FHIR Intermediate Course

MODULE 3: FHIR SERVERS

Assignments

Language: English

V1.0

Course Overview

Module I: Implementation Guides

Most relevant FHIR Implementation Guides: Argonaut & IPS
Argonaut Development and Roadmap
Argonaut Data Query IG: Scope, Use Cases
Argonaut Provider Directory IG: Scope, Use Cases
IPS FHIR IG: Scope, Use Cases

Module II: FHIR Clients

General Guidelines for FHIR Clients
FHIR Clients in JavaScript / C# / Java [1 - Elective]

Module III: FHIR Facades

Why use FHIR Server Facade: your system on FHIR
Specific FHIR Servers (FHIR Facade)
Facade Use Case / Scenarios
Facade Architecture / Patterns
Where to put the FHIR Facade
System Integration / Integration Engine / Bus / Messaging
Facade in Java / Node.JS [1 - Elective]

Module IV: FHIR Applications

Smart-On-FHIR
CDS-Hooks
Integration with Smart-On-FHIR / CDS-Hooks [1 - Elective]

Assignments For Module 3

Goals and Options

In this week will have formal “programming” assignments, so the assignments in this unit are mostly “Build your Own” . If you want to review code, just review the code described below, in the “Where to Look” section.

We will ask you to change or enhance some apps built using the Java or JS FHIR server facades.

Platform

Platform	Description
P01	Java
P02	node.js

Assignment Levels

L01	Add search parameter	40 points
L02	Add support for a new resource - Search / Search by Id	40 points
L03	Add support for a related resource	20 points (advanced)

Total grade for each track and platform is 100 points
You need to obtain at least 60 points in order to complete this unit.
You can try more than one platform and/or track if you are able to, we will grade you with the maximum grade you obtained (up to 100).

General Introduction

Disclaimer: The code included for these assignments is not supposed to be production-ready and of course is not ready for storing or retrieving information about actual patients.
It's only used to illustrate FHIR concepts and techniques and make you think on possible strategies and implementation of the methods described through this unit.

For this assignment we are providing you with similar projects that are already running, developed using the 2 technologies we use in this course. All of them are **server applications exposing an API (they don´t have a front end)**

- **Java (Spring Boot project)**
- **Node.JS**
-

These projects are located in the FHIR Server course block, under the Material Tools FHIR Servers

You can edit them using your favorite editor/IDE (we used **Visual Studio Code 3** for and Node.JS, and **Netbeans** for Java)

The applications share the same data model

The node.js app has a local database (**sqlite)** Node.JS
The java app uses an **H2** (in memory) database
The local database stores patient basic demographic data, and some identifiers

The projects implement a façade (create, read, search) for the legacy database.

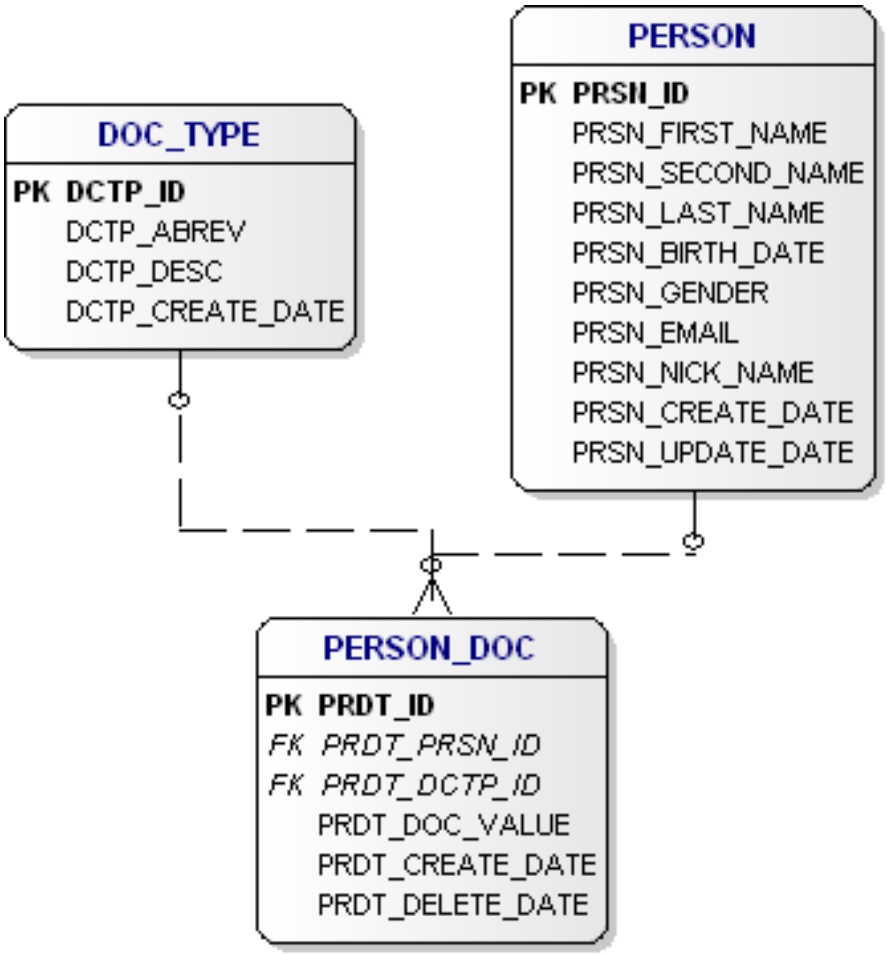
We tried to use the bare minimum dependencies to make a small functional app, like a minimum viable product embryo, just to facilitate the deployment and test of your solutions for the assignment.

Legacy Database Structure

The legacy database has the same three tables for Java and node.js

The database is pre-loaded with data so you can search by parameters and id without adding any record.
Disclaimer: The data inside of the database is completely fictional, if there is any name of actual persons, is just a coincidence.

Database Diagram



Database Dictionary

Tables

PERSON: Patient records, one for each patient

COLUMN	DESCRIPTION
PRSN_ID	Person unique identifier for the server, autonumeric
PRSN_FIRST_NAME	First official name
PRSN_SECOND_NAME	Second official name
PRSN_LAST_NAME	Family Name
PRSN_BIRTH_DATE	Date of Birth
PRSN_GENDER	Person Gender “male”,”female”
PRSN_EMAIL	Email address for the person
PRSN_NICK_NAME	Nick name for the person
PRSN_CREATE_DATE	time stamp for creation
PRSN_UPDATE_DATE	time stamp for update

DOC_TYPE: Document (as an identifying documentation) types: passport, SSN, etc.

COLUMN	DESCRIPTION
DCTP_ID	Identifier document ID
DCTP_ABREV	Identifier document abbreviated code
DCTP_DESC	Identifier document description

PERSON_DOC: Identifying documents for each person

COLUMN	DESCRIPTION
PRDT_ID	Unique identifier for the relationship between person
PRDT_PRSN_ID	Person holding the identifying document
PRDT_DCTP_ID	Id for the identifying document
PRDT_DOC_VALUE	Actual identifying document number (‘identifier’)
PRDT_CREATE_DATE	Time stamp for the creation
PRDT_UPDATE_DATE	Time stamp for the deletion

Each person has 1 or more records in PERSON_DOC
DOC_TYPE just defines the kind of identifier document

We are not allowed to change the database structure, so if anything is missing from this picture, we need to inject it through the adapters and mappers.

Mapping from Legacy To Patient

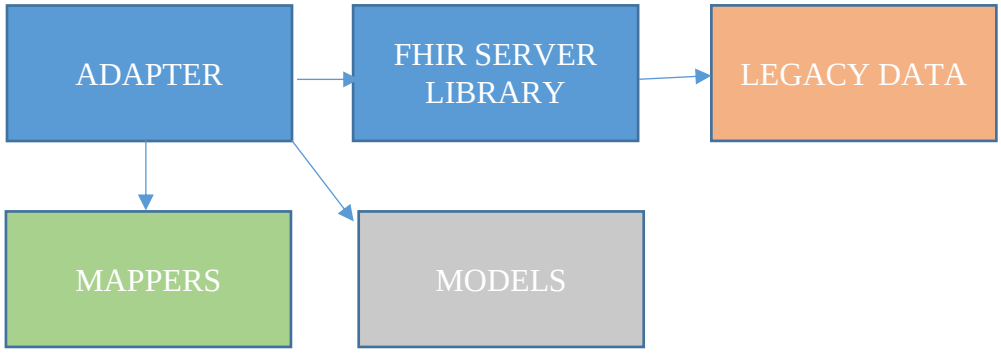
This is the result of mapping the legacy database content for a patient into a FHIR Patient resource.

```
{
    "resourceType": "Patient",
    "id": "{{PERSON.PRSN_ID}}",
    "text": {
        "status": "generated",
        "div": "<div
xmlns=\"http://www.w3.org/1999/xhtml\">{{PERSON.PRSN_ID}}</div>"
    },
    "identifier": [**THE OFFICIAL IDENTIFIER IS THE HOSPITAL ID - RESOURCE ID**
        {
            "use": "official",
            "system": "https://saintmartinhospital.org/patient-id",
            "value": "{{PERSON.PERSON.PRSN_ID}}",
            "period": {"start": "{{PERSON. PRSN_CREATE_DATE}}"
            }
        },
        { **OTHER IDENTIFIERS ARE 'USUAL' AND THERE IS MAP DOCTYPE->SYSTEM**
            "use": "usual",
            "system": "https://www.national-office.gov/ni",
            "value": "{{PERSON_DOC.PRDT_DOC_VALUE}}",
            "period": {"start": "{{PERSON_DOC.PRDT_CREATE_DATE}}"}
        }
    ],
    "name": [
        {
            "use": "official",
            "text": "{{PERSON.PRSN_FIRST_NAME}} {{PERSON.PRSN_SECOND_NAME}}
{{PERSON.PRSN_LAST_NAME}} ",
            "family": "{{PERSON.PRSN_LAST_NAME}}",
            "given": [ "{{PERSON.PRSN_FIRST_NAME}}",
                       "{{PERSON. PRSN_SECOND_NAME}}"
            ]
        },
        {
            "use": "nickname",
            "given": ["{{PERSON.PRSN_NICK_NAME}}"]
        }
    ],
    "telecom": [
        {
            "system": "email",
            "value": "{{PERSON.EMAIL}}"
        }
    ],
    "gender": "{{PERSON.PRSN_GENDER}}", **NO MAPPING NEEDED**
    "birthDate": "{{PERSON. PRSN_BIRTH_DATE}}" ** NO CONVERSION NEEDED**
}
```

Where to Look

Where is the FHIR? (a FHIR pun is mandatory in every FHIR related material, we are very sorry about this):

Remember that these are mainly API facades, so usually you will have a model for the legacy database, and some combinations of services implementing adapters and mappers from/to the FHIR format from/to the legacy format.



Platform	Adapter/Models/Mappers	Model
JS fhir-server-core	patient.service.js	Models*.* (for legacy db)
Java FHIR HAPI	<<Please see the videos>>	<<Please see the videos>>

In order to test the APIs, we included **two postman collections**, one for the Java server and one for the Node.js server (could have been the same, but for comfort, they have the default server URL/port configured)

1. Assignment L01-1: Add search parameter

Valid for all tracks and platforms: P01,P02

	**BUILD YOUR OWN**

The current project does not support the searching by email address.
Customers are requiring to search a person by email.
Your task: fix the app to include this search

2. Assignment L01-2: Add support for Practitioner

Valid for all tracks and platforms: T01,T02

BUILD YOUR OWN

We've found out that the organization uses the same db structure for Practitioners.
Your job: create the same façade but for the Practitioner resource, only implementing basic search and read by id.
Option 1: You can create a different project and database.
Option 2: It should work in the same project. The only difference is that they've added a new identifier called NPI, so the persons with an NPI document identifier are practitioners, all the others are just patients.

3. Assignment L01-3: Add support for a related resource

Valid for all tracks and platforms: P01,P02

	**BUILD YOUR OWN**

Add support for searching the medication requests related to a patient.