HL7 FHIR DevDays 2017

HL7® FHIR®

Overview of FhirPath

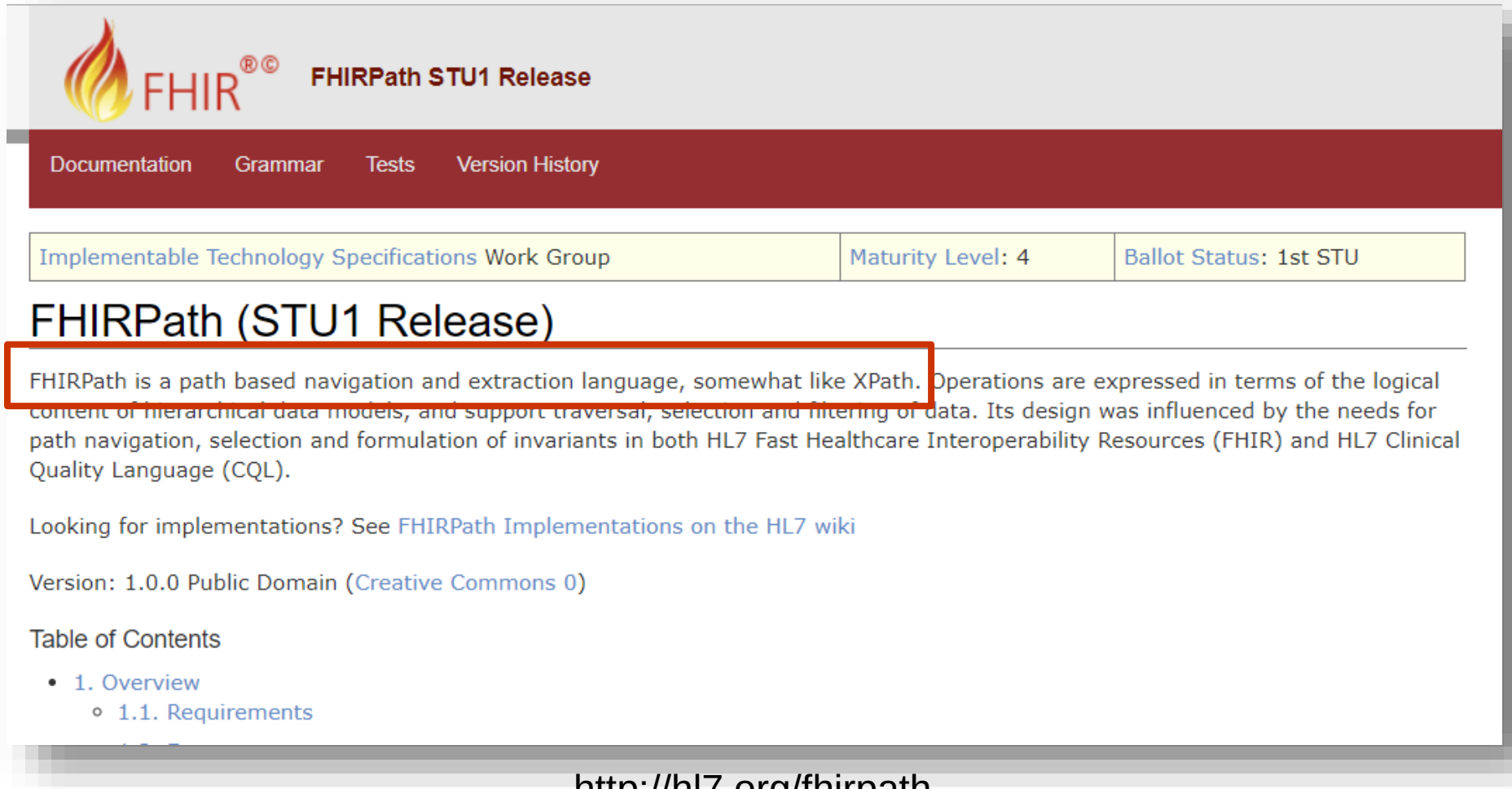
Ewout Kramer, Furore Health Informatics

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

Who am I?

- **Name:** Ewout Kramer
- **Company:** Furore Health Informatics
- **Background:**
  - Computer Science (operating systems)
  - In Health IT since 1999
  - FHIR Core team
  - Lead dev on the .NET API
- e.kramer@furore.com, @ewoutkramer
- http://thefhirplace.com

What is FhirPath?



http://hl7.org/fhirpath

Uses for FhirPath

- Navigate or “point” to parts of an instance, e.g. to define (new) search parameters
- Formulate predicates against the model in StructureDefinition
- Extract data from an instance, e.g. narrative generation

  - FluentPath has no facilities for updates to your data or doing transformations

What about XPath?

- XPath is tied to XML, we need it to work on “the FHIR data model” (whatever the serialization)
- Needed XPath 2.0 features – but no support for .NET exists
- (for most people) XPath statements are hard to write and get right...

XPath versus FhirPath

“If the resource is contained in another resource, it SHALL be referred to from elsewhere in the resource”

contained.all('#'+id in %resource.descendants().reference)

not(exists(for $id in f:contained/*/@id return $id[not(ancestor::f:contained/parent::*/descendant::f:reference/@value=concat('#', $id))]))

Standing on the shoulders...

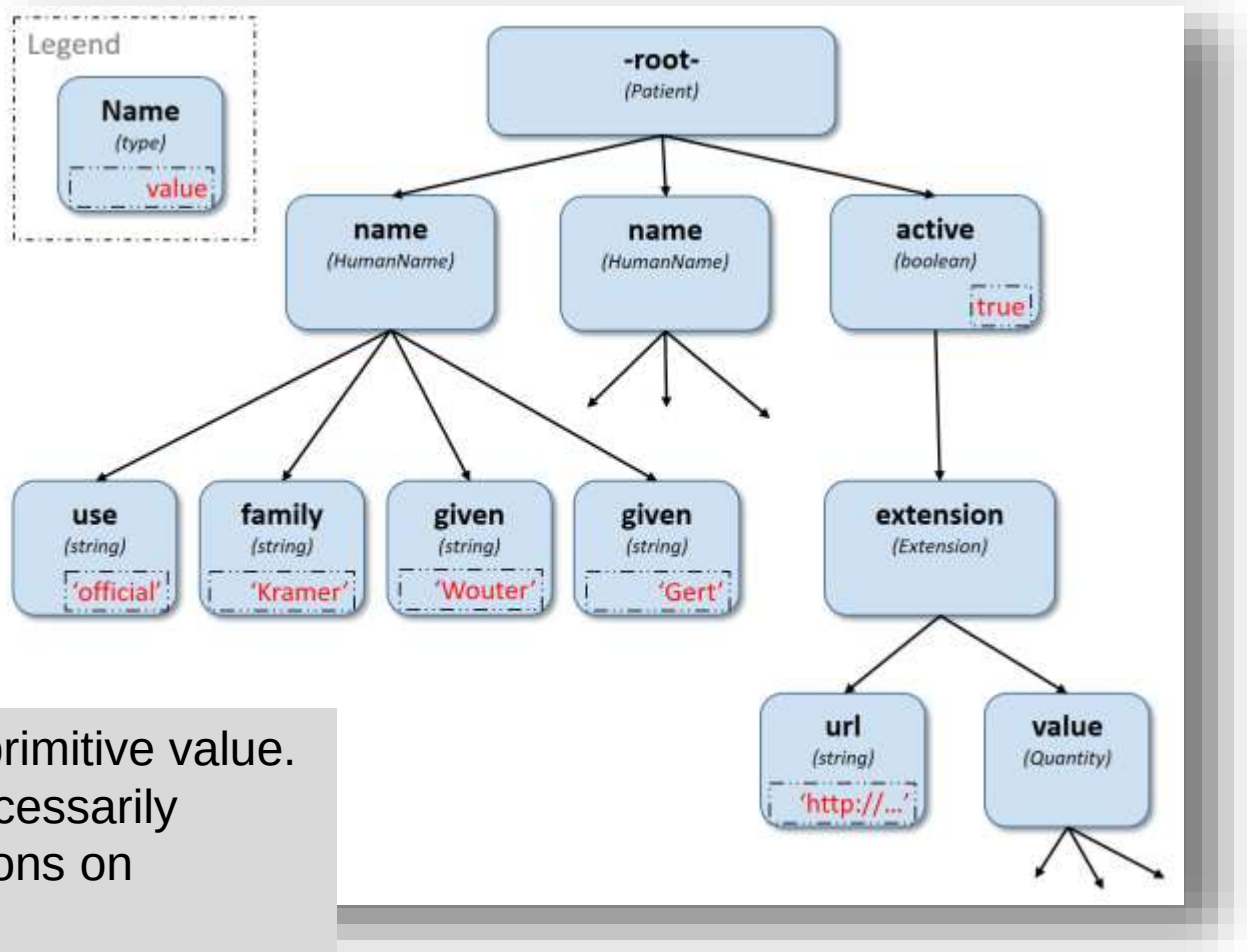
- We looked at .NET LINQ, Underscore.js, Haskell....

LINQ	Java Streams	Scala	Ruby	Underscore	Haskell	Clojure	JavaScript
Where	Filter	Filter	Select	Filter	Filter	Filter	Filter
Select	Map	Map	Collect	Map	Map	Map	Map
Aggregate	Reduce	Fold	Reduce	Fold	Fold?	Reduce	Reduce
Take	Limit	Take	Take	First(n)	Take	Take	
Skip	Substream	Drop	Drop	Rest(n)	Drop	Drop	Slice(n)
First	FindFirst	Head	First	First	Head	First	
Last		Last	Last	Last	Last	Last	

- Integrates in HL7 CQL (Clinical Quality Language)
- Binding to HL7 v2 Message
- So, it's not just "FHIR"Path....

OVERVIEW

Data as a Tree



Nodes MAY have a primitive value. Primitives are not necessarily leaves (think extensions on primitives!)

Navigating the tree

- By name (=direct child with a given name)
  - Patient.contained.name.given
- By (child) axis
  - Patient.contained.children()
  - Patient.identifier.descendants()
- Combinations
  - Patient.descendants().use

Datatypes

```
boolean: true, false
string: 'test string', 'urn:oid:3.4.5.6.7.8'
integer: 0, 45
decimal: 0.0, 3.1415926535897932 36
datetime: @2015-02-04T14:34:28Z (`@` followed by ISO8601 compliant date/time)
time: @T14:34:28+09:00 (`@` followed
quantity: 10 'mg', 4 days
```

FHIR primitive type	FHIRPath type
boolean	boolean
string, uri, code, oid, id, uuid, sid, markdown, base64Binary	string
integer, unsignedInt, positiveInt	integer
decimal	decimal
date, dateTime, instant	dateTime
time	time

OPERATORS AND FUNCTIONS

Filtering

- Takes a list, and produces a list
- where() function, with a predicate as an argument
- Patient.identifier.where(use='official')
- Implicit (lambda) argument "$this":
  - Patient.identifier.where($this.use='official')
  - Patient.name.given.where($this > 'aaa')
  - Patient.name.where(given > 'aaa')

Mapping

- Take a list and produce a list
- select() function, with a function that produces an item (or set of items)
- In FhirPath, called select()
  - Patient.contained[0].name.select(given & ' ' & family)

- Actually works as a “SelectMany”: if function results in a set, it’s flattened
  - Patient.name.select(descendants())
- Works if name is just a single value

The usual operators

- Boolean operators: ‘and’, ‘or’, ‘xor’, ‘implies’
- String manipulation:
  - substring, startsWith, endsWith, contains
  - matches(regex), replaceMatches(regex)
- *, +, -, /, div, mod, & (concat)

Everything is a set

- In fact, in FhirPath, everything is a set:
  - Patient.contained[0].name.**first()**.select(given & ' ' & family)
  - Patient.contained[0].name.count()
  - Patient.contained[0].name.**first()**.count()
  - 1.count()
- No special logic for sets/single instances
- Behind the scenes operators like ‘+’ work on two sets of one element (only)

WORKING WITH MISSING DATA

Missing data

```
Patient.name.given = 'Duck'
```

- What does this actually mean when there is no given name at all?
  - As an *invariant*, this means "Name must always be 'Duck'" => FALSE
  - As a *filter* in a query => empty set

Missing data (3)

- Functions and operators propagate null:
  - 3 + {} → {}, 4 < {} → {}, 4 = {} → {} (!!)
- This also means our boolean operators use 3-valued logic (just like SQL).
- E.g. for “OR” this means:

	true	false	**empty ({ })**
true	true	true	true
false	true	false	empty ({ })
empty ({ })	true	empty ({ })	empty ({ })

Missing data (2)

```
Patient.name.given = 'Duck'
```

- FhirPath propagates empty collections over its operators and functions, so in this case the result is....the “empty collection” (aka ‘{}’)
- Alternatives always returning a boolean:
  - Invariant: given.exists() and given.all(given = ‘Duck’)
  - Filter: Patient.name.where(given=‘Duck’)

TYPED DATA

Type filters

- Functions like Patient.descendants()
- Observation.value

...May result in sets with elements of different types

- Patient.descendants().ofType(HumanName) [previously called ‘as’] function
  - filters a collection on items of a given type
- Observation.value is Quantity or Observation.value is Ratio
  - returns whether an item is of a given type

Conversions

- Simple conversions: Patient.active.toString()

- Using “immediate if”:
  - Observation.value.select(

    iif($this is Quantity, value.toString() & unit, value.toString()))

SOME EXAMPLES

Inspiration from the spec

ValueSet.compose.include.concept.designation.use	Details of how a designation would be used.	Extensible	Designation Use
ValueSet.compose.include.filter.op	The kind of operation to perform as a part of a property based filter.	Required	FilterOperator

4.8.4.2 Constraints

- **vsd-1**: On ValueSet.compose.include: A value set include/exclude SHALL have a value set or a system (expression on ValueSet.compose.include: `valueSet.exists() or system.exists()`)
- **vsd-10**: On ValueSet.expansion.contains: Must have a system if a code is present (expression on

Implies....

6.5.4. implies

If the left operand evaluates to `true`, this operator returns the boolean evaluation of the right operand. If the left operand evaluates to `false`, this operator returns `true`. Otherwise, this operator returns `true` if the right operand evaluates to `true`, and the empty collection (`{ }`) otherwise.

	true	**false**	**empty ({ })**
`true`	`true`	`false`	empty (`{ }`)
`false`	`true`	`true`	`true`
empty (`{ }`)	`true`	empty (`{ }`)	empty (`{ }`)

Some examples...

- "If the mode is 'fixed', a code must be provided”

  (mode = 'fixed') implies code.exists()

- “If the resource is contained in another resource, it SHALL be referred to from elsewhere in the resource”

  contained.all('#'+id in %resource.descendants().reference)

More examples

- “Types must be unique by the combination of code and profile”

  type.select(code&profile&targetProfile).isDistinct()

- “enableWhen must contain either a 'answer' or a 'hasAnswer' element"

  hasAnswer.exists() xor answer.exists()

QUESTIONS?