Lesson Notes for Lab - Implementing CRUD Operations

Welcome to our first hands on lab: Implementing CRUD Operations on FHIR Resources.

In this lesson, we will practice what we learned over the last two lessons focusing on the fundamental CRUD operations—Create, Read, Update, and Delete—used to interact with FHIR resources.

We will start by examining Create operations using HTTP POST requests using a fictitious patient. You'll first create a FHIR patient resource, make sure you have a valid FHIR resource payload, and then post it to the FHIR server. .

Next, we will create an Observation resource and link it to the patient we just created. This will require that we update the Observation resource with the Patient URL.

Lastly, we will create a Medication Resource linking it to the Patient. We will also use a PATCH operation to change the dose of the prescribed medication.

This is a hands-on laboratory, and you should use the development environment that you set up in lesson three to practice these operations. Take 20-30 minutes to solve each challenge, but if you get stuck know that I will show you how you can solve the challenge.

However, I want to emphasize you will learn more by doing than watching, so try each of the challenges yourself. You have learned all the knowledge to succeed.

To quickly review CRUD operations—**Create, Read, Update, and Delete**—form the foundation of how FHIR resources are managed in a RESTful API.

**Create (POST)**, is used to add new resources to a FHIR server. When sending a **POST request**, the server generates a unique **FHIR ID** and returns a **201 Created** response.

Next, we have **Read (GET)**, which allows clients to **retrieve a resource** by specifying its ID or using **search parameters**. This is useful for **fetching patient records, clinical observations, or medication details**. If the resource is found, the server returns a **200 OK** status along with the requested data. If it does not exist, the response is **404 Not Found**.

For modifying existing resources, FHIR supports two update methods: **PUT (Full Update)** and **PATCH (Partial Update)**. **PUT** replaces the entire resource with the new data, making it useful for **overwriting outdated records**. **PATCH**, on the other hand, updates **only specific fields** in a resource, which is more efficient for **small modifications**

Finally, **Delete (DELETE)** removes a resource from the FHIR server. While some systems perform **hard deletes**, others use **soft deletes**, where the resource remains but is marked as inactive.

FHIR follows a **structured URL format**, ensuring **consistency in accessing and managing resources**.

Let’s look at a patient we created using ClinFHIR. To access this patient's information on the fhir server we could use a GET request using a Rest API.

Breaking it down:

* http://127.0.0.1:8080/ → The **FHIR server address** (localhost for testing).
* /csp/healthshare/demo/fhir/r4/ → The **FHIR endpoint** specifying the **R4 version**.
* /Patient/cfsb1739559272444 → The **resource type (Patient)** and its **unique ID**.

FHIR Headers We Will Use

These have been provided in the exercise templates – you can refer to your lab setup.

FHIR Lab Instructions

Just read the slide

FHIR Lab Scenario Just read the slides