HL7 FHIR DevDays 2017

Validation in .NET and Java

Ewout Kramer, Furore Health Informatics and James Agnew, University Health Network

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

Who am I?

- **Name:** James Agnew
- **Company:** UHN / Smile CDR
- **Background:**
  - Lead dev on the Java HAPI library
- jamesagnew@gmail.com, @jamesagnew

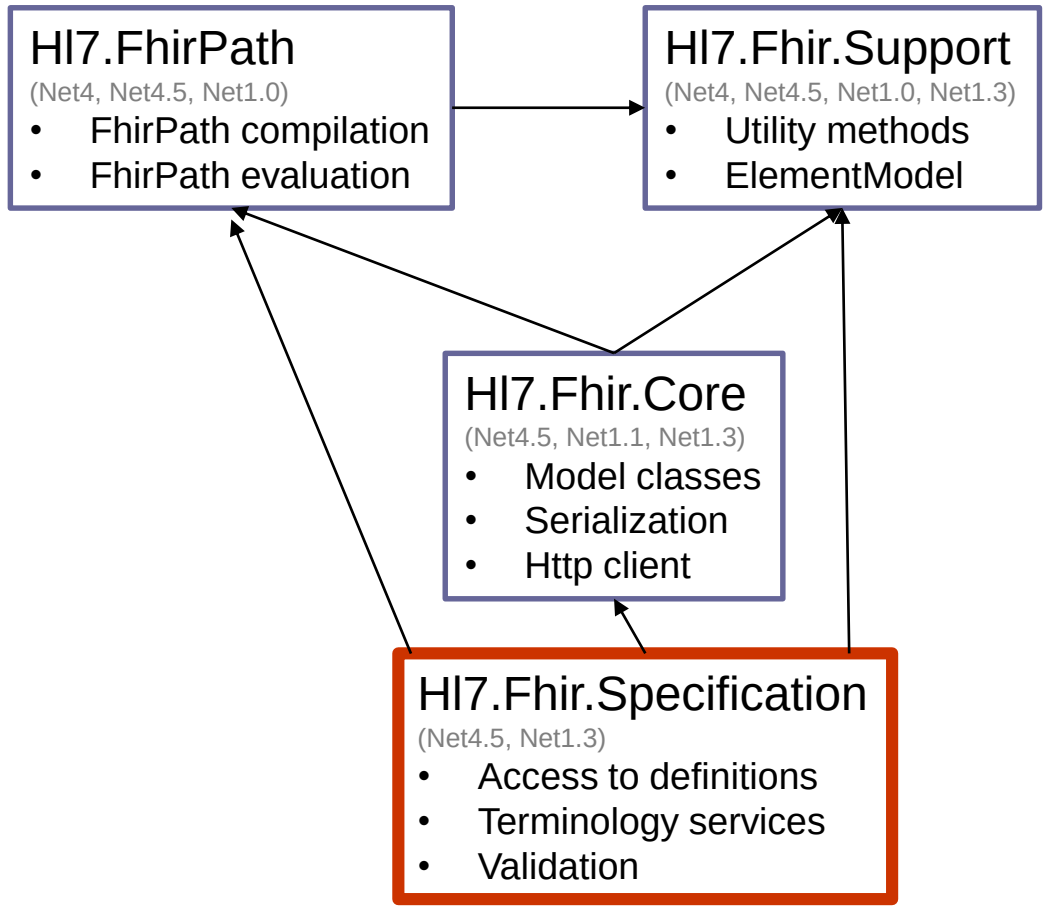
Validation inputs

- Instance data (resource)
- Definition of what data should look like (StructureDefinitions)
- Terminology services (ValueSet)
- Execution of FhirPath constraints

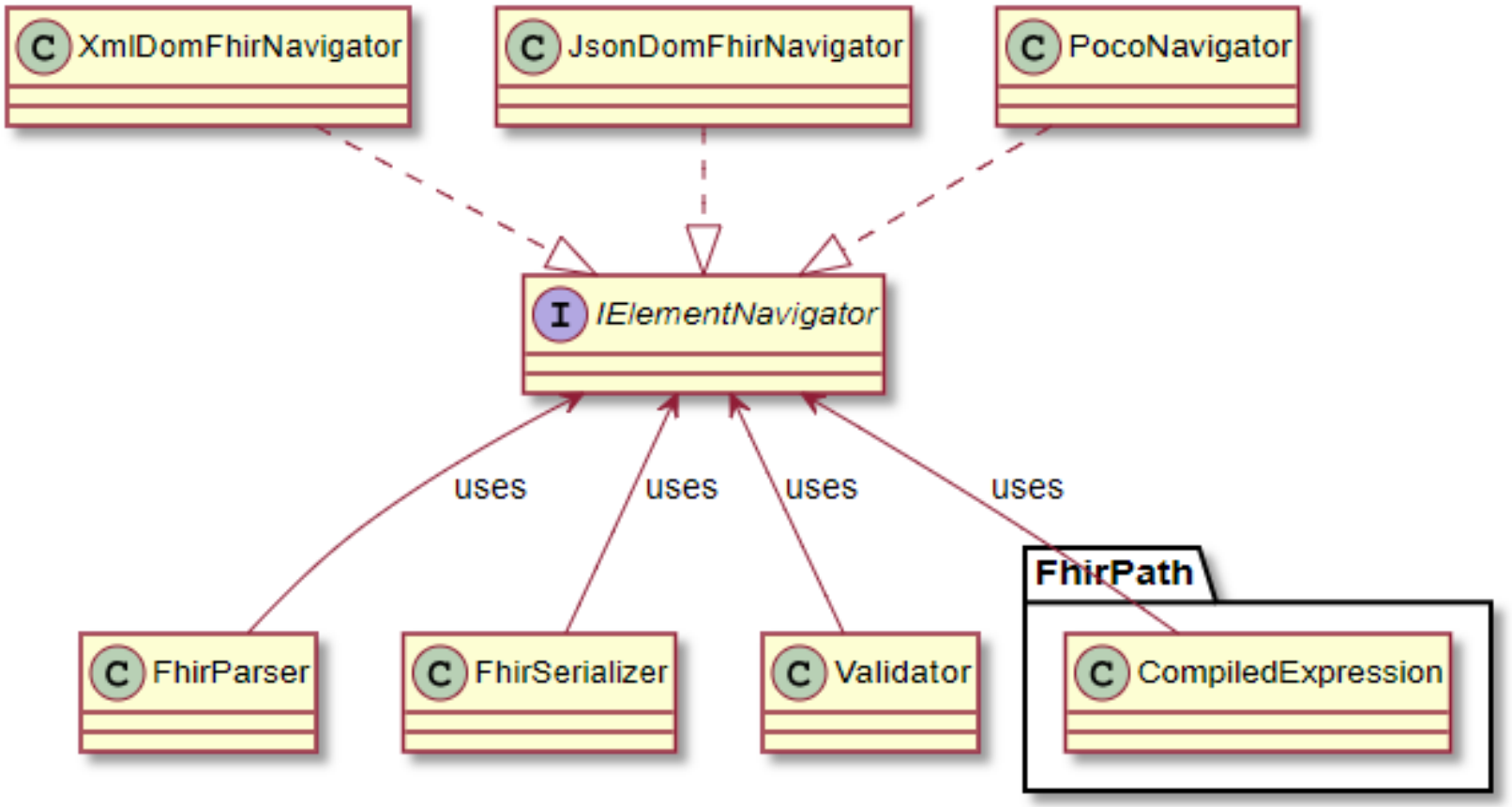
- Some way to combine it all, validate and return the outcomes

VALIDATION IN .NET

Structure of packages



Instance data: IElementNavigator



Specification data

- All the definitions of the core resources, types, search parameters, operations, etc. are available through the `Hl7.Specification.[STU3/DSTU2]` package on NuGet
- The package contains a zip (specification.zip) with meta data produced by the FHIR publication process
  - profiles-resources.xml, profiles-types.xml, extension-definitions.xml
  - search-parameters.xml
  - v2-tables.xml, v3-codesystems.xml, valuesets.xml
  - xsd schemas, schematrons, others....

Definitions: IResourceResolver

```
public interface IResourceResolver
{
    Resource ResolveByUri(string uri);
    Resource ResolveByCanonicalUri(string uri);
}
```

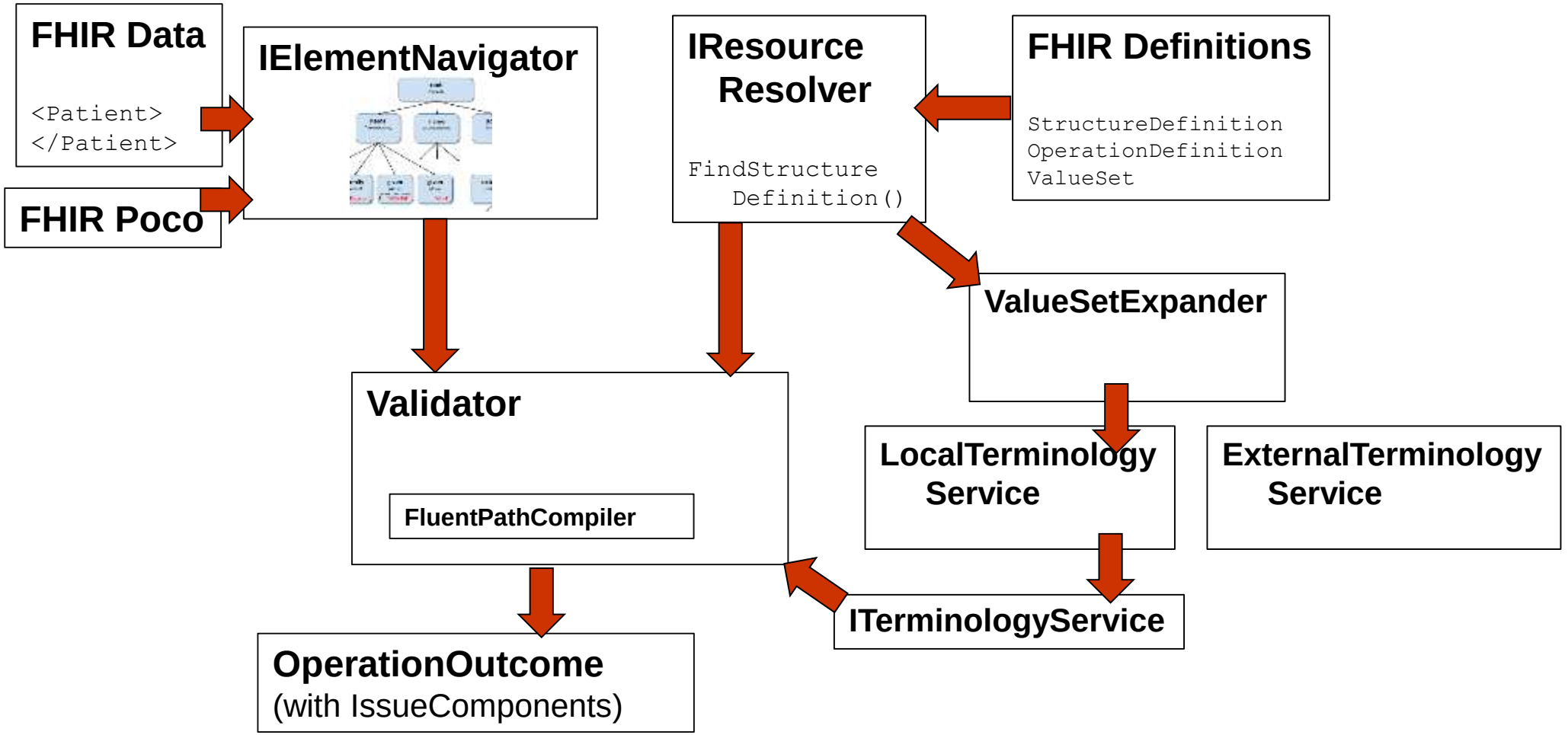
- Concrete implementations in API:
  - DirectorySource, ZipSource
  - WebSource
  - CachedResolver (wraps another resolver)
  - MultiResolver (tries a list of resolvers)

Terminology: ITerminologyService

- Current implementations:
  - `LocalTerminologyService`
    Does an in-memory expand & lookup
  - `ExternalTerminologyService`
    Uses the FhirClient calls to validate a code (possibly even sending your valueset across!)
  - `FallbackTerminologyService`
    First tries LocalTerminologyService, if that fails (too complex!), invoke an external service

- Note: returns `OperationOutcome`

The Big Picture



Practicalities

- `class Validator` in Hl7.Fhir.Validation namespace (Hl7.Fhir.Specification assembly)
- Configure: `new Validator(settings)`
  - Resolver, terminology service to use
- Validate: call one of the overloads:
  - `Validate(IElementNavigator)`
  - `Validate(Base)`
  - `Validate(XmlReader)`
- Result is an `OperationOutcome`

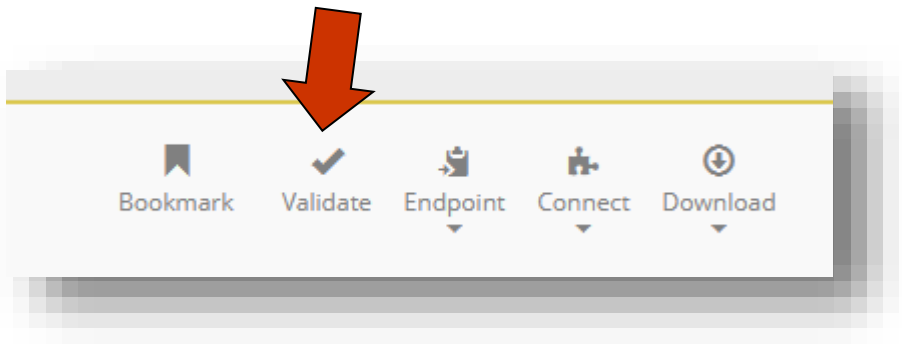
```
var source =new CachedResolver(new MultiResolver(
                            new DirectorySource(@"c:\data\MyProfiles"),
                            ZipSource.CreateValidationSource()  ));

var ctx = new ValidationSettings()
{
    ResourceResolver = Resolver,  GenerateSnapshot = true,
    EnableXsdValidation = true, Trace = false,  ResolveExteralReferences = true
};

var validator = new Validator(ctx);

var myPatient = new Patient { // data for your patient };

// Validates against the core profile - or others if present
// in patient.meta.profile
var results = validator.Validate(myPatient);
if(!results.Success) { // ouch }

// Presumably available in the c:\data\MyProfiles directory
result = validator.Validate(myPatient,
    "http://myprofiles.com/fhir/StructureDefinition/MyPatient");
```

Caveats

- Validator is still being improved
- Support most features already, except for slicing – only discriminator-less slicing is supported
- Other improvements: e.g. loop-detection

Playing with the validator

- Windows executable:
  - https://github.com/ewoutkramer/Furore.Fhir.ValidationDemo/releases
- Directly calling the Validator in code
- Using “Validate” on Simplifier.net



OperationOutcome - on steroids

- Lots of helper methods
- Properties:
  - Success (!information,!warning)
  - Fatals, Errors, Warnings properties
- Extension methods (in Hl7.Fhir.Support)
  - ErrorsAt(string path)
  - Where(severity, type, …)
  - Set/GetHierachyLevel()
  - Include(outcome), Add(outcome)

QUESTIONS?

Suggestions for a pleasurable afternoon

VALIDATION IN HAPI FHIR

When and how to Validate

There is no right answer to this question!

- Validate strictly during development but be loose in production? Always be strict?
- Validate some resource types but not others?
- Validate on the way in but not the way out?
- Structural vs semantic validation? (there is a performance cost!)

Validation in HAPI FHIR

Validator

(semantic validation)

- Applies a complete set of rules to a resource instance
- Currently far more powerful

Parser Error Handler

(structural validation only)

- Parser can be configured with an “Error Handler” which logs or fails on error
- Only catches structural issues

Parser Error Handler

- Callback mechanism during parsing
- Very fast!
- Catches structural problems:
  - Invalid elements
  - Invalid cardinalities
  - JSON data type mismatches

Parser Validator

```
String input = "{\"resourceType\":\"Patient\",\"isnice\":true}";

FhirContext ctx = FhirContext.forDstu3();
IParser parser = ctx.newJsonParser();

parser.setParserErrorHandler(new LenientErrorHandler(true));
parser.parseResource(input);
```

Parser Validator

```
String input = "{\"resourceType\":\"Patient\",\"isnice\":true}";

FhirContext ctx = FhirContext.forDstu3();
IParser parser = ctx.newJsonParser();

parser.setParserErrorHandler(new LenientErrorHandler(true));
parser.parseResource(input);
```

[main] WARN ca.uhn.fhir.parser.LenientErrorHandler - Unknown element 'isnice' found while parsing

Parser Validator

```
String input = "{\"resourceType\":\"Patient\",\"isnice\":true}";

FhirContext ctx = FhirContext.forDstu3();
IParser parser = ctx.newJsonParser();

parser.setParserErrorHandler(new StrictErrorHandler());
parser.parseResource(input);
```

Parser Validator

```
String input = "{\"resourceType\":\"Patient\",\"isnice\":true}";

FhirContext ctx = FhirContext.forDstu3();
IParser parser = ctx.newJsonParser();

parser.setParserErrorHandler(new StrictErrorHandler());
parser.parseResource(input);
```

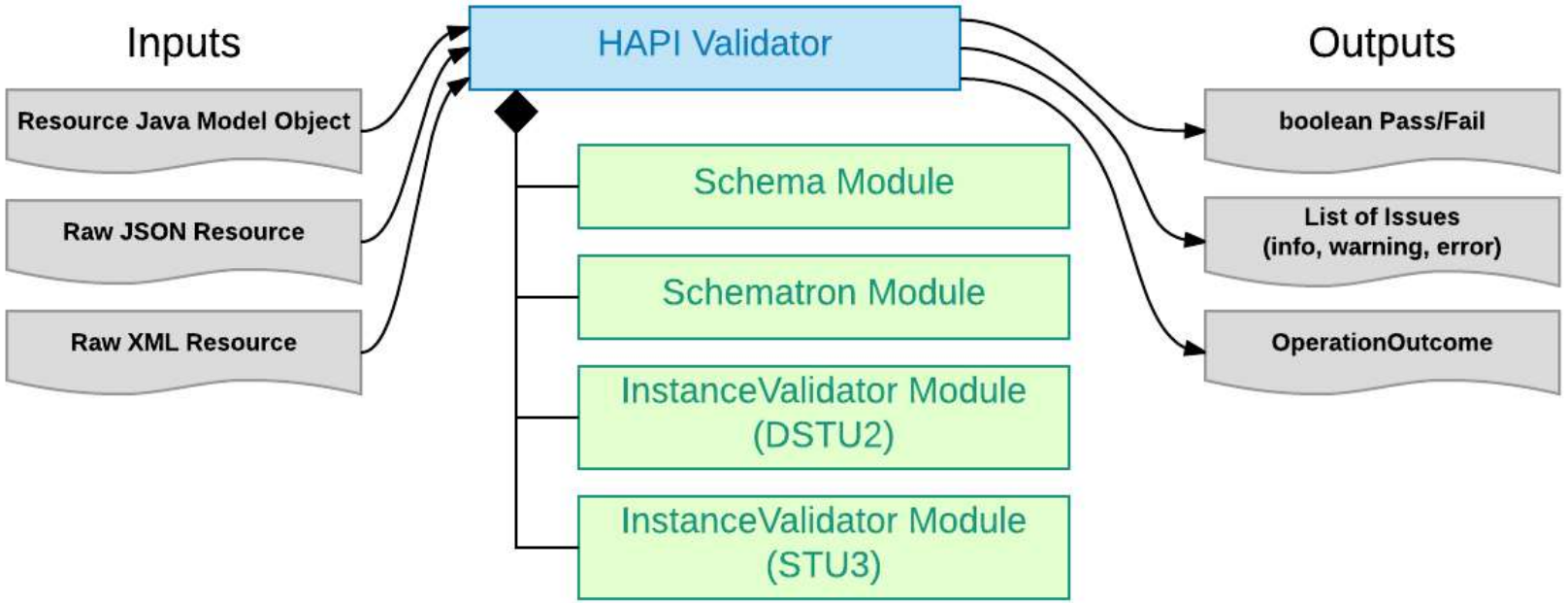
ca.uhn.fhir.parser.DataFormatException:
Unknown element 'isnice' found during parse

Profile Validation

- Applies semantic validation based on “Profiles”
- Slower than parser error handler!
- Validates various things:
  - All the same structural validation as parser error handler
  - Terminology bindings
  - Invariants
  - Constraints and extensions

Validator

- HAPI's validator uses modules and collects the results from any that are enabled
- Create your own if you want!



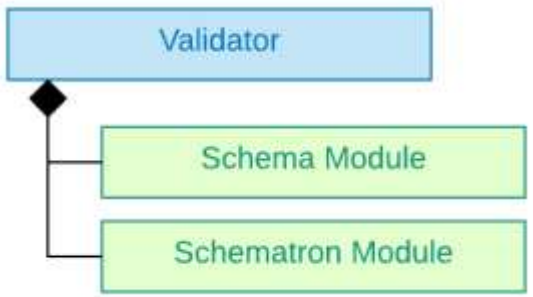
Schema/Schematron Validator

- FHIR provides schemas + schematrons for validating conformance to basic resource requirements:
  - Cardinality
  - Optionality
  - Required bindings
  - Datatype rules
- The Schema + Schematron modules are the defaults if nothing else is selected

Schema/Schematron Validator (2)

Dependencies:

- hapi-fhir-validation-resources-dstu3-2.5.jar
- phloc-schematron-2.7.0.jar



Inputs (pick one):

- Resource Java model object
- Raw JSON Resource
- Raw XML Resource

Using the Schema Validator


```
public class Example20_ValidateResource {
 public static void main(String[] args) {

  // Create an incomplete encounter (status is required)
  Encounter enc = new Encounter();
  enc.addIdentifier().setSystem("http://acme.org/encNums").setValue("12345");

  // Create a new validator
  FhirContext ctx = FhirContext.forDstu3();
  FhirValidator validator = ctx.newValidator();

  // Did we succeed?
  ValidationResult result = validator.validateWithResult(enc);
  System.out.println("Success: " + result.isSuccessful());

  // What was the result
  OperationOutcome outcome = (OperationOutcome) result.toOperationOutcome();
  IParser parser = ctx.newXmlParser().setPrettyPrint(true);
  System.out.println(parser.encodeResourceToString(outcome));
 }
}
```

Using the Sc

```
public class Example20_ValidateResource {
 public static void main(String[] args) {

  // Create an incomplete encounter (status i
  Encounter enc = new Encounter();
  enc.addIdentifier().setSystem("http://acme

  // Create a new validator
  FhirContext ctx = FhirCont                    );
  FhirValidator validator = ct                  or();

  // Did we succeed?
  ValidationResult result = validator.validateWithResult(enc);
  System.out.println("Success: " + result.isSuccessful());

  // What was the result
  OperationOutcome outcome = (OperationOutcome) result.toOperationOutcome();
  IParser parser = ctx.newXmlParser().setPrettyPrint(true);
  System.out.println(parser.encodeResourceToString(outcome));
 }
}
```

false

```
<OperationOutcome xmlns="http://hl7.org/fhir">
  <issue>
        <severity value="error"/>
        <code value="processing"/>
        <diagnostics value="cvc-complex-type.2.4.b: The content of
element 'Encounter' is not complete. One of
'{&amp;quot;http://hl7.org/fhir&amp;quot;:identifier,
&amp;quot;http://hl7.org/fhir&amp;quot;:status}' is expected."/>
        <location value="Line[1] Col[140]"/>
  </issue>
</OperationOutcome>
```

Schema Validation with String Input

(JSON is converted to XML!)

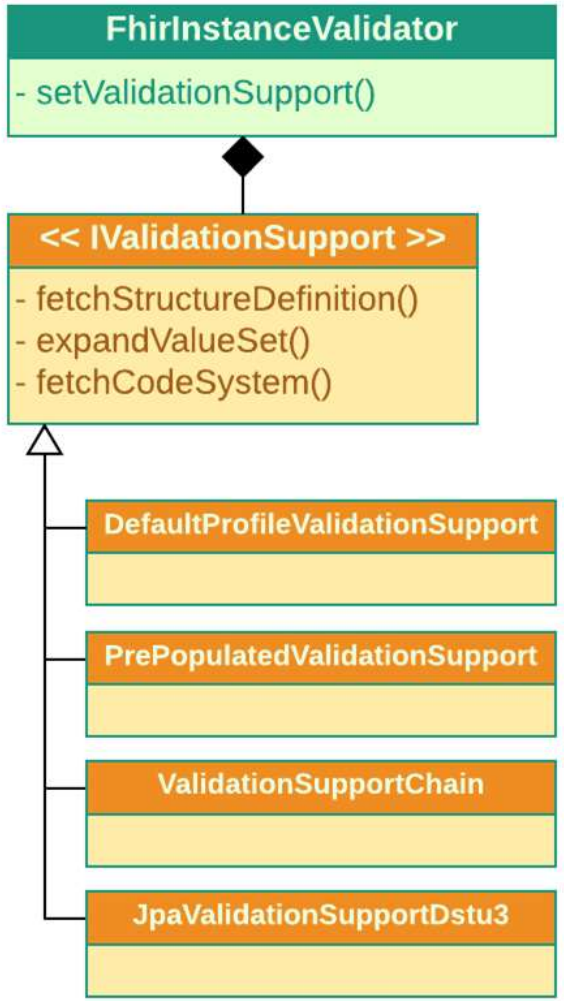
```
public class Example21_ValidateResourceString {
 public static void main(String[] args) {

  String input = "<Encounter xmlns=\"http://hl7.org/fhir\"></Encounter>";

  // Create a new validator
  FhirContext ctx = FhirContext.forDstu3();
  FhirValidator validator = ctx.newValidator();

  // Did we succeed?
  ValidationResult result = validator.validateWithResult(input);
  System.out.println("Success: " + result.isSuccessful());

  // What was the result
  OperationOutcome outcome = (OperationOutcome) result.toOperationOutcome();
  IParser parser = ctx.newXmlParser().setPrettyPrint(true);
  System.out.println(parser.encodeResourceToString(outcome));
 }
}
```

Profile Validation

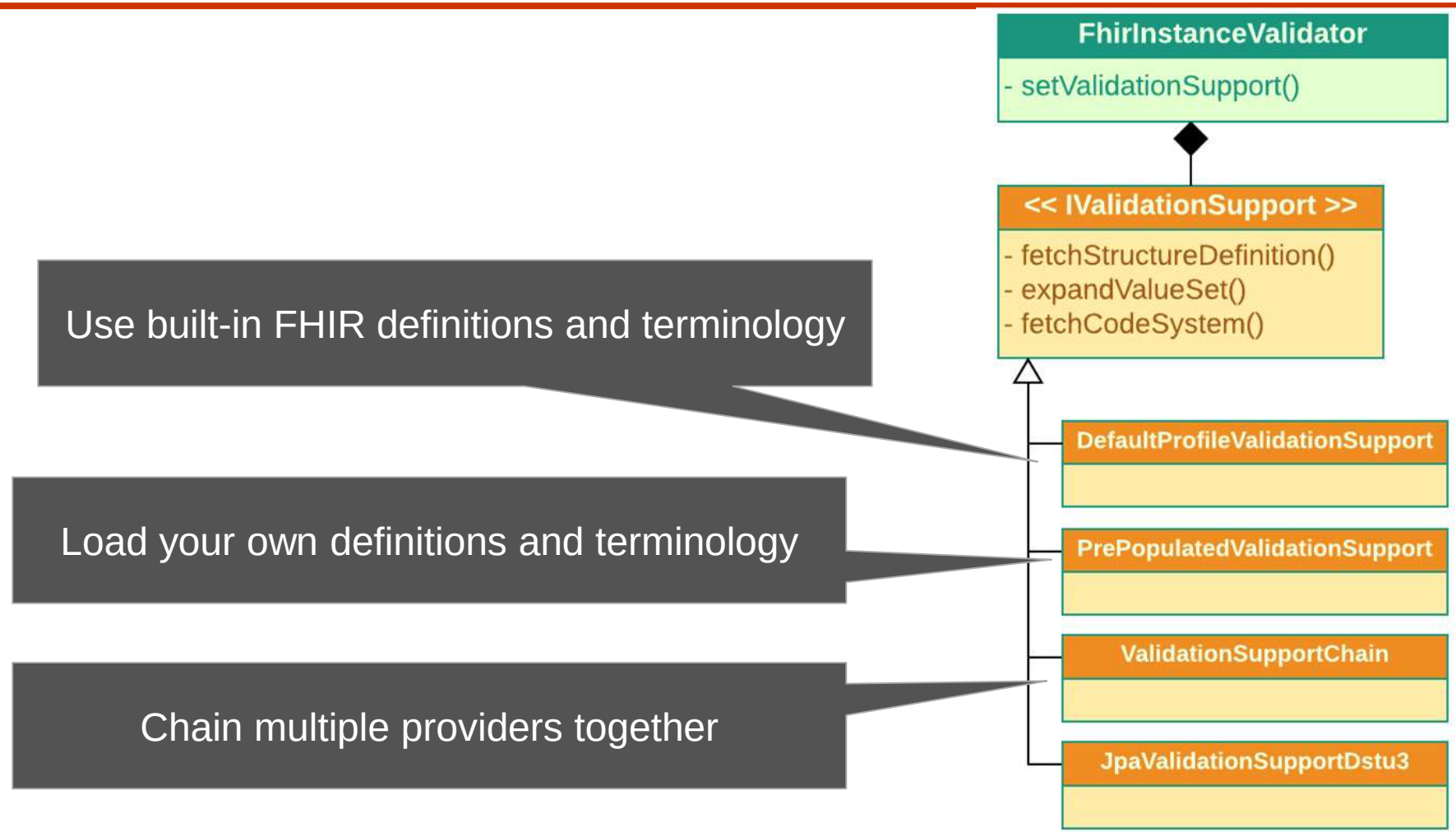
- In FHIR, a “Profile” is a collection of special resources:
  - StructureDefinition
  - ValueSet
  - CodeSystem
- A resource can declare conformance to a profile in its metadata
- A server can require conformance to a profile

Profile Validator

- FhirInstanceValidator module requires an instance of IValidationSupport
- Several implementations are supplied with HAPI FHIR
- You can also create your own

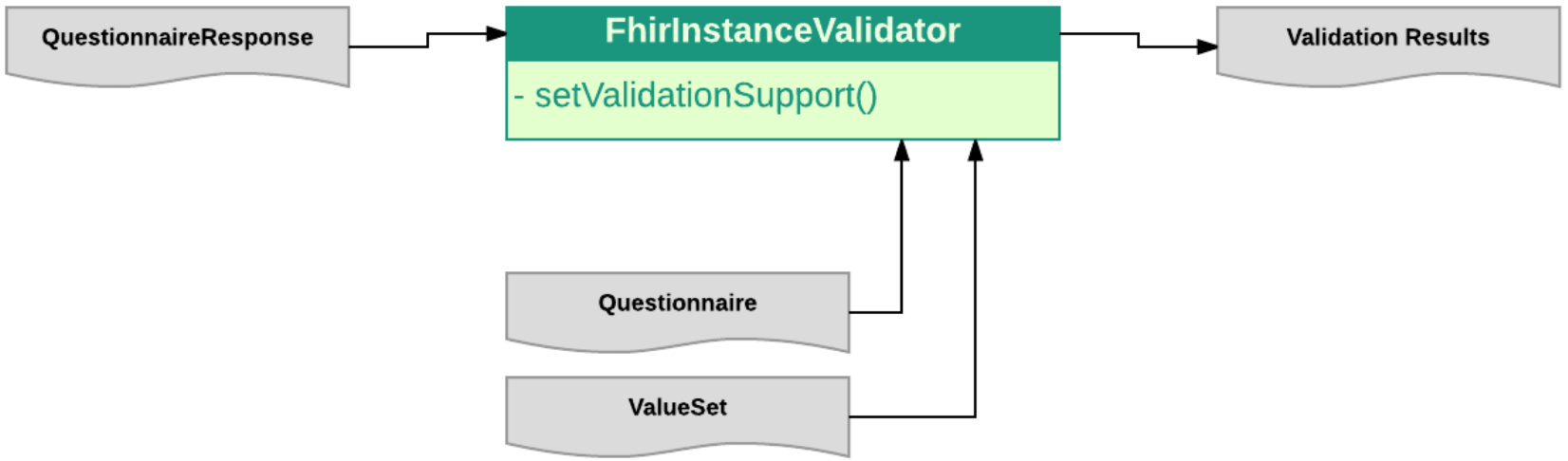


Validation Support



Questionnaires are Validated

- FhirInstanceValidator will also validate QuestionnaireResponse compliance to Questionnaire



Using the Profile Validator

```
public class Example22_ValidateResourceInstanceValidator {
 public static void main(String[] args) {
   // Create an incomplete encounter (status is required)
   Encounter enc = new Encounter();
   enc.addIdentifier().setSystem("http://acme.org/encNums").setValue("12345");

   // Create a new validator
   FhirValidator validator = FhirContext.forDstu3().newValidator();

   // Cache this object! Supplies structure definitions
   DefaultProfileValidationSupport support = new DefaultProfileValidationSupport();

   // Create the validator
   FhirInstanceValidator module = new FhirInstanceValidator(support);
   validator.registerValidatorModule(module);

   // Did we succeed?
   IParser parser = FhirContext.forDstu3().newXmlParser().setPrettyPrint(true);
   System.out.println(parser.encodeResourceToString(validator.validateWithResult(enc).toOperationOutcome()));
 }
}
```

Using the Profile Validator


```
public class Example22_ValidateResourceInstanceValidator {
 public static void main(String[]
   // Create an incomplete encou
   Encounter enc = new Encoun
   enc.addIdentifier().setSystem

   // Create a new validator
   FhirValidator validator = FhirC

   // Cache this object! Supplies
   DefaultProfileValidationSuppo

   // Create the validator
   FhirInstanceValidator module
   validator.registerValidatorModule(module);

   // Did we succeed?
   IParser parser = FhirContext.forDstu3().newXmlParser().setPrettyPrint(true);
   System.out.println(parser.encodeResourceToString(validator.validateWithResult(enc).toOperationOutcome()));
 }
}
```

```
<OperationOutcome xmlns="http://hl7.org/fhir">
  <issue>
        <severity value="error"/>
        <code value="processing"/>
        <diagnostics value="Profile
http://hl7.org/fhir/StructureDefinition/Encounter, Element
'Encounter.status': minimum required = 1, but only found 0"/>
        <location value="Encounter"/>
  </issue>
</OperationOutcome>
```

Suggestions for a pleasurable afternoon

HANDS-ON TRACK

Play with the validator

- Check out https://github.com/ewoutkramer/Furore.Fhir.ValidationDemo & compile
- Try to incorporate your resolver and terminology service (if any)
- Try running the examples at https://github.com/ewoutkramer/fhir-net-api/tree/develop/src/Hl7.Fhir.Specification.Tests/TestData/validation
- Alter the examples so they trigger your code (change a binding…)