Let's build a FHIR app - .Net

Day 3 – Validating FHIR resources

Exercise: For this exercise, we are going to validate our resources before sending them to a FHIR server. We will use the Validator of the Firely .NET SDK.

Exercise steps

- Open your previously created app, or clone the day-2 branch with the example solution: https://github.com/FirelyTeam/LetsBuildNetFall2020/tree/day-2
- To your main code, add the following using statement to include the validator:

```
using Hl7.Fhir.Validation;
using Hl7.Fhir.Specification.Source;
```

  For this, you have to include the NuGet package `Hl7.Fhir.Specification.R4`.

- Create a new Validator instance:

```
var validator = new Validator();
```

- Try to validate a single observation:

```
var outcome = validator.Validate(obs);
```

- Check the outcome of the validation operation. This outcome has some properties that you can use:
  - Success: a Boolean which indicates whether the validation was successful or not
  - Issue: a list of issues that were raised during validation
  - See this link for more information about the `OperationOutcome`.
- You will notice that the validation fails. The message "`[ERROR] Unable to resolve reference to profile 'http://hl7.org/fhir/StructureDefinition/Observation'`" is shown. The validator needs the standard Observation profile (StructureDefinition) to validate the instance. So, we must tell the validator where to find this this profile. We do this by passing a ResourceResolver to the validator. For all the standard HL7 FHIR resources, the SDK has a special ResourceResolver already made for you: `ZipSource.CreateValidationSource()`:

```
var resolver = ZipSource.CreateValidationSource();
var settings = ValidationSettings.CreateDefault();

settings.ResourceResolver = new CachedResolver(resolver);
var validator = new Validator(settings);
```

  Note that we wrap the standard ResourceResolver in a CachedResolver. This will speed up the validation when you validate more than 1 resource.

- Run the program again and you will see that the validation of Observation is successful.

- The field code in Observation is mandatory (see also https://www.hl7.org/fhir/observation.html). When we remove this code, the validator should report this. Try this out.

- The validator can also use other profiles to validate against. In the subdirectory profile, there is such a profile: MyObservation.StructureDefinition.xml. You can download this profile here.
  This profile is derived from the standard Observation profile and restrict the category of an Observation: at least 2 categories are mandatory.
  In order to use this profile we have to tell the validator where to find this profile. We do this with a `DirectorySource`:

  ```
  var directoryResolver = new DirectorySource("profiles");
  ```

  This will read all profiles in the subdirectory profiles.

  To combine this profile with the standard profiles we use the class `MultiResolver`. The code would be then:

  ```
  var resolver = ZipSource.CreateValidationSource();
  var directoryResolver = new DirectorySource("profiles");

  var settings = ValidationSettings.CreateDefault();
  settings.ResourceResolver = new CachedResolver(
                  new MultiResolver(resolver, directoryResolver));

  var validator = new Validator(settings);
  ```

- Let's validate an observation with the new profile:

  ```
  var outcome = validator.Validate(obs, new[] {
                 "http://fire.ly/fhir/StructureDefinition/MyObservation" });
  ```

  You will see that the validation fails, because we have no observation with minimal 2 categories.

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