



FHIR API for .Net programmers

Introduction

Mirjam Baltus

FHIR Developer Days

November 16, 2016



Who am I?



- **Name:** Mirjam Baltus
- **Company:** Furore, Amsterdam
- **Background:**
 - Furore FHIR team
 - IT trainer & Support
- **Contact:** m.baltus@furore.com





Using the Reference Implementation

HL7.FHIR API

First step



- Adding the HL7.Fhir package to your solution
 - NuGet Package manager, HL7.Fhir.DSTU2 package

A screenshot of the NuGet Package Manager window in Visual Studio. The window title is "NuGet Package Manager: Solution 'SkeletonFHIRClient'". The interface shows a list of packages on the left and a detailed view of the selected package on the right. The selected package is "HL7.Fhir.DSTU2". The "Action" dropdown is set to "Install" and the "Version" dropdown is set to "Latest stable 0.90.5". Under "Select which projects to apply changes to:", the checkbox for "1 project(s)" is checked, and the list below it shows "SkeletonFHIRClient" with its checkbox also checked. The "Show all" checkbox is unchecked.

NuGet Package Manager: Solution 'SkeletonFHIRClient'

Package source: Filter: ☐ Include prerelease

HL7.Fhir.DSTU2
This is the core support library for HL7's FHIR standard (<http://hl7.org/fhir>). It contains the core functionality to working with R...

HL7.Fhir
This is the core support library for HL7's FHIR standard (<http://hl7.org/fhir>).

HL7.Fhir.Specification.DSTU2
This library offers additional support beyond core for HL7's FHIR standard (<http://hl7.org/fhir>). It contains functionality to workin...

HL7.Fhir.DSTU2

Action: Version:

Select which projects to apply changes to:

☒ 1 project(s) ☐ Show all

☒ SkeletonFHIRClient

HL7.Fhir.DSTU2



■ Core contents

- Model – classes generated from the spec
- REST functionality – FhirClient
- Parsers and Serializers

■ Source on GitHub:

<http://github.com/ewoutkramer/fhir-net-api>

















THE MODEL

`using Hl7.Fhir.Model;`

A FHIR Resource



Name	Flags	Card.	Type	Description & Constraints
 Observation	I		DomainResource	Measurements and simple assertions <i>SHALL only be present if Observation.value[x] is not present</i> <i>Component code SHALL not be same as observation code</i>
 identifier		0..*	Identifier	Unique Id for this particular observation
 status	?! Σ	1..1	code	registered preliminary final amended + ObservationStatus (Required)
 category		0..1	CodeableConcept	Classification of type of observation Observation Category Codes (Example)
 code	Σ	1..1	CodeableConcept	Type of observation (code / type) LOINC Codes (Example)
 subject	Σ	0..1	Reference(Patient Group Device Location)	Who and/or what this is about
 encounter		0..1	Reference(Encounter)	Healthcare event during which this observation is made
 effective[x]	Σ	0..1		Clinically relevant time/time-period for observation
 effectiveDateTime			dateTime	
 effectivePeriod			Period	
 issued	Σ	0..1	instant	Date/Time this was made available
 performer	Σ	0..*	Reference(Practitioner	Who is responsible for the observation



A FHIR Resource in C#



```
public partial class Observation : Hl7.Fhir.Model.DomainResource
```

```
    status          ?! Σ  1..1  code          registered | preliminary | final |  
                                                            amended +  
                                                            ObservationStatus (Required)
```

```
/// <summary>
```

```
/// Codes providing the status of an observation.
```

```
/// (url: http://hl7.org/fhir/ValueSet/observation-status)
```

```
/// </summary>
```

```
public enum ObservationStatus {Registered, Preliminary, Final, ...}
```


```
var obs = new Observation();
```

```
obs.Status = Observation.ObservationStatus.Preliminary;
```




A FHIR Resource in C#



 code	Σ	1..1	CodeableConcept	Type of observation (code / type) LOINC Codes (Example)
--	---	------	-----------------	--

```
public CodeableConcept Code { get; set; }
```

```
obs.Code = new CodeableConcept("http://example.org", "my-example-code");
```

 identifier	0..*	Identifier	Unique Id for this particular observation
--	------	------------	---

```
public List<Identifier> Identifier{ get; set; }
```

```
obs.Identifier.Add(new Identifier("http://identifiers.example.org", "12345"));
```



A FHIR Resource in C#



value[x]	Σ	0..1	Actual result
valueQuantity			Quantity
valueCodeableConcept			CodeableConcept
valueString			string
valueRange			Range



```
public Element Value { get; set; }
```

```
var q = new Quantity();  
q.Value = 25;  
q.Unit = "sec";  
q.System = "http://unitsofmeasure.org";  
q.Code = "s";  
  
obs.Value = q;
```



A FHIR Resource in C#



 referenceRange	I	0..*	BackboneElement	Provides guide for interpretation <i>Must have at least a low or a high or text</i>
 low	I	0..1	SimpleQuantity	Low Range, if relevant

Observation (DomainResource)
identifier : Identifier [0..*] status : code [1..1] « ObservationStatus! » category : CodeableConcept [0..1] « Observation Category ?? » code : CodeableConcept [1..1] « LOINC ?? » subject : Reference [0..1] « Patient Group Device Location » encounter : Reference [0..1] « Encounter » effective[x] : Type [0..1] « dateTime Period »

ReferenceRange
low : Quantity(SimpleQuantity) [0..1] high : Quantity(SimpleQuantity) [0..1] meaning : CodeableConcept [0..1] « Observation Reference Range M...?? » age : Range [0..1] text : string [0..1]

[0..*]
referenceRange

[0..*]
referenceRange

```
public partial class ReferenceRangeComponent : BackboneElement  
{ ... }
```

```
var refRange = new Observation.ReferenceRangeComponent();  
// fill data for refRange  
obs.ReferenceRange.Add(refRange);
```



Primitives are not really primitive...



Patient (DomainResource)

identifier : Identifier [0..*]
active : boolean [0..1]

```
/// <summary>  
/// Whether this patient's record is in active use  
/// </summary>  
public Hl7.Fhir.Model.FhirBoolean Active {  
    get {  
        var pat = new Patient();  
        pat.ActiveElement = new FhirBoolean(true);  
        pat.Active = true;  
    }  
}
```



REST INTERACTIONS

`using Hl7.Fhir.Rest;`

Using the FHIR Client



- See [Publicly Available FHIR Servers](#) for available test servers

```
var client = new FhirClient("http://acme.org/fhir");  
  
// client options  
client.PreferredFormat = ResourceFormat.Xml;  
client.ReturnFullResource = true;
```



Adding headers to request

```
client.OnBeforeRequest +=  
    (object sender, BeforeRequestEventArgs e) =>  
{  
    e.RawRequest.Headers.Add("some_key", "some_value");  
};  
  
client.OnAfterResponse +=  
    (object sender, AfterResponseEventArgs e) =>  
{ ... };
```



C(RUD)



```
var obs = new Observation();  
obs.Status = Observation.ObservationStatus.Preliminary;  
obs.Code = new CodeableConcept("http://example.org",  
                                "my-example-code");  
  
var result = client.Create<Observation>(obs);  
  
// note that error handling should be added to catch exceptions
```



(C)RUD



```
var pat = client.Read<Patient>("Patient/1");
```

```
pat.Name.Add(HumanName.ForFamily("Kramer")  
            .WithGiven("Ewout"));
```

```
client.Update<Patient>(pat);
```

```
client.Delete(pat);  
client.Delete("Patient/12345");
```



Resource Identity



■ HL7.Fhir.Rest.ResourceIdentity

```
pat.ResourceIdentity().HasBaseUri  
pat.ResourceIdentity().HasVersion
```

```
pat.ResourceIdentity().BaseUri  
pat.ResourceIdentity().ResourceType
```

```
var id = new ResourceIdentity("Patient/3")  
                .WithBase("http://example.org/fhir");  
ResourceIdentity.Build(UrnType.OID, "1.2.3.4.5.6");
```



SEARCH AND BUNDLES

Making queries



```
var q = new SearchParams()  
    .Where("name=Ewout")  
    .Include("Patient:organization")  
    .LimitTo(10)  
    .SummaryOnly()  
    .OrderBy("birthdate",  
            Hl7.Fhir.Rest.SortOrder.Descending);  
  
q.Add("gender", "male");  
  
Bundle result = client.Search<Patient>(q);
```



Paging through Bundle



```
while (result != null)
{
    foreach (var e in result.Entry)
    {
        Patient p = (Patient)e.Resource;
        // do something with the resource
    }

    result = client.Continue(result, PageDirection.Next);
}
```



Transaction builder



```
var b = new TransactionBuilder("http://example.org/fhir")
    .Create(pat)
    .ResourceHistory("Patient", "7")
    .Delete("Patient", "8")
    .Read("Patient", "9")
    .Search(q, "Patient")
    .ToBundle();

var t_result = client.Transaction(b);
```



PARSING/SERIALIZING

`using Hl7.Fhir.Serialization;`

Parsing/Serializing



```
// Create a file-based reader for JSON
JsonTextReader reader =
    new JsonTextReader(new StreamReader(@"input.json"));

var parser = new FhirJsonParser();
var new_obs = parser.Parse<Observation>(reader);

// Serialize an in-memory observation to a JSON string
var jsonText =
    FhirSerializer.SerializeResourceToJson(new_obs);
```





WHAT'S NEXT?

Support



- Google group:
groups.google.com/forum/#!forum/fhir-dotnet
- Support questions:
fhirapi@furore.com
- GitHub for issues:
github.com/ewoutkramer/fhir-net-api/



Next Steps for **you**



- Try the Beginners Track during the hands-on session
- See <https://github.com/furore-fhir/fhirstarters> for the track exercises and some .Net examples
- For questions during the track, consult me or use <https://chat.fhir.org/>
- Go to the FHIR for Advanced .NET developers session



**The End –
Questions?**