HL7 FHIR DevDays 2017

HL7® FHIR®

FHIR API for .Net programmers - an introduction

Mirjam Baltus, Furore Health Informatics

Amsterdam, 15-17 November | @fhir_furore | #fhirdevdays17 | www.fhirdevdays.com

Who am I?

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

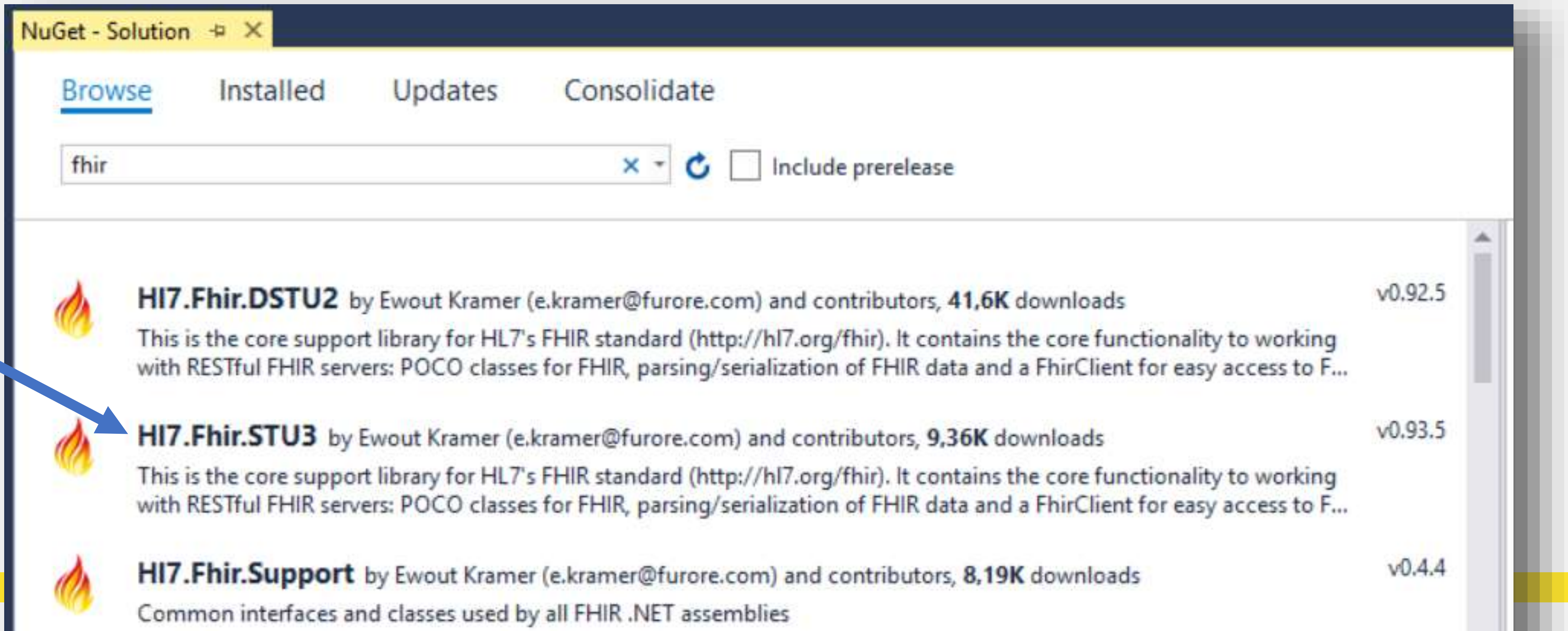
Using the Reference Implementation

Hl7.Fhir API

★ blue stars indicate example code

First step

- Adding the Hl7.Fhir package to your solution
  - NuGet Package manager, Hl7.Fhir.STU3 package



Hl7.Fhir.DSTU2 by Ewout Kramer (e.kramer@furore.com) and contributors, **41,6K** downloads v0.92.5
This is the core support library for HL7's FHIR standard (http://hl7.org/fhir). It contains the core functionality to working with RESTful FHIR servers: POCO classes for FHIR, parsing/serialization of FHIR data and a FhirClient for easy access to F...

Hl7.Fhir.STU3 by Ewout Kramer (e.kramer@furore.com) and contributors, **9,36K** downloads v0.93.5
This is the core support library for HL7's FHIR standard (http://hl7.org/fhir). It contains the core functionality to working with RESTful FHIR servers: POCO classes for FHIR, parsing/serialization of FHIR data and a FhirClient for easy access to F...

Hl7.Fhir.Support by Ewout Kramer (e.kramer@furore.com) and contributors, **8,19K** downloads v0.4.4
Common interfaces and classes used by all FHIR .NET assemblies

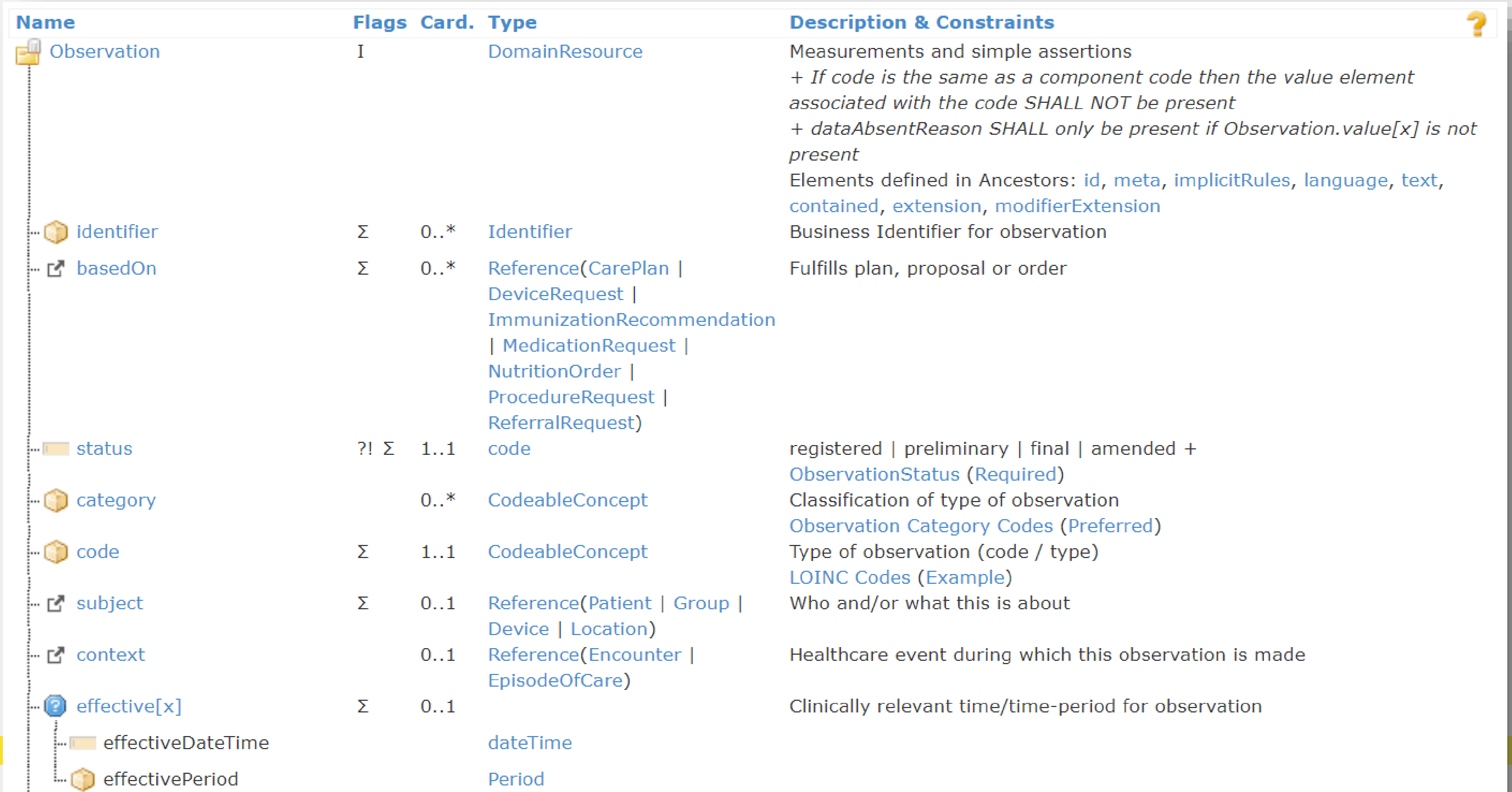
Hl7.Fhir.STU3

- Core contents
  - Model – classes generated from the spec
  - REST functionality – FhirClient
  - Parsers and Serializers
  - Helper functions
- Source on GitHub: http://github.com/ewoutkramer/fhir-net-api

The model

```
using Hl7.Fhir.Model;
```

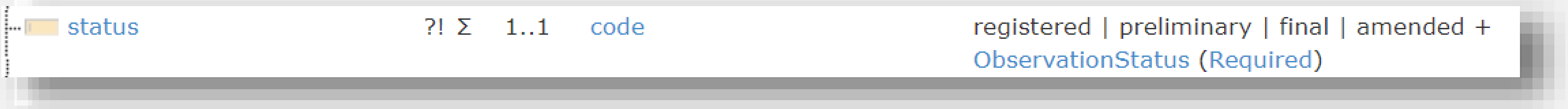
A FHIR Resource



Name	Flags	Card.	Type	Description & Constraints
Observation	I		DomainResource	Measurements and simple assertions *+ If code is the same as a component code then the value element associated with the code SHALL NOT be present* *+ dataAbsentReason SHALL only be present if Observation.value[x] is not present* Elements defined in Ancestors: id, meta, implicitRules, language, text, contained, extension, modifierExtension
identifier	Σ	0..*	Identifier	Business Identifier for observation
basedOn	Σ	0..*	Reference(CarePlan | DeviceRequest | ImmunizationRecommendation | MedicationRequest | NutritionOrder | ProcedureRequest | ReferralRequest)	Fulfills plan, proposal or order
status	?! Σ	1..1	code	registered | preliminary | final | amended + ObservationStatus (Required)
category		0..*	CodeableConcept	Classification of type of observation Observation Category Codes (Preferred)
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
context		0..1	Reference(Encounter | EpisodeOfCare)	Healthcare event during which this observation is made
effective[x]	Σ	0..1		Clinically relevant time/time-period for observation
effectiveDateTime			dateTime	
effectivePeriod			Period	

A FHIR Resource in C# - classes and enums

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



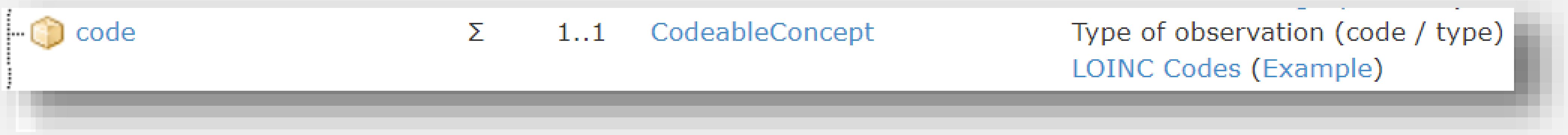
status	?! Σ	1..1	code	registered | preliminary | final | amended + ObservationStatus (Required)

```
   /// <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 = ObservationStatus.Preliminary;
  ```

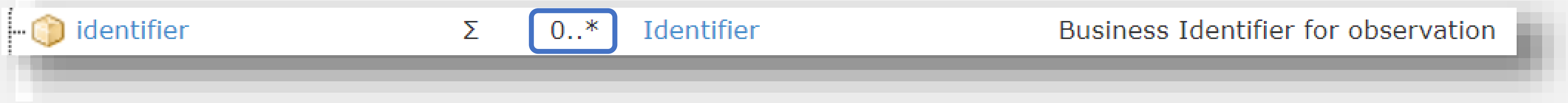
A FHIR Resource in C# - datatypes and lists



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", "EX123",
                                                   "Example code 123");
```

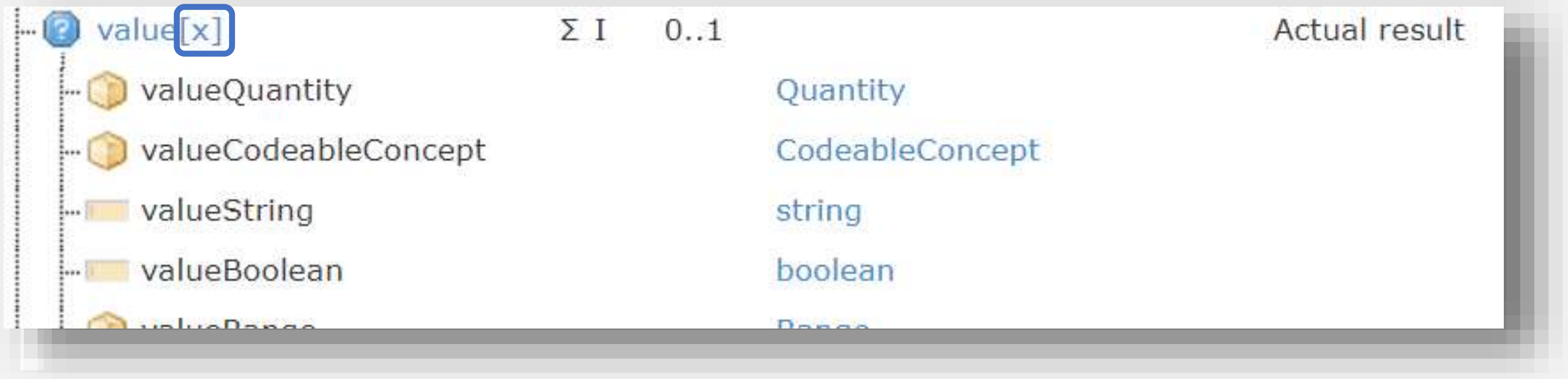


identifier	Σ	0..*	Identifier	Business Identifier for observation

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

★
```
obs.Identifier.Add(new Identifier("http://example.org", "123456"));
```

A FHIR Resource in C# - choice properties

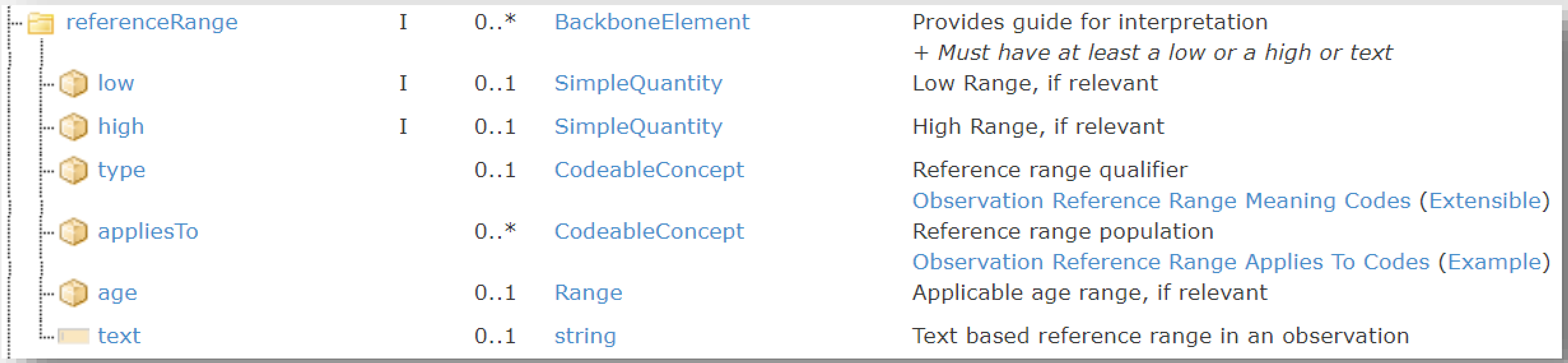


value[x]	Σ I	0..1	Actual result
valueQuantity		Quantity	
valueCodeableConcept		CodeableConcept	
valueString		string	
valueBoolean		boolean	
valueRange		Range	

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

```
★   var qty = new Quantity {
★       Value = 25,
★       Unit = "sec",
★       System = "http://unitsofmeasure.org",
★       Code = "s"          };

★   obs.Value = qty;
```

A FHIR Resource in C# - components



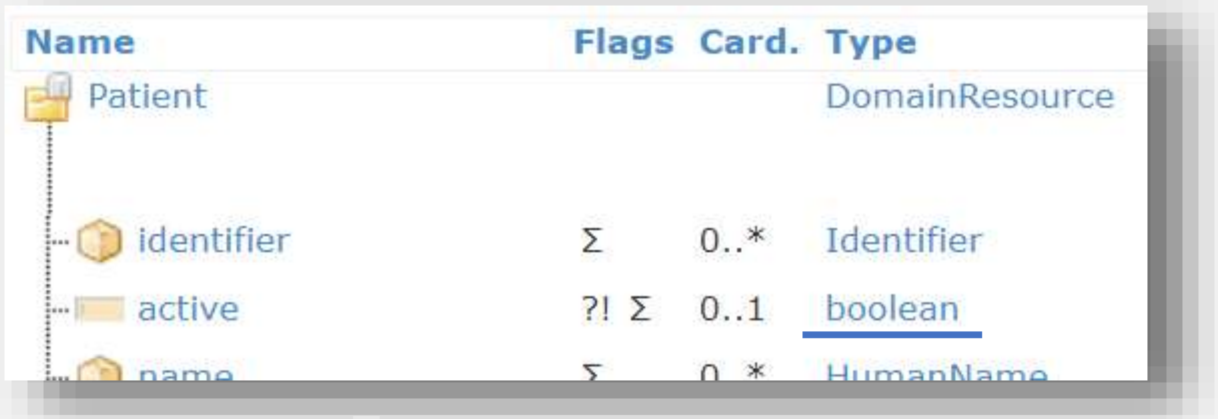
referenceRange	I	0..*	BackboneElement	Provides guide for interpretation + *Must have at least a low or a high or text*
low	I	0..1	SimpleQuantity	Low Range, if relevant
high	I	0..1	SimpleQuantity	High Range, if relevant
type		0..1	CodeableConcept	Reference range qualifier Observation Reference Range Meaning Codes (Extensible)
appliesTo		0..*	CodeableConcept	Reference range population Observation Reference Range Applies To Codes (Example)
age		0..1	Range	Applicable age range, if relevant
text		0..1	string	Text based reference range in an observation

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

```
★ var refRange = new Observation.ReferenceRangeComponent();
  // fill the values
★ obs.ReferenceRange.Add(refRange);
```

A FHIR Resource in C# - (non) primitives

```
/// <summary>
/// Whether this patient's record is
/// in active use
/// </summary>
public bool? Active { … }
public Hl7.Fhir.Model.FhirBoolean ActiveElement { … }
```

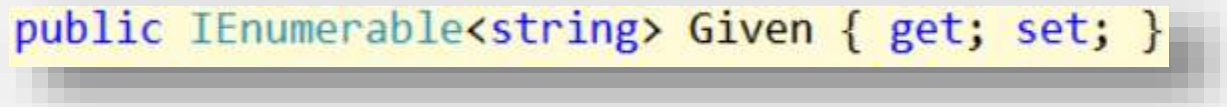


Name	Flags	Card.	Type
Patient			DomainResource
identifier	Σ	0..*	Identifier
active	?! Σ	0..1	boolean
name	Σ	0..*	HumanName

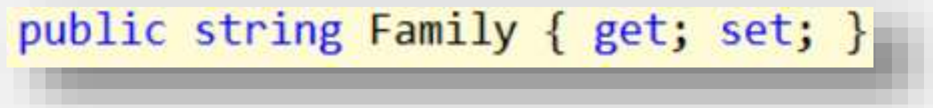
```
★ var pat = new Patient();

★ pat.Active = true; // or
★ pat.ActiveElement = new FhirBoolean(true);
```

Why would you use the non-primitive version?

```
★   var name = new HumanName();

★   name.Given = new string[] { "Mirjam" }; // or
★   name.GivenElement.Add(new FhirString("Mirjam"));

★   name.Family = "Baltus-Bakker";
```



```
public IEnumerable<string> Given { get; set; }
```



```
public string Family { get; set; }
```

- Adding extensions cannot be done on primitives!

Extensions

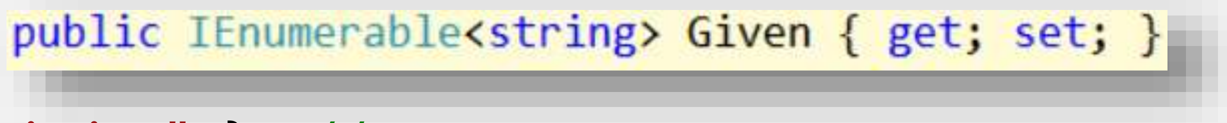
Key = location of formal definition

```
<Patient xmlns="http://hl7.org/fhir">
  <!-- some metadata and narrative -->
  <extension url="http://hl7.org/fhir/StructureDefinition/patient-
                                                      mothersMaidenName">
    <valueString value="Williams"/>
  </extension>
  <!-- more patient data -->
</Patient>
```

Value = type of value according to definition

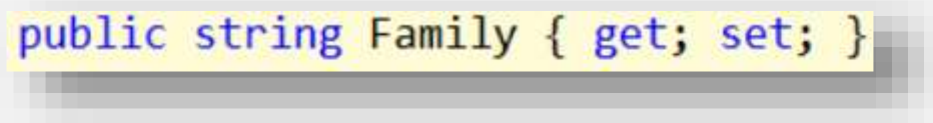
Why would you use the non-primitive version?

```
★    var name = new HumanName();
```



```
public IEnumerable<string> Given { get; set; }
```

```
★    name.Given = new string[] { "Mirjam" }; // or
★    name.GivenElement.Add(new FhirString("Mirjam"));
★    name.Family = "Baltus-Bakker";
```



```
public string Family { get; set; }
```

- Adding extensions cannot be done on primitives!

```
★    name.FamilyElement.AddExtension(
       "http://hl7.org/fhir/StructureDefinition/humanname-partner-name",
       new FhirString("Baltus"));
```

REST interactions

```
using Hl7.Fhir.Rest;
```

Using the FHIR Client

- For a list of test servers, see Publicly Available FHIR Servers

```
★    var client = new FhirClient("http://vonk.furore.com");

     // client options
★    client.PreferredFormat = ResourceFormat.Xml;
★    client.PreferredReturn = Prefer.ReturnRepresentation;
```

C(RUD)

```
★    var obs = new Observation();
★    obs.Status = ObservationStatus.Preliminary;
★    obs.Code = new CodeableConcept("http://example.org", "EX123",
                                                  "Example code 123");
     // fill in mandatory fields, plus other fields you have data for


     // send the observation to the server to be created
★    var result = client.Create<Observation>(obs);


     // note that this could generate an error,
     // so setup error handling to catch exceptions
```

(C)RUD

```
    // read a resource from the server
★   var pat = client.Read<Patient>("Patient/1");

    // update a resource on the server
★   pat.Name.Add(HumanName.ForFamily("Kramer").WithGiven("Ewout"));
★   client.Update<Patient>(pat);

    // delete a resource from the server
★   client.Delete(pat); // or
★   client.Delete("Patient/12345");
```

Adding headers to the request, inspecting raw response

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

client.OnAfterResponse +=
    (object sender, AfterResponseEventArgs e) =>
    {
        Console.WriteLine(e.RawResponse.StatusCode);
    };
```

Bundles and searches

```
using Hl7.Fhir.Rest;
```

Making queries

```
★    var q = new SearchParams()
★                  .Where("name=Ewout")
★                  .Include("Patient:organization")
★                  .LimitTo(10)
★                  .SummaryOnly()
★                  .OrderBy("birthdate", SortOrder.Descending);

★    q.Add("gender", "male");

★    Bundle result = client.Search<Patient>(q);
```

Paging through a 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);
}
```

Helper functionality

```
using Hl7.Fhir.Rest;
```

Resource Identity

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

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

★
```
var id = new ResourceIdentity("Patient/3")
                  .WithBase("http://example.org/fhir");
```

★
```
var id2 = ResourceIdentity.Build(UrnType.OID, "1.2.3.4.5.6");
```

Transaction builder

```
★    var trb = new TransactionBuilder("http://vonk.furore.com")
★                   .Create(pat)
★                   .ResourceHistory("Patient", "1")
★                   .Delete("Patient", "2")
★                   .Read("Patient", "pat2");


★    var q = new SearchParams().Where("name=Steve");
★    trb = trb.Search(q, "Patient");


★    var t_result = client.Transaction(trb.ToBundle());
```

Questions?

What’s next?

- Join me at the table for the hands-on session
- Look at the schedule for other tutorials in the developers track
- Code, have fun, and

ASK QUESTIONS!