Let's build a FHIR app - .Net

Day 4 – Extra's and finalizing your app

Goal: Playtime to try out any of the previously learned skills, plus some extra's mentioned below.

Exercise (optional): For this exercise, we are going to add an extension to our Patient resources, and create a transaction Bundle to send a Patient and its Observations to the server in one go.

Exercise steps

- Open your previously created app, or clone the day-3 branch with the example solution: https://github.com/FirelyTeam/LetsBuildNetFall2020/tree/day-3
- In the sample-data-2.csv, we have added a place of birth to the patient data. Change your mapping code for the Patient to include that information.
- To do so, you will need to add an extension:
    - First, find an extension that is suitable for the data, so you know the canonical URL to use, and which data type to use for the value. In this case, there's a standard extension for place of birth listed on this page: http://hl7.org/fhir/patient-profiles.html
      If you need non-standard extensions, a good starting point to find them is Simplifier.net
    - Knowing the URL and type, you can add the data to your Patient:

```
var birthPlace = new Address() { City = record.PATIENT_BIRTHPLACE };
patient.Extension.Add(new Extension(
     "http://hl7.org/fhir/StructureDefinition/patient-birthPlace",
     birthPlace));
```

In order to send the Patient and Observations resources to the server in one go, we can create a transaction Bundle. This way, if anything fails, a rollback will take place server side. You can add any FHIR Rest interaction in a transaction entry, also with any conditions you could put on a single interaction. In our exercise we will only use the Create interaction, but you can try out the others as well.

- After mapping the resources, create a new method to setup and send the transaction(s). In our example code we have chosen to create a separate transaction for each patient plus its observations, but one large transaction with all data could also be an option.
- You can use the TransactionBuilder from the SDK, which will create the correct Bundle structure. Use your previously created FhirClient, or create a new one in your method, and make sure to set the type of the Bundle to 'transaction':

```
foreach (var p in patients)
{
    var builder = new TransactionBuilder(client.Endpoint,
                                         Bundle.BundleType.Transaction);

}
```

- After this, add your resource to the Bundle like this:

```
builder.Create(p);
```

- And do the same for each observation that is linked to the patient:

```
var patsObservations = observations.FindAll(o =>
                          o.Subject.Reference.Equals("urn:uuid:" + p.Id));

foreach (var o in patsObservations)
    builder.Create(o)
```

- Now we want to make sure that the patient resource gets a fullUrl added to the Bundle entry, because that will be used by the server to update the references in the observations correctly:

```
var transactionBundle = builder.ToBundle();
transactionBundle.Entry[0].FullUrl = "urn:uuid:" + p.Id;
```

- The last step is to use the FhirClient to send the transaction to the server:

```
var response = client.Transaction(transactionBundle);
```

- If you now want to show some details of the resulting response Bundle, you could serialize it to your preferred format (XML/JSON) and output those.

- Add a using statement to use the serializer from the SDK:

```
using Hl7.Fhir.Serialization;
```

- Serialize the Patient and first Observation resource from the response:

```
var serializer = new FhirJsonSerializer();
var createdPatient = serializer.SerializeToString(response.Entry[0]);
var firstObservation = serializer.SerializeToString(response.Entry[1]);
```

- Output those to the console to see the data as the server has stored it. Specifically take a look at the fullUrl of the Patient, and the value in the Observation.subject field. They should be updated and also should match.

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