

**MODUL PRAKTIKUM  
STANDARISASI DAN INTEROPERABILITAS**



**Disusun oleh  
Tim Dosen**



## Argonaut Profiles

- FHIR allows a large amount of variability between systems
- Most EHR applications require very little variability in order to work correctly
- To address this fact, US implementors have created a set of profiles for this purpose in the US called the “Argonaut Profiles”

Argonaut provides constraints on a set of FHIR types commonly used for EHR Applications:

- Patient
- Condition
- Observation, DiagnosticReport
- Medication, MedicationStatement, MedicationOrder
- AllergyIntolerance
- Immunization
- CarePlan
- Goal

Each resource is constrained for use by an EHR Application. For example:

- Patient must have a name, gender, date of birth, and at least 1 identifier
- Extensions are specified for race and ethnicity
- A communication language should be specified

These requirements may not be perfect for Vietnam, but they are a great starting point

## FHIR Testing : Technical and community

### FHIR Testing

One of the best parts of working with FHIR is the existence of great test servers

- The following servers are available for free for testing around the world (and there are many more):

Grahame’s Server:

<http://test.fhir.org/r3>

James’s Server:

<http://hapi.fhir.org/baseDstu3>

### FHIR Community

Another great thing about FHIR is the large, helpful, international community.

- The focal point of this community is chat.fhir.org (Zulip)
- We have created a “stream” in Zulip for Vietnam, but there are many others as well

<http://chat.fhir.org>

#### Community Projects

FHIR also has a very large open source community devoted to helping implementors on various platforms:

- Java: HAPI FHIR (We will cover this tomorrow)
- .NET / C# API: <https://github.com/ewoutkramer/fhir-net-api>
- JavaScript FHIR.js: <https://github.com/FHIR/fhir.js>
- Python Client: <https://github.com/smart-on-fhir/client-py>
- iOS / Swift: <https://github.com/smart-on-fhir/Swift-FHIR>
- Android / Java: <https://github.com/jamesagnew/hapi-fhir>
- Pascal: <https://github.com/grahamegrieve/fhirserver>

#### Testing

- FHIR defines a special resource called TestScript which can be used to specify client and server tests
- There are currently two platforms for executing these tests:
  - Crucible (free tool): <https://projectcrucible.org/>
  - Touchstone (paid tool): <http://touchstone.com>

#### Community Starter Projects

The following link has a collection of starter projects in various languages:

<https://github.com/furore-fhir/fhirstarters>

#### Validation

- You are only interoperable if you can produce valid FHIR
- There are several kinds of valid:
  - Valid JSON / XML
  - Valid FHIR
  - Valid FHIR for a specific purpose
- FHIR servers define an endpoint called /\$validate which can be used to validate FHIR payloads, e.g.

POST /base/Patient/\$validate

Content-Type: application/fhir+json

```
{  
  "resourceType": "Patient"  
  "name": [ ]  
}
```

## Profiles

- To make FHIR useful in a specific context, we often want to create Profiles
  - E.g. "In my system, Observations will use LOINC codes"
- FHIR defines a special set of resources which may be used to constrain FHIR for a specific use:
  - **StructureDefinition**: Set field cardinality, add terminology binding, add extensions
  - **CodeSystem & ValueSet**: Define sets of codes for a given purpose
- Tools exist to validate against a Profile (we will cover HAPI on Thursday)

## FHIR Versions

### Versions

- The FHIR specification itself has had several releases:
  - FHIR DSTU1 (v0.0.82) - 2014
  - FHIR DSTU2 (v1.0.2) - 2015
  - FHIR STU3 - (v3.0.1) - 2017
  - FHIR R4 - Under development
- The version names mean slightly different things but people often use them interchangeably ( DSTU3 / STU3 / R3 )

## Vietnam Affiliate and Implementation Guide

### Making FHIR work for you

- International Specification defines overall framework
- Countries / Regions / Vendors / Institutions publish adaptations to local culture/regulations etc
- Individual projects use conformance resources to describe the project rules
  - Terminology usage rules
  - Rules about elements, usage, content flows
  - Extensions
- All of this can be published through <http://registry.fhir.org>

### Example National Profiles

- US: <http://www.hl7.org/fhir/us/core/>
- Australia: <http://build.fhir.org/ig/hl7au/au-fhir-base/>
- Concerns:
  - National identifiers
  - National code systems
  - Specific additional patient information (race/ethnicity)
  - Basic Documentation, Community Governance

### Candidate National Vietnamese IG

- Found at: <http://build.fhir.org/ig/grahamegrieve/vietnam-poc/index.html>
- Demonstration of the production of this

### HL7 Affiliate for Vietnam

- Some formal organization needs to manage the vietnamese national implementation guide
- Needs to be connected to HL7
- Best to be an HL7 Affiliate
- Registration in process

### Affiliate Requirements

- Must be an NGO
- Must accept any Vietnamese organization as a member
- Must have a constitution with leadership elected by members
- Can work very closely with Department of Health

### Welcome to the First FHIR Connectathon In Vietnam

#### Connectathon Goals

The FHIR Connectathon is an event for implementers. It is held 3 times each year by HL7, and often in other contexts like this one.

We have 2 equally important goals:

- Helping implementers learn to use the FHIR specification
- Helping to develop the FHIR specification

(We often use Connectathons to try new ideas too!)

## Where To Start

If you are a beginner:

- Do the Postman tutorial with James: <https://goo.gl/5a5RQg>

If you have your own application:

- Enter your details into the spreadsheet: <http://tiny.cc/tu15oy>

Advanced features:

- Security, Mapping existing data,

## FHIR and HAPI FHIR

### The FHIR Data Model

- FHIR's model is available online
  - <http://hl7.org/fhir/>
- The FHIR data model is useful even by itself
- Take advantage of the collective work of 100s of people!

### Data Types: Primitives

## Data Types: Primitives

- **string** Patient is awake
- **boolean** true
- **date** 2016-02-19
- **decimal** 12.347000
- **integer** 500
- **uri** http://snomed.info/sct
- **base64** rwr39o9h=
  
- **dateTime** 2015-01-26T15:33-05:00
- **instant** 2015-01-26T15:33:13.0-05:00
  
- **markdown** **\*\*woohoo\*\***

Activate Windows  
Go to Settings to activate Windows.

# Data Types: Primitives

## ISO8601 - Human Times: Timezone is mandatory

- date 2016-02-19
- decimal 12.347000
- integer 500
- uri http://snomed.info/sct
- dateTime 2015-01-26T15:33-05:00
- instant 2015-01-26T15:33:13-05:00
- markdown \*\*woohoo\*\*

Activate Windows  
Go to Settings to activate Windows.

119

Esa Unggul

Esa Unggul

Esa Unggul

# Data Types: Primitives

## System Times, fixed precision

- string
- boolean
- date 2016-02-19
- decimal 12.347000
- integer 500
- uri http://snomed.info/sct
- dateTime 2015-01-26T15:33-05:00
- instant 2015-01-26T15:33:13.0-05:00
- markdown \*\*woohoo\*\*

Activate Windows  
Go to Settings to activate Windows.

120

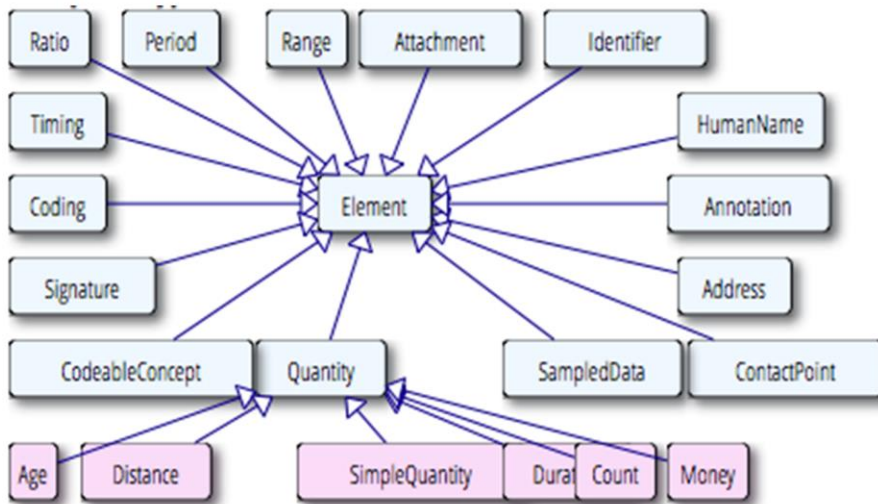
Esa Unggul

Esa Unggul

Esa Unggul



# Data Types: Composites



Activate Windows  
Go to Settings to activate Windows.

121

# Data Types: Composites

Name	Flags	Card.	Type	Description & Constraints
Address	Σ		Element	An address expressed using postal conventions (as opposed to Elements defined in Ancestors: <a href="#">id</a> , <a href="#">extension</a> )
use	?! Σ	0..1	code	home   work   temp   old - purpose of this address <a href="#">AddressUse</a> (Required)
type	Σ	0..1	code	postal   physical   both <a href="#">AddressType</a> (Required)
text	Σ	0..1	string	Text representation of the address
line	Σ	0..*	string	Street name, number, direction & P.O. Box etc. This repeating element order: The order in which lines should :
city	Σ	0..1	string	Name of city, town etc.
district	Σ	0..1	string	District name (aka county)
state	Σ	0..1	string	Sub-unit of country (abbreviations ok)
postalCode	Σ	0..1	string	Postal code for area
country	Σ	0..1	string	Country (e.g. can be ISO 3166 2 or 3 letter code)
period	Σ	0..1	Period	Time period when address was/is in use

Activate Windows  
Go to Settings to activate Windows.

122

## Other Model Concepts: Identifiers

- FHIR resources are scoped around identifiable things (Patients, Orders, Locations, etc.)
- Identifiers consist of a **System** and an **Identifier**
- For example:
  - System (URI): <http://uhn.ca/ns/mrn>



- Identifier: 7000135
- Other systems:
  - <http://hl7.org/fhir/sid/us-ssn> (US SSN)
  - <urn:oid:2.16.840.1.113883.4.3.1> (Alabama Driver's License)

### Identifier Systems

- Old identifiers are sometimes OIDs, example: 0.1.2.3.4.5
- New identifiers are URLs
- Creating your own is fine!

<http://hospital.vn/patient>

### Other Model Concepts: Coded Values

- Many things are drawn from a set of allowable coded values
- A coded value consists of a **Code System** and a **Code**, and optionally a **Display Text**
- For example:
  - System: <http://snomed.info/sct>
  - Code: 267038008
  - Display: Edema (finding)

## Resource Identities

- Every FHIR resource has a unique identity, which is in fact a URL



Activate Windows  
Go to Settings to activate Windows.

# Resource Identities

- Resources can also have a version specific ID



Activate Windows  
Go to Settings to activate Windows.

## The Bundle Resource

- Sometimes we need to package multiple resources together
- We use a special container resource called "Bundle"

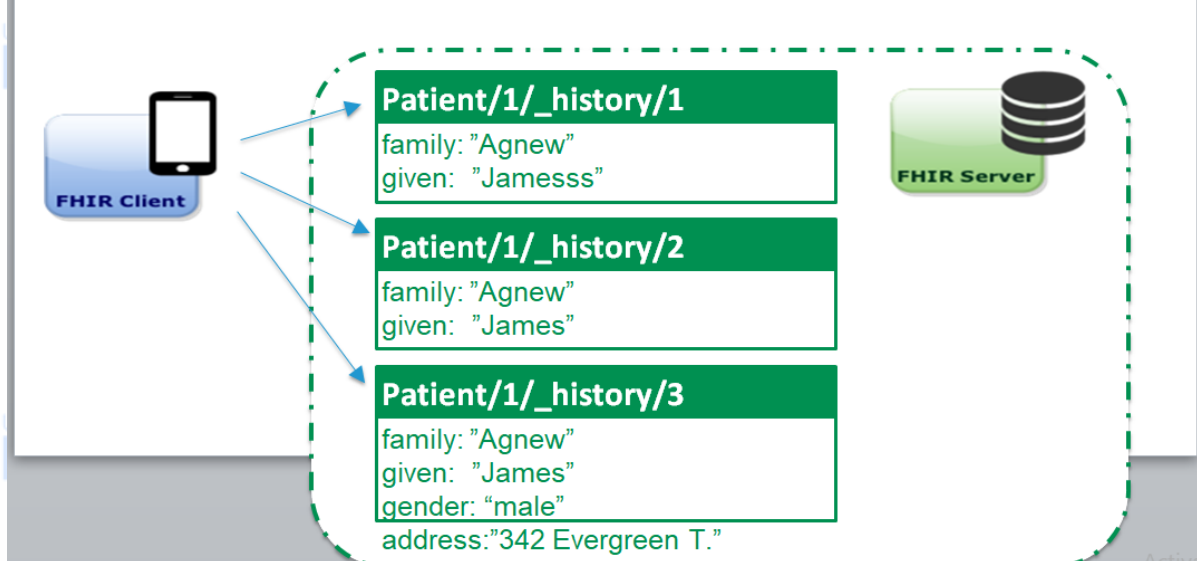
Bundle

Metadata
Patient/242 name: "John Smith"
Patient/105 name: "Frank Higgins"
Patient/3998 name: "Ross Frinz"
Patient/15 name: "Some Name" birthDate: "1999-01-02"



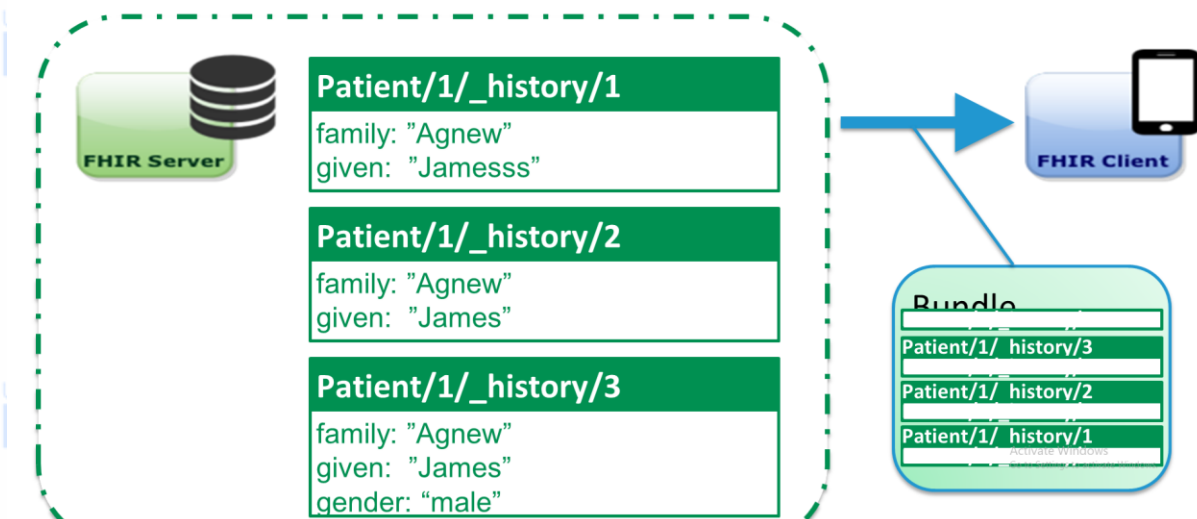
# Resource History

- FHIR servers keep track of the complete history of a given resource instance
- This way, the client doesn't need to keep track



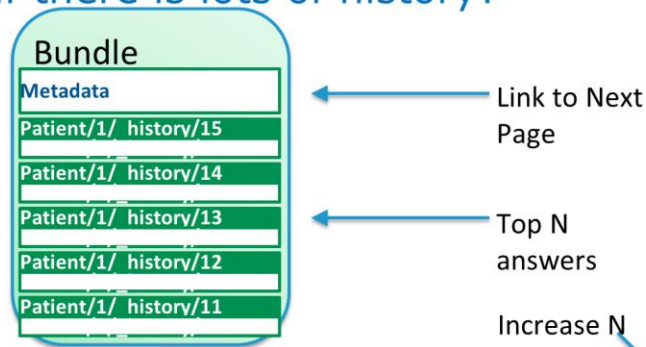
## The History Operation

[http://fhirtest.uhn.ca/baseDstu2/Patient/1/\\_history](http://fhirtest.uhn.ca/baseDstu2/Patient/1/_history)



# The History Operation: Paging

- What if there is lots of history?



[http://fhirtest.uhn.ca/baseDstu2/Patient/1/\\_history](http://fhirtest.uhn.ca/baseDstu2/Patient/1/_history)

[http://fhirtest.uhn.ca/baseDstu2/Patient/1/\\_history?\\_count=100](http://fhirtest.uhn.ca/baseDstu2/Patient/1/_history?_count=100)

# The History Operation: Modes

- There are 3 kinds of history
  - Server [http://fhirtest.uhn.ca/baseDstu2/\\_history](http://fhirtest.uhn.ca/baseDstu2/_history)  
(all resources)
  - Type [http://fhirtest.uhn.ca/baseDstu2/Patient/\\_history](http://fhirtest.uhn.ca/baseDstu2/Patient/_history)  
(same type)
  - Instance [http://fhirtest.uhn.ca/baseDstu2/Patient/1/\\_history](http://fhirtest.uhn.ca/baseDstu2/Patient/1/_history)  
(same ID)
- History can be used as a simple polling mechanism for subscription

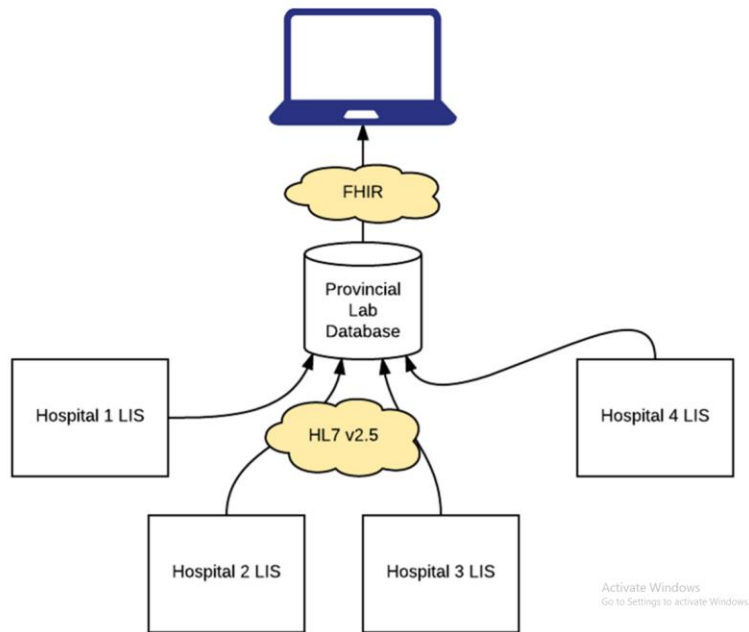
[http://fhirtest.uhn.ca/baseDstu2/Patient/1/\\_history?](http://fhirtest.uhn.ca/baseDstu2/Patient/1/_history?)

[\\_since=2011-02-23T15:00:01.0032-05:00](http://fhirtest.uhn.ca/baseDstu2/Patient/1/_history?_since=2011-02-23T15:00:01.0032-05:00)

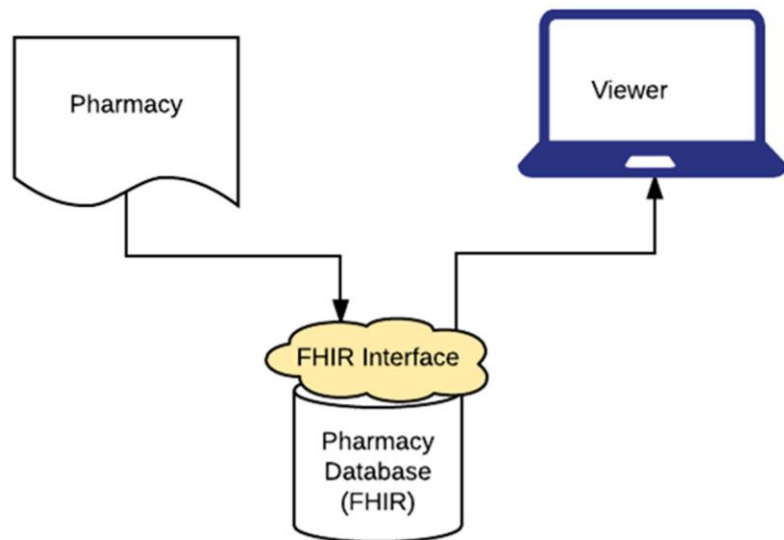


Examples

## National Lab System



## National Pharmacy System



HAPI

- HAPI FHIR Beginner? Intermediate? Expert?
- Using FHIR today?

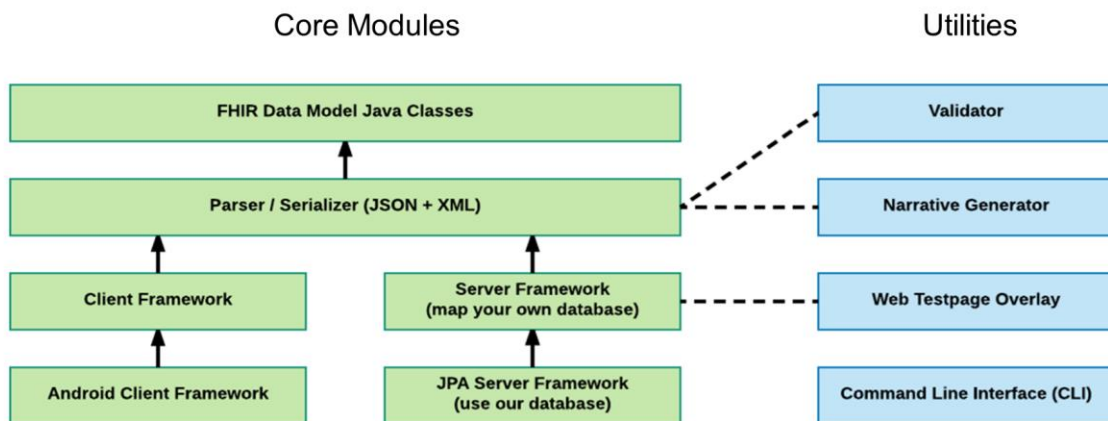
Design Goals

- Use Anywhere

- Apache 2.0 License for all components
- Minimal dependencies
- Be Flexible
  - Loosely coupled, pluggable components
- Be Powerful
  - “Borrow” all the best ideas from existing frameworks: JAX-WS, Springframework, .NET FHIR API ☺

..etc..

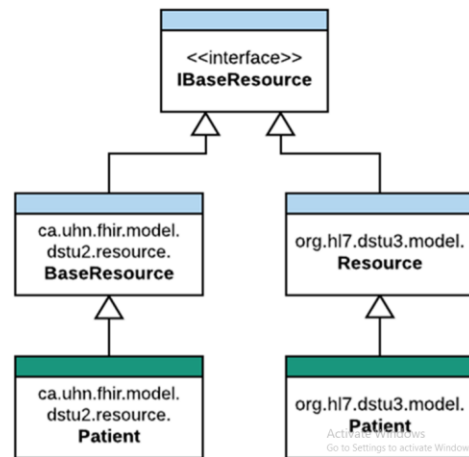
## HAPI FHIR Modules



Activate Windows  
Go to Settings to activate Windows.

## Structure Classes: Resources

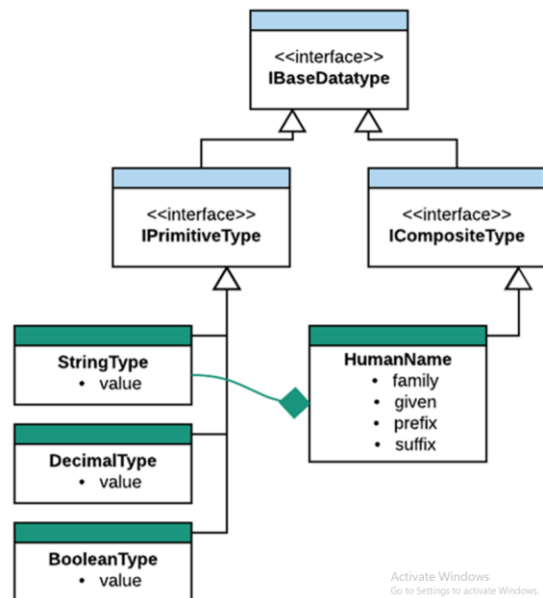
- HAPI Defines several sets of classes which form the data model
- Resource definition classes implement `IBaseResource`
- Examples: Patient, CarePlan, Encounter, Practitioner, Medication



141

## Structure Classes: Datatypes

- HAPI also defines a class for each data type
- Primitive classes are named `[name] Type`
- Primitive types include: `StringType`, `BooleanType`
- Composite types include: `Address`, `Ratio`, `HumanName`



142

Structure Classes:

### Docs

- JavaDocs for structures are available here:  
<http://hapifhir.io/apidocs-dstu2/index.html>  
<http://hapifhir.io/apidocs-dstu3/index.html>

### Creating A Resource

```
public class Example01_CreateAPatient {
    public static void main(String[] theArgs) {
        // Create a resource instance
        Patient pat = new Patient();
        // Add a "name" element
        HumanName name = pat.addName();
        name.setFamily("Simpson").addGiven("Homer").addGiven("J");
        // Add an "identifier" element
        Identifier identifier = pat.addIdentifier();
        identifier.setSystem("http://acme.org/MRNs").setValue("7000135");
        // Model is designed to be chained
        pat.addIdentifier().setSystem("http://acme.org/MRNs").setValue("12345");
    }
}
```

### Use your IDE Autocomplete

```
public class Example01_CreateAPatient {
    public static void main(String[] theArgs) {
        // Create a resource instance
        Patient pat = new Patient();
        // Add a "name" element
        HumanName name = pat.addName();
        name.setFamily("Simpson").addGiven("Homer").addGiven("J");
        // Add an "identifier" element
        Identifier identifier = pat.addIdentifier();
        identifier.setSystem("http://acme.org/MRNs").setValue("7000135");
        // Model is designed to be chained
        pat.addIdentifier().setSystem("http://acme.org/MRNs").setValue("12345");
    }
}
```



```
}  
}
```

```
addName () HumanName  
addName (HumanName t) Patient  
addAddress () Address  
addAddress (Address t) Patient  
addChild (String name) Base  
addCommunication () PatientCommunicationComponent  
addCommunication (PatientCommunicationComponent... Patient  
addContact () ContactComponent  
addContact (ContactComponent t) Patient  
addGeneralPractitioner () Reference  
addGeneralPractitioner (Reference t) Patient  
Did you know that Quick Documentation View (F2) works in completion lookups as well? >> .π
```

### Enumerated Types

```
public class Example02_EnumeratedTypes {  
    public static void main(String[] theArgs) {  
        Patient pat = new Patient();  
  
        pat.addName().setFamily("Simpson").addGiven("Homer").addGiven("J");  
  
        pat.addIdentifier().setSystem("http://acme.org/MRNs").setValue("7000135");  
  
        // Enumerated types are provided for many coded elements  
  
        ContactPoint contact = pat.addTelecom();  
  
        contact.setUse(ContactPoint.ContactPointUse.HOME);  
        contact.setSystem(ContactPoint.ContactPointSystem.PHONE);  
  
        contact.setValue("1 (416) 340-4800");  
  
        pat.setGender(Enumerations.AdministrativeGender.MALE);  
    }  
}
```

### Primitive Types

```
DateTimeType effective = new DateTimeType();
effective.setValue(new Date());
effective.setValue(new Date(), TemporalPrecision.FINEST());
effective.setValueAsString("2017-09-11T14:05:00");

BooleanType active = new BooleanType();
active.setValue(true);
active.setValueAsString("true");
```

Primitive Representation

String Representation

```
DecimalType value = new DecimalType();
value.setValue(1.2d);
value.setValueAsString("1.20000");
```

Activate Windows  
Go to Settings to activate Windows.



## Primitive Types (2)

```
Observation obs = new Observation();

// These are equivalent
obs.setComment("This is a comment");
obs.setCommentElement(new StringType("This is a comment"));

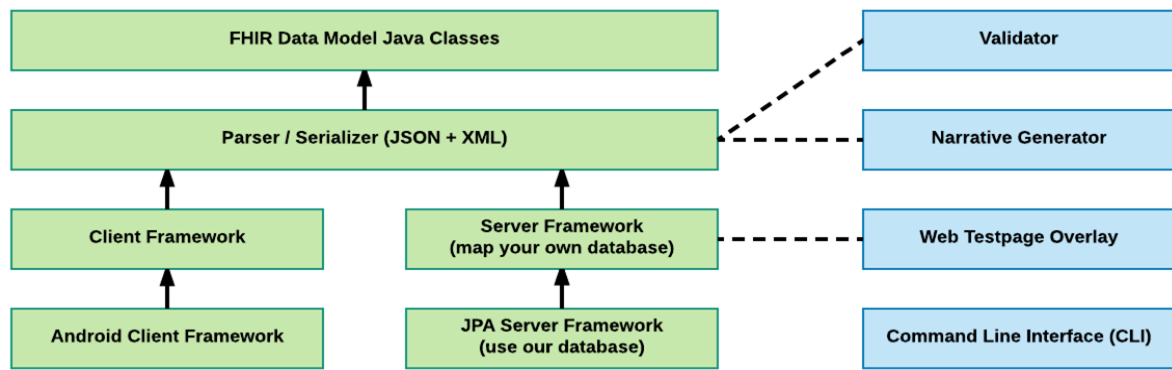
// Get the primitive or get the FHIR type
String comment = obs.getComment();
StringType commentElement = obs.getCommentElement();
```

Activate Windows  
Go to Settings to activate Windows.



### Server Framework





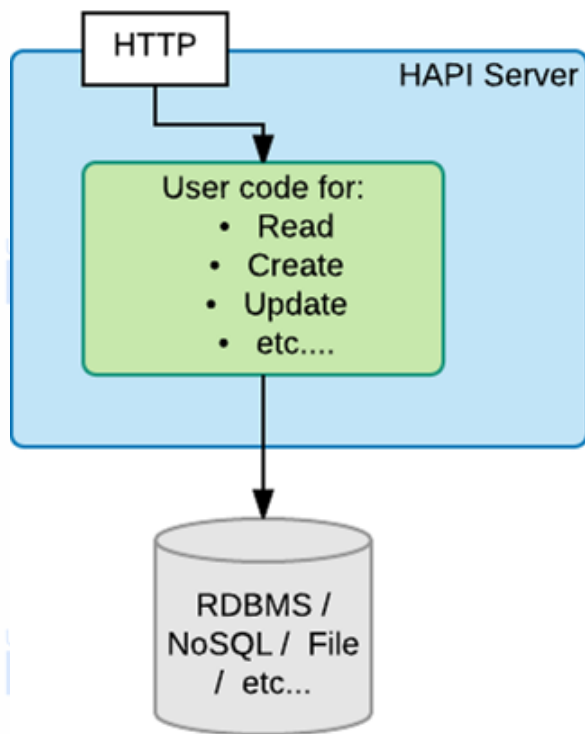
### Server Architecture

- HAPI FHIR provides a REST Server framework
- Based on standard JEE/Servlet 2.5+ (Tomcat, Glassfish, Websphere, JBoss, etc)
- Inspired by (but not based on) JAX-RS, RestEasy, Spring REST, etc.

\* A JAX-RS HAPI module is available but it is not covered here

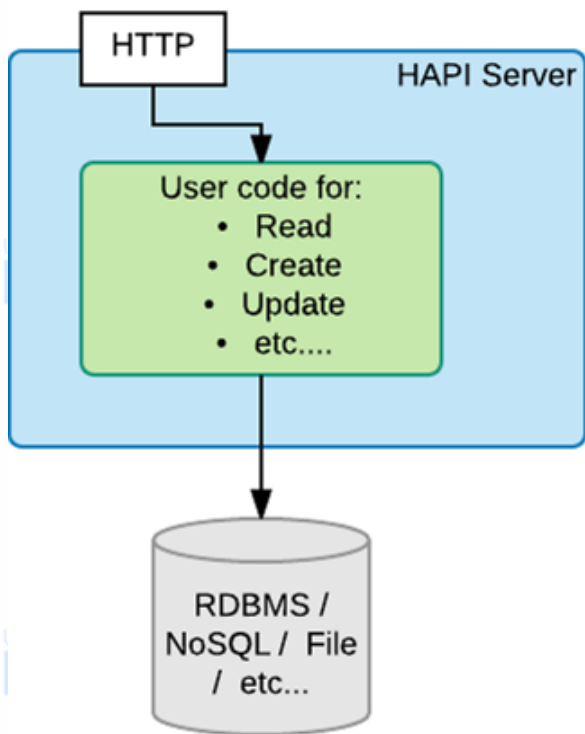
### Server Architecture (2)

- You supply Java code for CRUD operations you want to support in your server
  - Read
  - Create
  - Update
  - Delete
  - Search
  - etc...



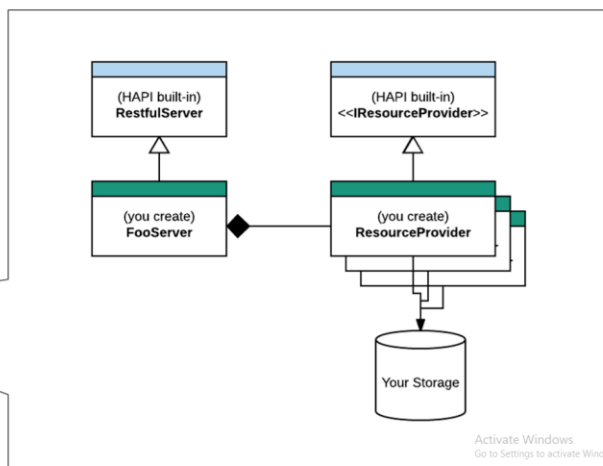
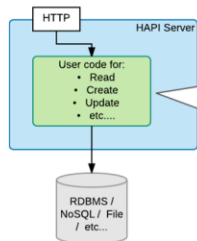
### Server Architecture (3)

- **HAPI FHIR will:**
  - **Handle parsing and encoding**
  - **Route URLs, Verbs, and parameters to appropriate methods**
  - **Understand FHIR escaping rules**



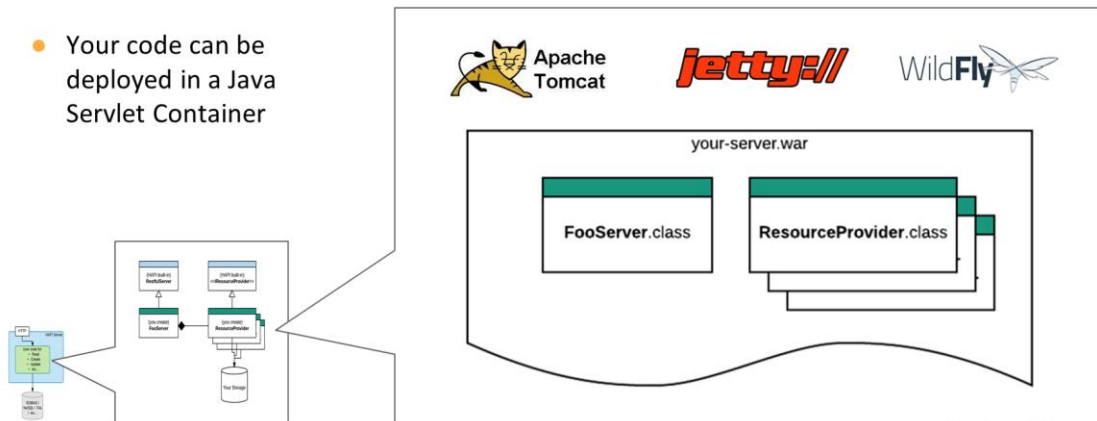
### Server Architecture (3)

- The primary component is the **IResourceProvider** implementation which you create



## Server Architecture (3)

- Your code can be deployed in a Java Servlet Container



154

## Resource Providers

- ResourceProviders implement the `IResourceProvider` interface and the `getResourceType()` method
- You create one resource provider **per resource type**

```
public class Example01_StubResourceProvider implements IResourceProvider {  
  
    public Class<? extends IBaseResource> getResourceType() {  
        return Patient.class;  
    }  
}
```

```
// your code to handle resources
```

Activate Windows  
Go to Settings to activate Windows.

155

```
public class Example01_StubResourceProvider implements IResourceProvider {  
    public Class<? extends IBaseResource> getResourceType() {  
        return Patient.class;  
    }  
    @Read  
    public Patient read(@IdParam IdType theId) {
```

```

    return null; // populate this
}
@Create
void create(@ResourceParam Patient thePatient) {
    // save the resource
}

@Search
List<Patient> search(
    @OptionalParam(name="family") StringParam theFamily,
    @OptionalParam(name="given") StringParam theGiven
){
    return null; // populate this
}
}
}

```

## Rest Server

- Not much needs go in your REST server (set a context and register providers)

```

@WebServlet("/")
public class Example03_SimpleRestfulServer extends RestfulServer {

    @Override
    protected void initialize() throws ServletException {
        // Create a context for the appropriate version
        setFhirContext(FhirContext.forDstu3());
    }
}

```

```

// Register resource providers

```

```

registerProvider(new Example04_PatientResourceProviderWithCreate());

```

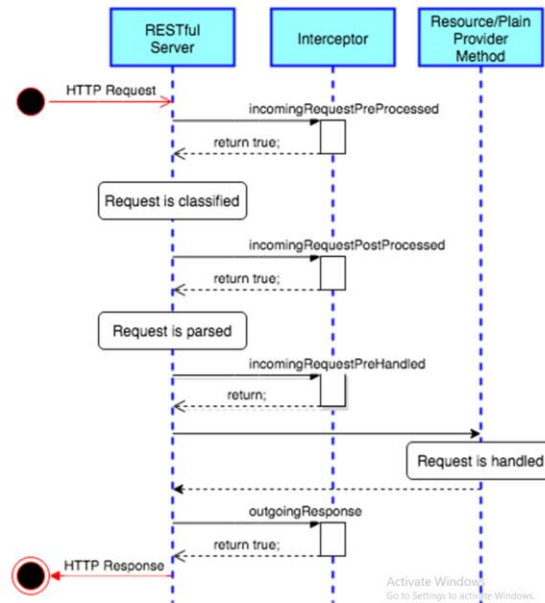
Activate Windows  
Go to Settings to activate Windows.

158

# Server Interceptors

Server Interceptors are registered to the server and they can:

- Examine the request
- Change the request
- Deny the request
- Examine the response
- Change the response



## Built-in Interceptors

- **LoggingInterceptor**
  - Log requests as they come in (highly configurable)
  - <http://hapifhir.io/apidocs/ca/uhn/fhir/rest/server/interceptor/LoggingInterceptor.html>
- **CorsInterceptor**
  - Allow CORS (JavaScript requests from another server)
- **RequestValidatingInterceptor and ResponseValidatingInterceptor**
  - Validate payloads (more on validation later)
- **ResponseHighlighterInterceptor**
  - Use a nice HTML response for browsers
- **AuthorizationInterceptor**
  - Authorize individual requests (more shortly)



# Using Interceptors

- Interceptors are registered with the server just like resource providers

```
@WebServlet("/*")
public class Example02_SimpleRestfulServer extends RestfulServer {

    @Override
    protected void initialize() throws ServletException {
        // Create a context for the appropriate version
        setFhirContext(FhirContext.forDstu3());

        // Register resource providers
        registerProvider(new Example01_PatientResourceProvider());

        // Format the responses in nice HTML
        registerInterceptor(new ResponseHighlighterInterceptor());
    }
}
```

Activate Windows  
Go to Settings to activate Windows.

161

## Authorization Interceptor

- AuthorizationInterceptor is a class you extend to provide authorization (AuthZ) and possibly authentication (AuthN) on your FHIR server
- You supply permissions that the requestor should have
- HAPI enforces these permissions
- E.g:
  - Based on an incoming header, the user has read access but not write access

```
public class Example03_AuthorizationInterceptor extends AuthorizationInterceptor {
```

```
    @Override
```

```
    public List<IAuthRule> buildRuleList(RequestDetails theRequestDetails) {
```

```
        // Process this header
```

```
        String authHeader = theRequestDetails.getHeader("Authorization");
```

```
        // Apply rules
```

```
        RuleBuilder builder = new RuleBuilder();
```

```
        builder
```

```
            .allow().metadata().andThen()
```

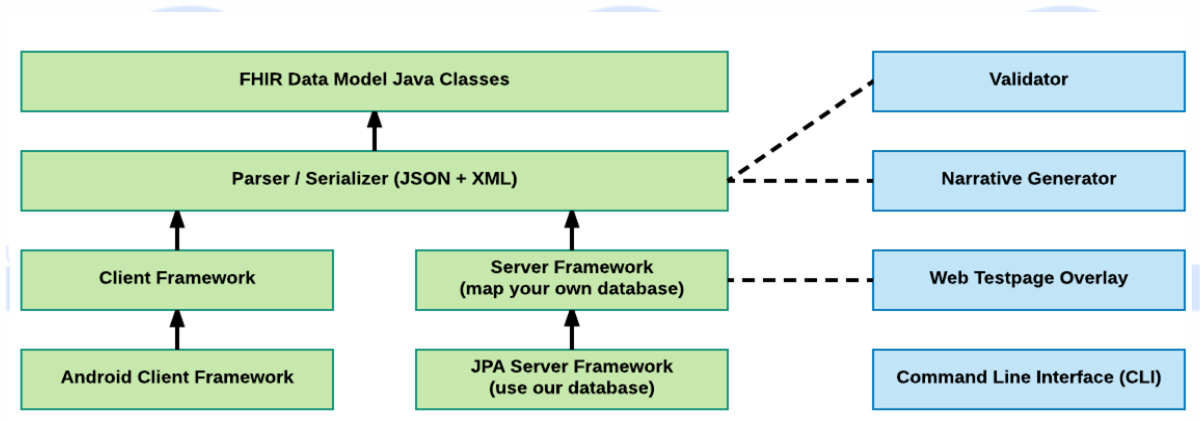
```
            .allow().read().allResources().withAnyId().andThen()
```

```

.allow().write().resourcesOfType(Observation.class).inCompartment("Patient", new
IdType("Patient/123"));
return builder.build();
}
}

```

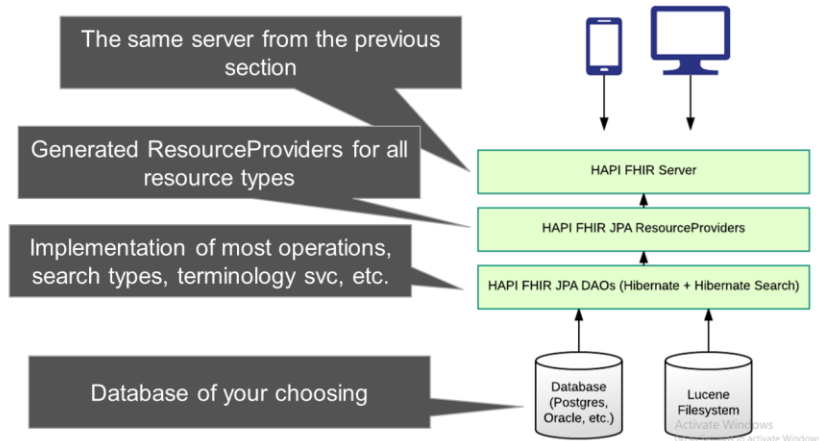
### JPA Server Framework



### JPA Server Framework

- HAPI JPA Server is a complete server implementation from the database schema up
- It includes:
  - All standard REST verbs (create, read, update, delete)
  - Many fancy REST features (ETag, conditional, patch, etc.)
  - Extensive search support including custom parameters
  - Terminology services
  - Subscription services
  - Many configurable settings

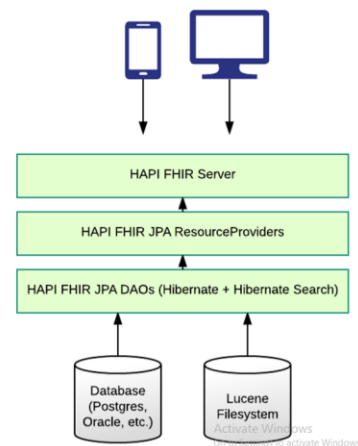
# JPA Architecture



166

# JPA Architecture

- The JPA Server uses Hibernate, which means it supports several RDBMS platforms:
  - Oracle, Postgres, MySQL, SQL Server
- Most examples use Derby
  - Derby is great for testing, but it not a production option!



167

## Lucene

- HAPI uses Apache Lucene to provide two features:
  - Fulltext searching within resources ( \_text and \_content parameters)
  - Terminology Services
- Lucene stores its files on the filesystem
- Lucene can be safely disabled

## Using JPA

- JPA Server is a collection of components that need to be “glued together”
- Examples are available which provide this glue

<https://github.com/furore-fhir/fhirstarters/tree/master/java/hapi-fhirstarters-jpaserver-example>

HAPI as a Potential Architecture for a National HER

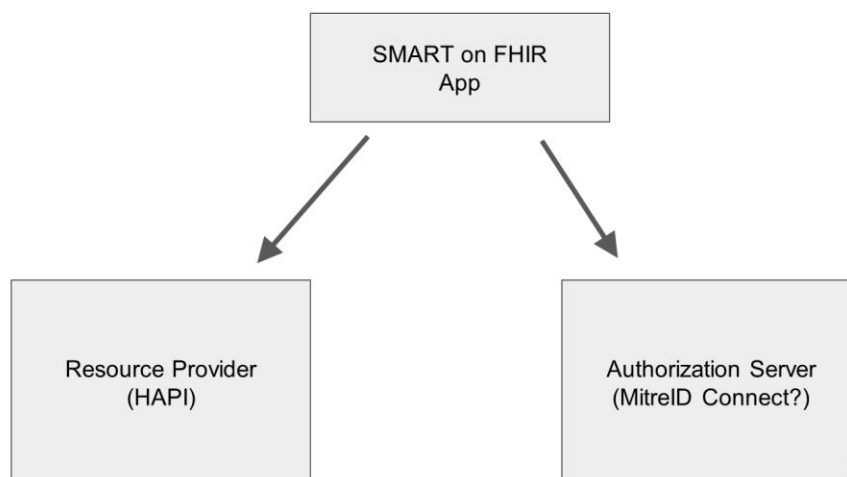
Model: Service Façade

- This refers to building reusable services on existing sources of data
- Examples include:
  - Hospital and doctor EHRs
  - Laboratory systems
  - Radiology
- This pattern allows you to create consistent APIs (consistent in terms of data, API, Security, etc.)
- HAPI RestfulServer can act as a bridge between existing databases and your FHIR interfaces

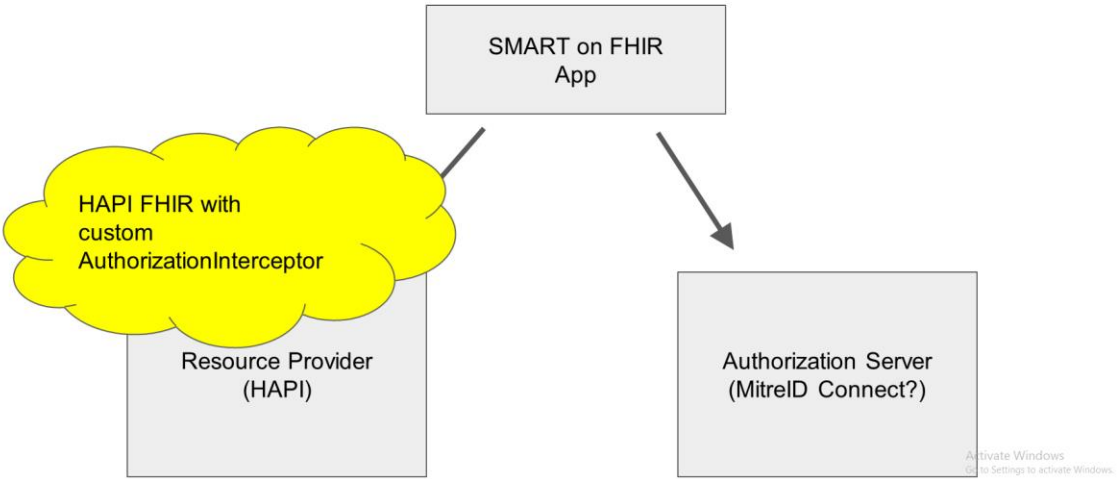
Model: Repository

- The HAPI Server can also act as a complete FHIR repository
- This could be useful as:
  - A Patient index
  - A central store of lab tests, radiology reports
  - A backend for applications

## SMART on FHIR Security



# SMART on FHIR Security



Esa Unggul

Esa Unggul

Esa Unggul

