FHIR® DevDays Virtual
November 17-20, 2020

Let's build a FHIR app - .Net

Day 2 – Communicate with a FHIR server

Exercise: For this exercise, we will be using the mapped data from the first day to send the FHIR resources to a FHIR server. In addition, we will use a SMART on FHIR token to request data from a secured server, and display that.

Exercise steps

- Open your previously created app, or clone the day-1 branch with the example solution: https://github.com/FirelyTeam/LetsBuildNetFall2020/tree/day-1
- To your main code, add a using statement to include the FHIR RESTful client:

  ```
  using Hl7.Fhir.Rest;
  ```
- Create a new FhirClient object, pointing it to the public Firely test server "https://vonk.fire.ly/r4"
- With the FhirClient, you can use methods for the RESTful interactions
- After mapping a Patient, send it to the server to be created
  - If you use the Create method, the server will assign a new technical ID to the resource. This ID is important for the link between the Observation and the Patient, so you will have to make sure to update that reference in the Observations, if you want to send the Observations later.
  - Using the Update method, your Patient will be created with the technical ID you have assigned, or updated if a Patient with that technical ID already exists – please note that a production server will not always allow this
- Make sure to use the correct technical id for the reference in the Observation.Subject field
  - Either by using the server assigned ID, or – after an Update – the ID that was in the CSV or that you have chosen
  - You can choose to do this while mapping the Observation, or by updating the already mapped information if you have the server assigned ID
- Post the Observations to the server as well, using the Create interaction
  - Note that you would probably want to use a transaction for each set of Observations (1 WBC, 1 RBC, 1 HB), to make sure they are all created in one go, or – if one fails – the transaction as a whole fails. This goes beyond our exercise for today, but please ask any questions on a later day if you want to try this.
- Display your data, now including the technical ID of the Patient
  - With an HTTP tool like Postman, you can check to see the Patient resource is created on the server, using a 'GET <server>/Patient/<id>'

ORGANIZER firely

The public test server is open to everyone, and does not use authorization/authentication. In production systems you will run into needing this, and often it is implemented with OAuth2. Servers that support the SMART on FHIR app launch will be able to recognize and use a SMART on FHIR token. The next steps in this exercise will guide you to add such a token to the FhirClient, in order to be used on a request.

We assume you have a valid token – during DevDays we will provide one – and will not cover the whole OAuth2 dance to obtain one. There is a good topic on that to be found on chat.fhir.org, pointing to a couple of code projects to help you with that: https://chat.fhir.org/#narrow/stream/179171-dotnet/topic/SMART.20on.20FHIR.20app.20sample.20code.20in.20dotnet

- Adding a token to be sent with every request, is done by implementing your own message handler to be used by the FhirClient. An example for authorization would be:

```
public class AuthorizationMessageHandler : HttpClientHandler
{
        public System.Net.Http.Headers.AuthenticationHeaderValue Authorization { get;
set; }
        protected async override Task<HttpResponseMessage> SendAsync(HttpRequestMessage
request, CancellationToken cancellationToken)
        {
                if (Authorization != null)
                        request.Headers.Authorization = Authorization;
                return await base.SendAsync(request, cancellationToken);
        }
}
```

- Then add that to the FhirClient – note the different server base!:

```
var handler = new AuthorizationMessageHandler();
var bearerToken = "AbCdEf123456"; //example-token, replace with provided token
handler.Authorization = new AuthenticationHeaderValue("Bearer", bearerToken);
var client = new FhirClient("https://labs.vonk.fire.ly/r4", null, handler);
```

- Now you can request a Patient:

```
var pat = client.Read<Patient>("Patient/test");
```

- You can also perform a search for the Patient's white blood cell count Observations, resulting in a Bundle resource:

```
var q = new SearchParams("code", "http://loinc.org|6690-2");
var result = client.Search<Observation>(q);
```

- Add those results to an Observation list, and display the Patient and Observation data

- Do the same for red blood cell count, and hemoglobin Observations

Have fun, and remember to ask for help if you get stuck!