CAustralian%20product%7C%3A%2070000010100003610
8%20%7ChasTP%7C%20%3D%2017311000168105%20%7CPanadol%7C
There are 22 concepts in the expression:
 - 1510571000168109 |Panadol 500 mg suppository, 10|
 - 1510581000168107 |Panadol 500 mg suppository, 10, blister pack|
 - 1209501000036105 |Panadol 500 mg suppository, 24|
 - 1846401000036107 |Panadol 500 mg suppository, 24, strip pack|
```

Conversion to FHIR Resources

- Create CodeSystem for local codes

- ValueSet for mapped LOINC codes
  - Validation of all concepts, since they might be entered-in-error

- Target ValueSet clarifies the context of our mapping ⊕

- ConceptMap for mapping from Local → LOINC

Conversion to FHIR Resources – Step 1: Query Database

```python
import sqlite3

# select everything from the db
sql = "SELECT code, display, unit, loinc FROM lab_codes;"
sqlconn = sqlite3.connect("../legacydb.sqlite3")
sqlconn.row_factory = sqlite3.Row # access rows using Row interface, instead of tuples
cur = sqlconn.cursor()
# list the available concepts from the DB as list of dict
# loads everything into memory. Propably not a good idea for all real applications...
defined_concepts : List[Dict[str, str]] = []
for row in cur.execute(sql):
    defined_concepts.append(dict(zip(row.keys(), row)))
    # this produces a list of dicts with the column names as keys
```

Conversion to FHIR Resources – Step 2: Define Attributes

```python
# query for attributes of the FHIR CodeSystem
cs_answers = questionary.form(
    url = questionary.text("Canonical URL of the CodeSystem?"),
    valueSet = questionary.text("Canonical URL of the implicit ValueSet?"),
    version = questionary.text("Version?", validate=NotEmptyValidator) # …).ask()
cs_answers.update({
    "id": cs_answers["name"], # the name is used as the ID for simplicity
    "content": "complete" # content will generally be complete for simple CS
})
code_system = CodeSystem(**cs_answers)
code_system.property = [CodeSystemProperty(**{"code": "unit", "type": "string"})]
```

Conversion to FHIR Resources – Step 3: Create Concepts

```python
code_system.concept = [
    CodeSystemConcept(**{
        "code": c["code"],
        "display": c["display"],
        "property": [
            CodeSystemConceptProperty(**{
                "code": "unit",
                "valueString": c["unit"]
                })
            ]
        }) for c in defined_concepts
    ]
```

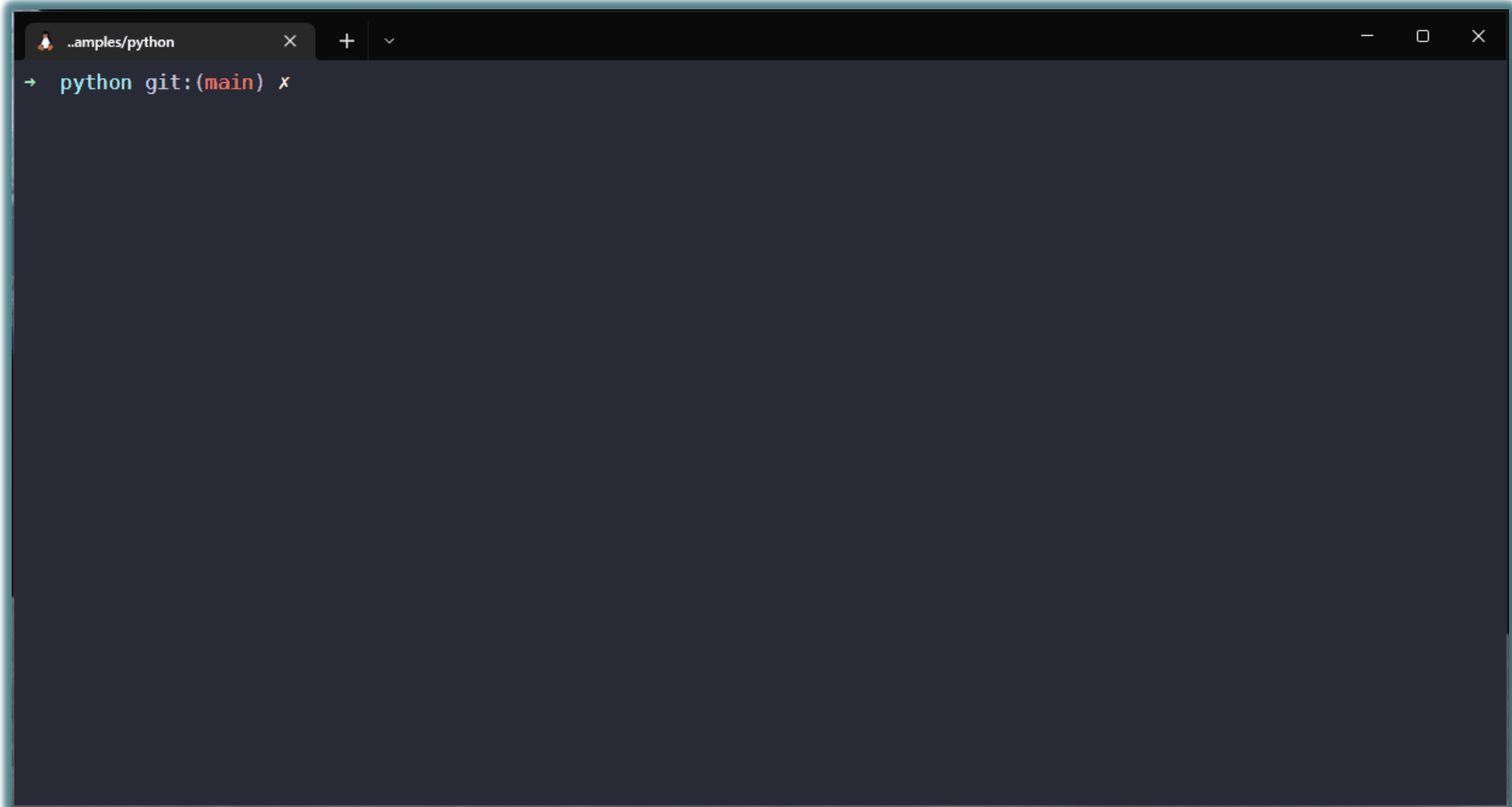# Conversion to FHIR Resources – Step 4: Enumerate & Validate LOINC

```python
# create valueset from user answers, as with CodeSystem
valueset_concepts = []
loinc_concepts = []
for concept in defined_concepts:
    loinc = concept["loinc"]
    if loinc is None:
        # loinc is nullable in DB
        continue
    valid, display = fhir_api.validate_code_get_display("http://loinc.org", loinc)
    # requests http://endpoint.de/fhir/CodeSystem/$validate-code
    # ?system=http://loinc.org&code={code}
    if not valid:
        # skip the concepts that are invalid
        continue
    loinc_concepts.append({"code": loinc, "display": display})
```

Conversion to FHIR Resources – Step 5: Create VS Compose

```python
valueset.compose = ValueSetCompose(**{
    "include": [{
        "system": "http://loinc.org",
        "concept": [ValueSetComposeIncludeConcept(**c) for c in loinc_concepts]
    }]
})


# creating the ConceptMap is more of the same patterns!
```

Conversion to FHIR Resources


```
..amples/python

→  python git:(main) ✗
```

File: python/create_resources.py

31

Contact

Joshua Wiedekopf, M.Sc.

Research Associate

University of Lübeck

IT Center for Clinical Research &

Institut für Medizinische Informatik

Ratzeburger Allee 160

23562 Lübeck

- j.wiedekopf@uni-luebeck.de
- @jpwiedekopf
- www.linkedin.com/in/jpwiedekopf