FHIRの実装
～オープンソースによる
患者情報検索(PDQm)を例にして～

日本IHE協会 ITI技術委員会
木村　雅彦

– IHE Tutorial –　2023-06-29

第27回日本医療情報学会春季学術大会
COI開示

演題名： FHIRの実装～オープンソースによる
患者情報検索(PDQm)を例にして～
筆頭演者名： 木村　雅彦

私が発表する今回の演題について開示すべき

COIはありません。

発表のねらい

- 患者情報検索(PDQm)をオープンソース(HAPI FHIR※1)を利用して実装してみる
- アクタとして実現すべき機能を理解する
- HAPI FHIRの特徴や実装の概要を知る
- FHIR※2の実装の具体的な内容※3を理解する

※1: HAPI FHIRはSmile CDR Inc.の製品

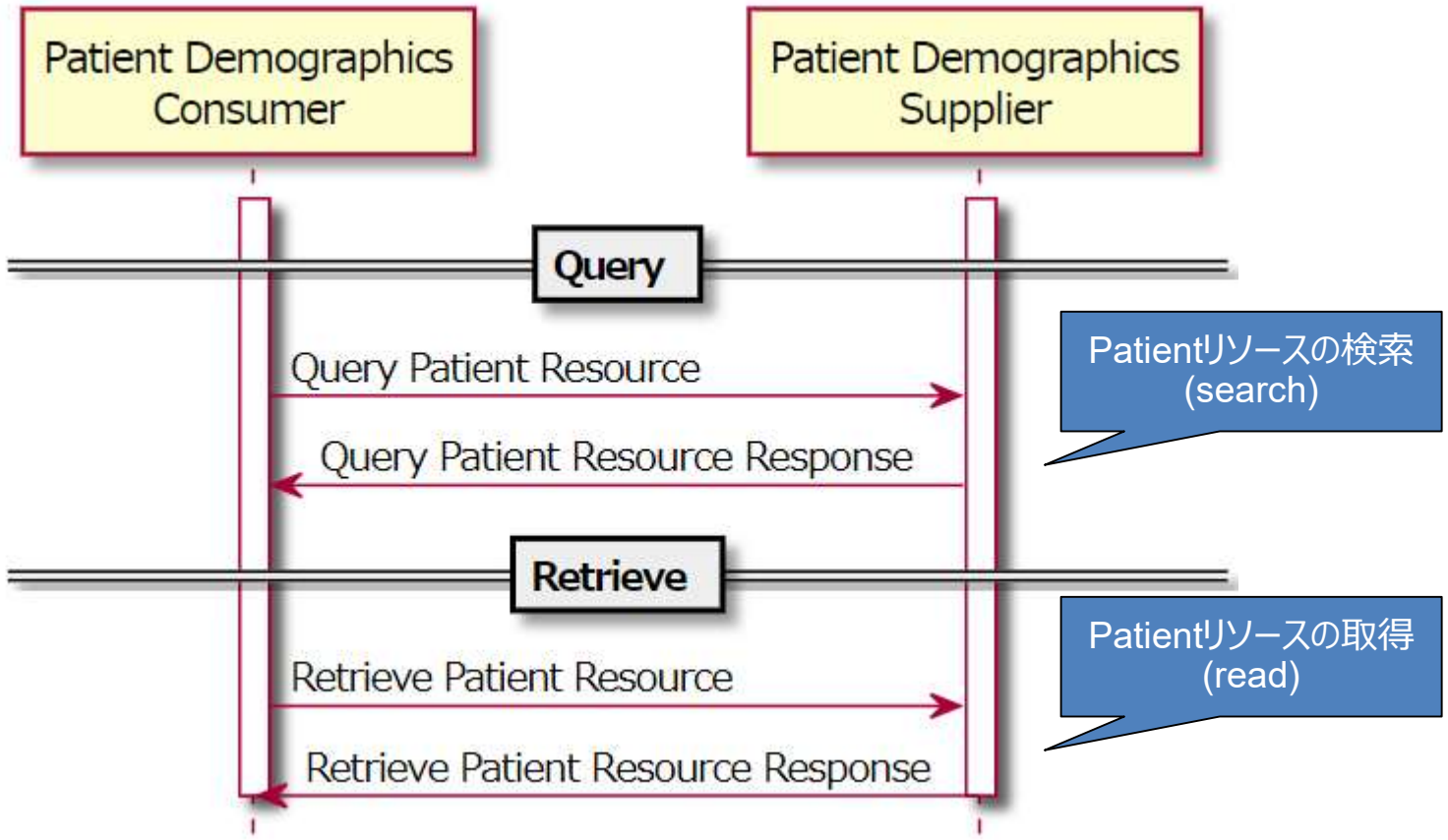
※2: HL7®, FHIR®はHL7 Internationalの登録商標

※3: ソースコードは以下のURLで入手可能
https://github.com/tmskmr/restful-server-example

PDQmについて

- 「Patient Demographics Query for Mobile」の略
- モバイルデバイスから患者の基本情報を検索するIHEの統合プロファイル（PDQのFHIR版）
- 現在 v2.4.0でTrial Implementation
- HL7 FHIR R4のPatientリソースに対する検索と取得をそのままの仕様で利用する
- 下記のURLで技術仕様が公開されている
  https://profiles.ihe.net/ITI/PDQm/index.html

PDQmのアクタと トランザクション[ITI-78]



https://profiles.ihe.net/ITI/PDQm/ITI-78.htmlより引用

PDQmの検索パラメータ

- PDQmで定義されている主な検索パラメータ

検索パラメータ	データタイプ	内容	検索パラメータの例
_id	String	リソースID	_id=1234
family given	String	患者氏名の姓と名	family=田中 given:exact=実
identifier	Token	患者ID	identifier=urn:oid:2.16.840.1.113883.4.1|123456789
telecom	Token	電話番号	telecom=03-1234-5678
birthdate	Date	生年月日	birthdate=eq1995-01-01 birthdate=ge2000-12-31
address	String	住所	address=東京都中央区日本橋
gender	Token	性別	gender=male

完全一致

割り当て権限者のチェック

範囲指定等のPrefix

IHE-Jコネクタソン2022でのシナリオ

No.	R/O※	内容	備考
120	O	「family=金田」の検索	
130	O	「family=金田」および「given=満子」の検索	
140	O	「family=金」（前方一致）の検索	
150	O	「identifier=1234567890」（割り当て権限者を含む）の検索	
160	O	「birthdate=1982-02-02」の検索	
170	O	「family=栄田」（exactパラメータを使用）および「gender=female」の検索を実行	
180	O	「family=福岡」および「birthdate=1990-10-10」の検索	
250	R	それまでの検索（search）の応答メッセージの1つから、Patientリソースを取得（read）	
300	R	Supplierに一致するものがないクエリ	

割り当て権限者のチェック

完全一致

複数の検索パラメータの組み合わせ

取得

※R=必須, O=オプション

HAPI FHIRとは

- Smile CDR Inc.という会社が公開している
  HL7 FHIR用のオープンソースのライブラリ
- Java言語で記述されている
  →OSを選ばずサーバーの開発に適している
- Apache 2.0でライセンスされている
  →商用のアプリケーションにも利用しやすい
- 公式サイト:
  https://hapifhir.io/

FHIR対応のライブラリ

- リファレンス実装が下記サイトで公開されている
  http://hl7.org/fhir/R4/downloads.html

Reference Implementations

There are many open source reference implementations available to help implem
the more common implementations used by implementers:

Java	HAPI-FHIR: Object Models, Parsers, Client + Server Framework, FHIR Validator, & Utilities. The specification is built with this Java code
C#	HL7.FHIR: Object models, Parsers/Serializers, Utilities, and a Client. Source code on GitHub at http://github.com/ewoutkramer/fhir-net-api
Pascal	FhirServer: Object models, Parsers/Serializers, Validator, Utilities, Client, and the FHIR Reference server. Requires Delphi (Unicode versions)
XML	XML Tools: Document Rendering Stylesheet, supplementary implementation schemas and transforms
Javascript	See the HL7 wiki for Javascript libraries (Clients and Utilities for both servers and clients)
Swift	Swift-FHIR: Object Model, Client and Utilities

HAPI FHIRはJava言語のリファレンス実装

Java以外の開発言語のライブラリも公開されている

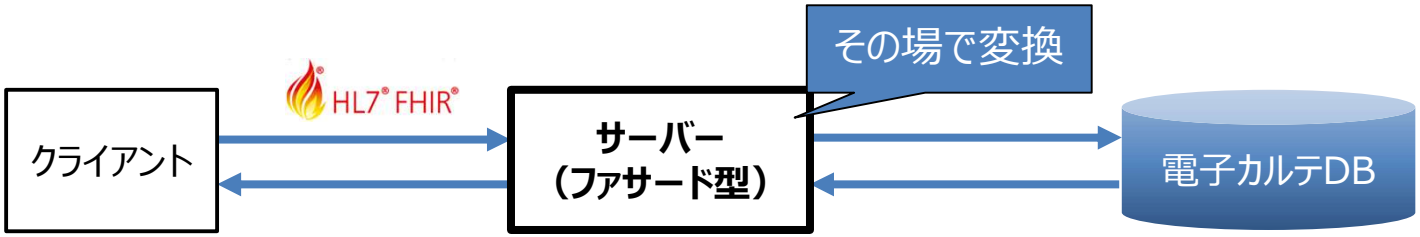
HAPI FHIRの特長

- ファサード型のサーバーに対応している
- リソース、データ型、列挙値などのオブジェクトモデルがあらかじめ定義されている
  →プログラムによるロジックの作り込みが可能
  →統合開発環境の開発支援機能が利用できる
- 必要な外部ライブラリを自動的に取得できる
- テストページを自動生成できる
- ドキュメントが充実している

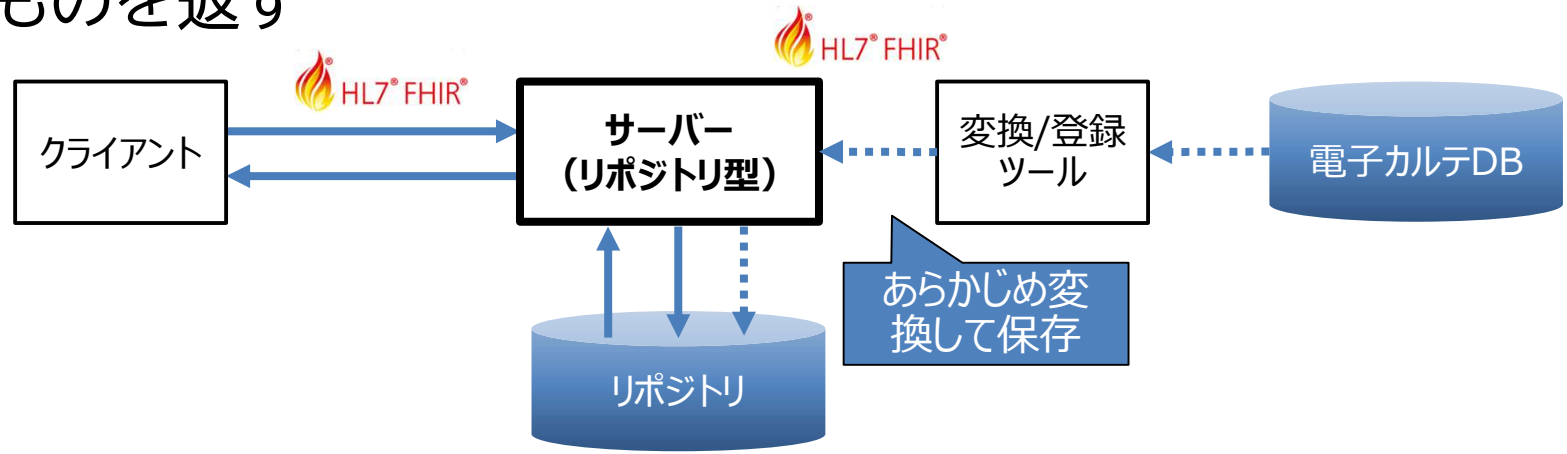
ファサード型とリポジトリ型

- **ファサード※型**：クライアントからの要求時にその場でFHIRリソースに変換して返す

※建物正面の装飾を指す建築用語



- **リポジトリ型**：あらかじめFHIRに変換して保存しておいたものを返す



開発支援機能の利用例①

```
1  package ca.uhn.example.converter;
2
3  import org.apache.ibatis.session.SqlSession;
20
21 public class MyPatientConverter {
22     static public Patient CreatePatientResource(SqlSession session, MyPatient table) {
23         try {
24             Patient resource = new Patient();
25
26             // meta
27             Meta meta =
28
29
30
31
32
33
34
35
36
37
38
39
40             //  name
41             resource.addName(Common.CreateHumanName(table.getPt_family_ide(), table.getPt_given_ide(), "IDE"));
42             resource.addName(Common.CreateHumanName(table.getPt_family_syl(), table.getPt_given_syl(), "SYL"));
43             resource.addName(Common.CreateHumanName(table.getPt_family_abc(), table.getPt_given_abc(), "ABC"));
44
```

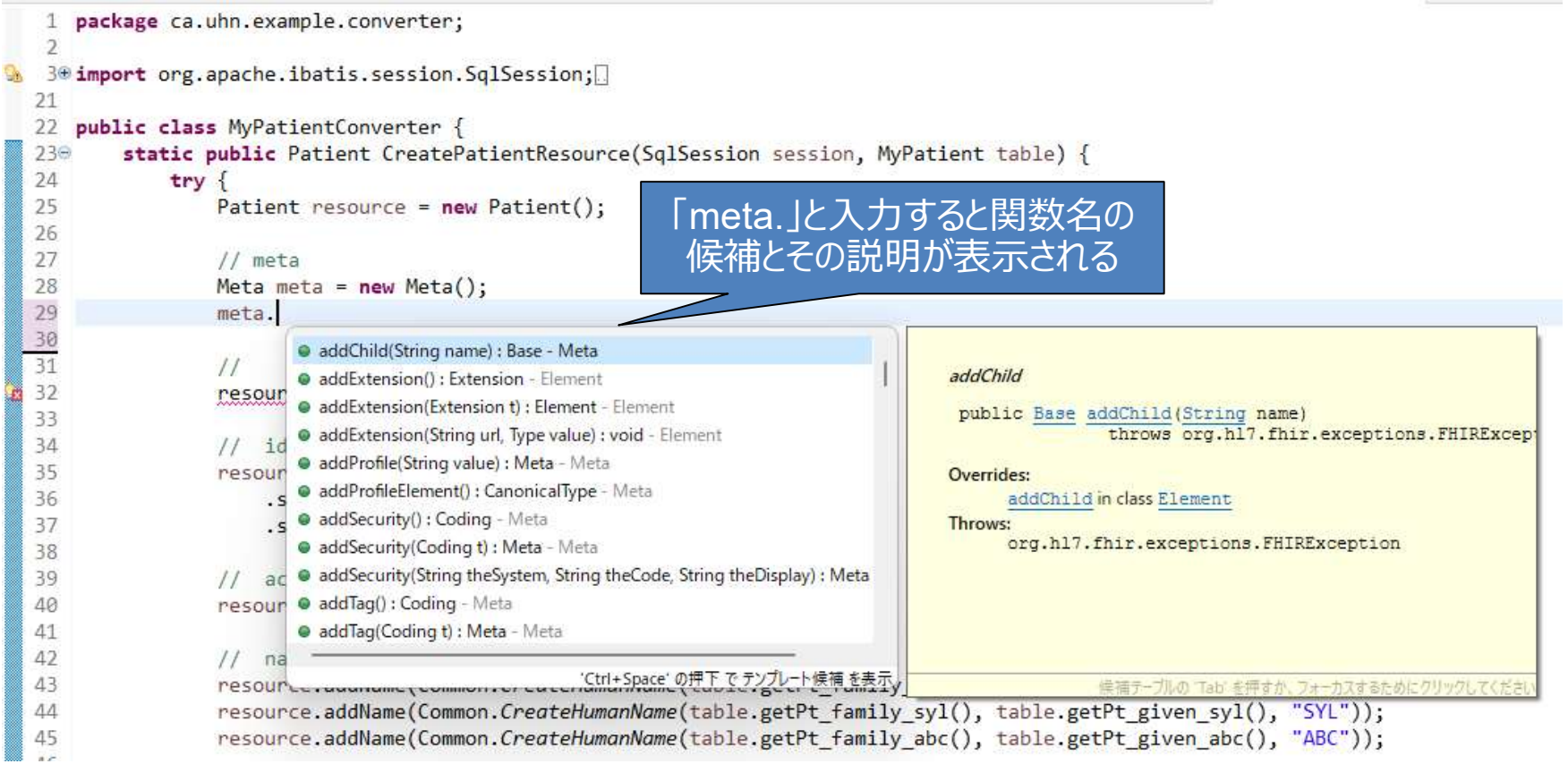
Meta を型に解決できません

48 個のクイック・フィックスが使用可能です:

- 'Meta' をインポートします (org.hl7.fhir.dstu2.model)
- 'Meta' をインポートします (org.hl7.fhir.dstu2016may.model)
- 'Meta' をインポートします (org.hl7.fhir.dstu3.model)
- 'Meta' をインポートします (org.hl7.fhir.r4.model)
- 'Meta' をインポートします (org.hl7.fhir.r4b.model)
- 'Meta' をインポートします (org.hl7.fhir.r5.model)
- クラス 'Meta' を作成します

エラーになった「Meta」にカーソルを置くとインポートすべきオブジェクトモデルの候補が表示される

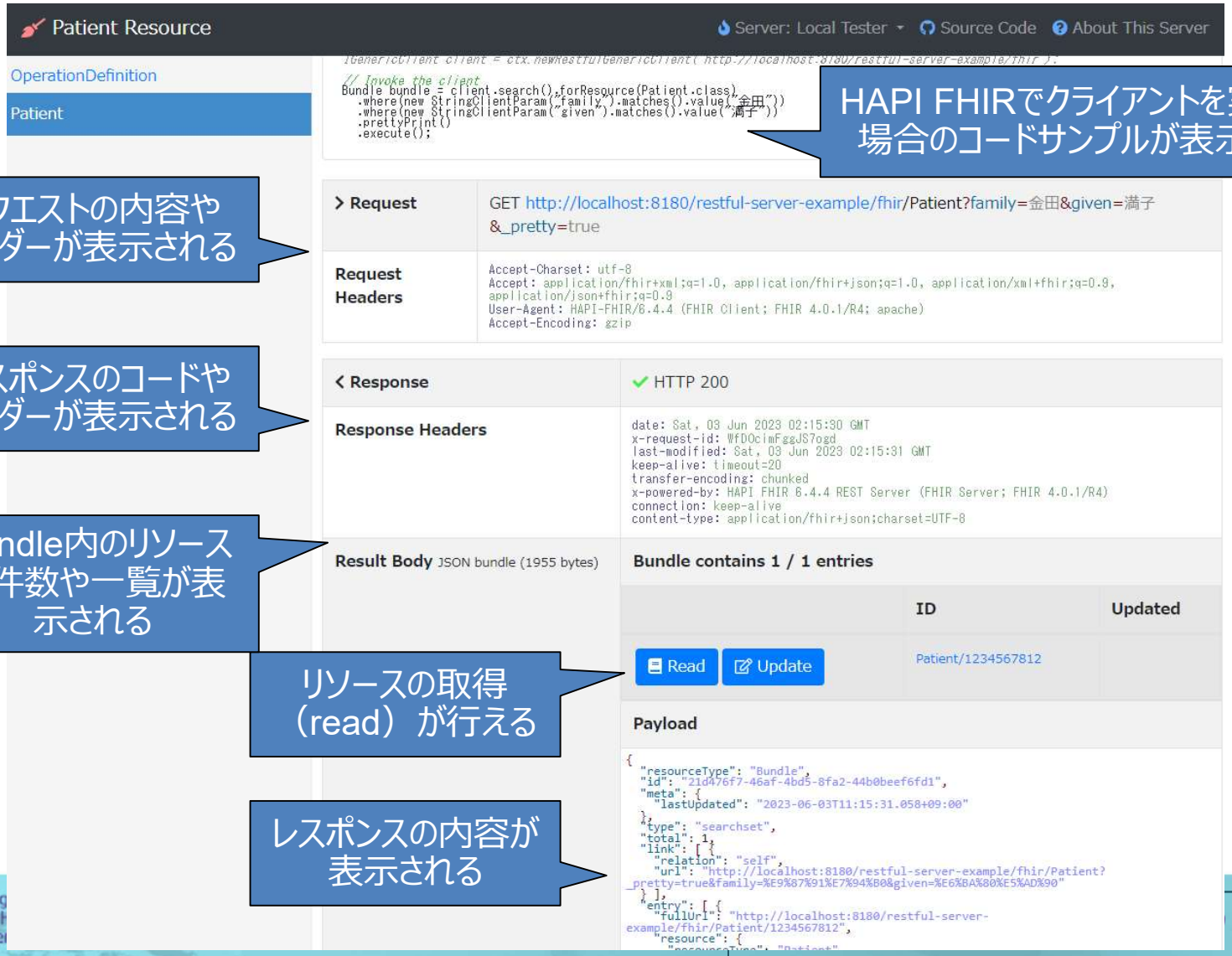
開発支援機能の利用例②




```
1  package ca.uhn.example.converter;
2
3  import org.apache.ibatis.session.SqlSession;
21
22 public class MyPatientConverter {
23     static public Patient CreatePatientResource(SqlSession session, MyPatient table) {
24         try {
25             Patient resource = new Patient();
26
27             // meta
28             Meta meta = new Meta();
29             meta.
30
31             //
32             resour
33
34             //  id
35             resour
36                 .s
37                 .s
38
39             //  ac
40             resour
41
42             //  na
43             resour
44             resource.addName(Common.CreateHumanName(table.getPt_family_syl(), table.getPt_given_syl(), "SYL"));
45             resource.addName(Common.CreateHumanName(table.getPt_family_abc(), table.getPt_given_abc(), "ABC"));
```

テストページの例ー検索画面

テストページの例ー結果画面

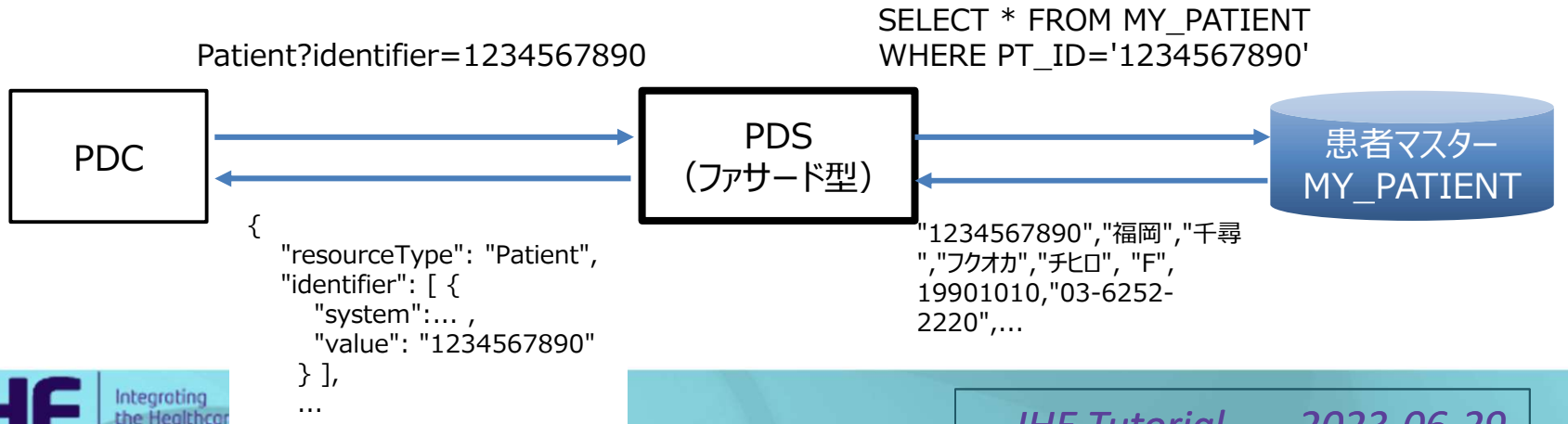


HAPI FHIRのドキュメント例

- https://hapifhir.io/hapi-fhir/docs/

実装するシステム

- PDQmのPatient Demographics Supplier(PDS)アクタを実装する
- 電子カルテを想定した患者マスターテーブルの情報を返すファサード型サーバーとする
- DBにはPostgreSQLを、DBアクセスにはMyBatisを使用する



患者マスターテーブル

- テーブルのスキーマの内容

フィールド名	データタイプ	PKEY	NULL	内容
PT_ID	CHAR(10)	Y	N	患者ID
PT_FAMILY_IDE	CHAR(20)		N	患者漢字氏名（姓）
PT_GIVEN_IDE	CHAR(20)		N	患者漢字氏名（名）
PT_FAMILY_SYL	CHAR(20)		N	患者カナ氏名（姓）
PT_GIVEN_SYL	CHAR(20)		N	患者カナ氏名（名）
PT_FAMILY_ABC	CHAR(20)		N	患者ローマ字氏名（姓）
PT_GIVEN_ABC	CHAR(20)		N	患者ローマ字氏名（名）
PT_GENDER	CHAR(1)		N	患者性別（M=男、F=女、O=その他）
PT_BIRTHDATE	DATE		N	患者生年月日
PT_PHONE	CHAR(20)		N	患者電話番号
PT_ZIP	CHAR(10)		N	患者郵便番号
PT_ADDRESS	CHAR(64)		N	患者住所

患者IDをリソースIDとする想定

姓と名を別のフィールドに格納

患者マスターテーブル

- テスト用データはコネクタソン2022で指定されたものを使用

開発・実行に必要なソフトウェア

- **Eclipse**※1:統合開発環境(IDE)
- **MyBatis Generator** ※2**プラグイン:**
  MyBatis※2用のマッパークラスの生成ツール
- **Maven**※2:プロジェクト構成管理、ビルドツール
- **Tomcat**※2:ビルドしたJavaプログラム(WARファイル)を実行するアプリケーション・サーバー
- **Java**※3:Tomcat、Javaプログラムの実行環境
- **PostgreSQL**※4**:**患者マスター用データベース

※1:EclipseはEclipse Foundationの著作物、※2:MyBatis, MyBatis Generator, Maven, TomcatはThe Apache Software Foundationの著作物、※3:JavaはOracleの登録商標、※4:PostgreSQLはThe PostgreSQL Global Development Groupの著作物

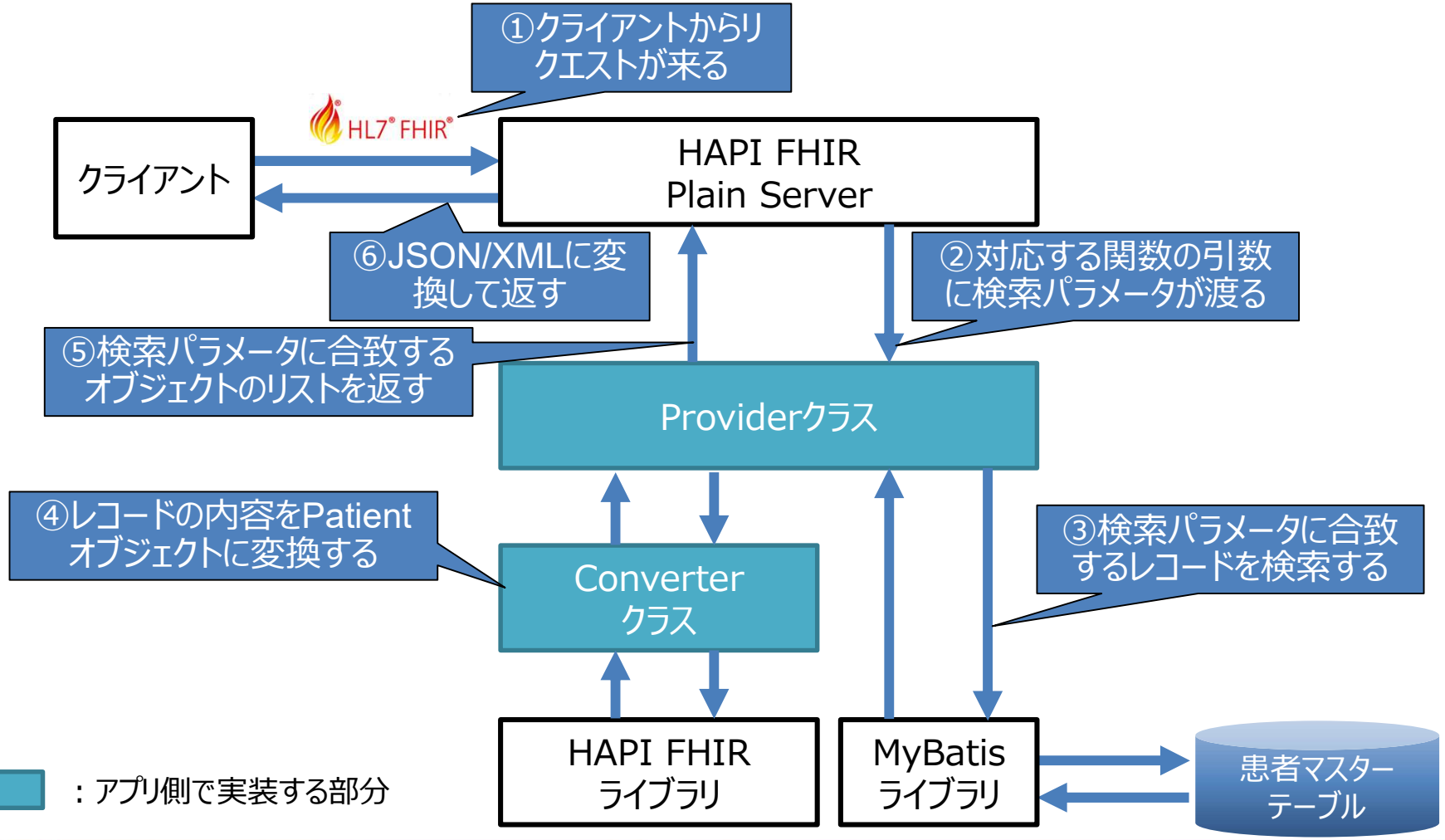
開発の主な手順

1. GitHubからサンプルプロジェクトをダウンロードし、Eclipseにインポート
2. MyBatis Generatorでマッパーを生成
3. MyBatisの初期化、DB接続処理等を実装
4. **Providerクラス(検索・取得処理)を実装**
5. **Converterクラス(リソース生成)を実装**
6. 実装したProviderクラスをRestfulServerクラスに登録(Patientは登録済みなので不要)
7. ビルドしてテストする

HAPI FHIRの
サンプルプロジェクト

- https://github.com/hapifhir/hapi-fhir-examplesから入手可能
- restful-server-exampleがファサード型FHIRサーバーに相当
- HL7 FHIRやHAPI FHIRのバージョンが古いので修正が必要
  - DSTU2→R4（サーバー、テストページ両方）
  - 5.6.0-PRE1-SNAPSHOT→6.4.4

検索時の処理の流れ


①クライアントからリクエストが来る
HL7® FHIR®
クライアント
HAPI FHIR
Plain Server
⑥JSON/XMLに変換して返す
②対応する関数の引数に検索パラメータが渡る
⑤検索パラメータに合致するオブジェクトのリストを返す
Providerクラス
④レコードの内容をPatientオブジェクトに変換する
Converter
クラス
③検索パラメータに合致するレコードを検索する
HAPI FHIR
ライブラリ
MyBatis
ライブラリ
患者マスター
テーブル
：アプリ側で実装する部分

HAPI FHIRのメリット

- RESTサーバーの必要な処理を受け持ってくれる
- 検索パラメータを解析し、Providerクラスの適切な関数を呼び出して引数に変換して渡してくれる
  - :exactやge, eq等の情報も取得できる
- クライアントの要求に従って、オブジェクトモデルからXML/JSONへの変換を自動で行ってくれる
- コンフォーマンス情報を自動で生成してくれる
- 認証などの各種割り込み関数が用意されている

→本当に必要な処理だけを実装すればよい

Providerクラスの実装

- 対応するリソースごとにIResourceProviderクラスの子クラスとして実装する
- 検索用の関数(複数可)は@Search、
  取得用の関数は@Readでアノテーション※する
- 対応する検索パラメータに対する引数を用意して、@RequiredParam等でアノテーションする
- 検索パラメータに合致するレコードを検索し、オブジェクトのリストに変換して返す

※ アノテーションとは、プログラムのソースコードに注釈を加えて、コンパイラに情報を伝えること

Providerクラスの検索用関数の例①

検索用であることを示す

Patientオブジェクトのリストを返す

引数が対応する検索パラメータを示す
必須パラメータの場合は
@RequiredParamを使用する

```
@Search()
public List<Patient> search(
      @OptionalParam(name = Patient.SP_IDENTIFIER) TokenParam theIdentifier,
      @OptionalParam(name = Patient.SP_NAME) StringParam theName,
      @OptionalParam(name = Patient.SP_FAMILY) StringParam theFamily,
      @OptionalParam(name = Patient.SP_GIVEN) StringParam theGiven,
      @OptionalParam(name = Patient.SP_BIRTHDATE) DateRangeParam theBirthDate,
      @OptionalParam(name = Patient.SP_GENDER) TokenParam theGender,
      @OptionalParam(name = Patient.SP_TELECOM) TokenParam theTelecom,
      @OptionalParam(name = Patient.SP_ADDRESS) StringParam theAddress,
      @Sort SortSpec theSort
) {
...
```

検索パラメータのタイプごとにクラスが定義されている

_sortパラメータに対するクラスも定義されている

Providerクラスの検索用関数の例②

```
List<Patient> list = new ArrayList<Patient>();
SqlSession session = null;
try {
   session = Common.OpenSqlSession();
   List<MyPatient> tables = selectMyPatient(session, theIdentifier, theName,
       theFamily, theGiven, theBirthDate, theGender, theTelecom, theAddress );

   for (MyPatient table : tables) {
      Patient resource = MyPatientConverter.CreateResource(session, table);
      list.add(resource);
   }

   sortResourceList(list, theSort);
}
...
return list;
```

Patientオブジェクトのリストを生成する

DBに接続する

検索条件に合うMY_PATIENTのレコードを検索する

レコードの内容を順にPatientオブジェクトに変換する

リストに追加する

リストをソートする

DBとの接続を閉じる（省略）

リストを返す

Providerクラスでの DB検索の例

```
public List<MyPatient> selectMyPatient(SqlSession session, ...) {
    MyPatientMapper mapper =
        session.getMapper(MyPatientMapper.class);

    MyPatientExample example = new MyPatientExample();
    MyPatientExample.Criteria criteria = example.createCriteria();

    if (theIdentifier != null)
        criteria.andPt_idEqualTo(theIdentifier.getValue());
    if (theGiven != null)
        criteria.andPt_given_sylEqualTo(theGiven.getValue());
    ...
    return mapper.selectByExample(example);
}
```

テーブルごとのMapperクラスを取得する

ExampleクラスとCriteriaクラスのオブジェクトを生成する

指定された検索条件をANDでcriteriaに追加する
→検索パラメータの組み合わせに対応できる

検索を実行すると、テーブルオブジェクトのリストが返る

検索パラメータの実装例 String型（given）

```
if (theGiven != null) {
  String given = theGiven.getValue();
  if (Common.IsFullKatakana(given)) {
    if (theGiven.isExact())
      criteria.andPt_given_sylEqualTo(given);
    else
      criteria.andPt_given_sylLike(given + "%");
  }
  else{
    if (theGiven.isExact())
      criteria.andPt_given_ideEqualTo(given);
    else
      criteria.andPt_given_ideLike(given + "%");
  }
}
```

検索パラメータの値を取得する（例：「実」）

:exactが指定されているか（例：true）
→**:exactパラメータに対応できる**

:exactが指定されていたらEQUAL検索を行う

:exactが指定されていなければLIKE検索を行う

漢字による検索の場合

パラメータ例：「given:exact=実」

検索パラメータの実装例
Date型(birthdate)

```
if (theBirthDate != null) {
  if (theBirthDate.getLowerBound() != null) {
    Date sdate = theBirthDate.getLowerBound().getValue();
    ParamPrefixEnum sprefix = theBirthDate.getLowerBound().getPrefix();
    if (sprefix == ParamPrefixEnum.GREATERTHAN_OR_EQUALS)
      criteria.andPt_birthdateGreaterThanOrEqualTo(sdate);
    else if (sprefix == ParamPrefixEnum.GREATERTHAN)
      criteria.andPt_birthdateGreaterThan(sdate);
    else if (sprefix == ParamPrefixEnum.EQUAL)
      criteria.andPt_birthdateEqualTo(sdate);
  }

  if (theBirthDate.getUpperBound() != null) {
    Date edate = theBirthDate.getUpperBound().getValue();
    ParamPrefixEnum eprefix = theBirthDate.getUpperBound().getPrefix();
    if (eprefix == ParamPrefixEnum.LESSTHAN_OR_EQUALS)
      criteria.andPt_birthdateLessThanOrEqualTo(edate);
    else if (eprefix == ParamPrefixEnum.LESSTHAN)
      criteria.andPt_birthdateLessThan(edate);
  }
```

下限値を取得する
(例：2000-12-31)

Prefixを取得する（例:ge）
→**Prefixに対応できる**

ge (以上)が指定されている場合
GreaterThanOrEqualToを使用

gt (より大きい)が指定されている場合
GreaterThanを使用

eq (等しい)が指定されている場合
EqualToを使用

上限値を取得する

le (以下)が指定されている場合
LessThanOrEqualToを使用

lt (より小さい)が指定されている場合
LessThanを使用

パラメータ例：「birthdate=ge2000-12-31」

検索パラメータの実装例 Token型(identifier)

```
if (theIdentifier != null) {
  String system = theIdentifier.getSystem();
  if (system != null && !system.equals(Common.GetNSPatient()))
    throw new ResourceNotFoundException("Not recognized identifier system:" +
system);

  String value = theIdentifier.getValue();
  if (value != null)
    criteria.andPt_idEqualTo(value);
}
```

system部分を取得して、値をチェックする
→割り当て権限者に対応できる

HTTP STATUS 404 (Not Found) を返す

code部分を取得してEqualToで検索する（例：123456789）

パラメータ例：「identifier=urn:oid:2.16.840.1.113883.4.1|123456789」

Providerクラスの取得用関数の例

```
@Read(version = false)
public Patient readPatient(@IdParam IdType theId) {
   SqlSession session = null;
   try {
      session = Common.OpenSqlSession();
      MyPatientMapper mapper = session.getMapper(MyPatientMapper.class);
      MyPatient table = mapper.selectByPrimaryKey(theId.getIdPart());
      if (table == null)
         throw new ResourceNotFoundException(theId);
      return MyPatientConverter.CreateResource(session, table);
   }
   catch(Exception e) {
      throw new InternalErrorException(e);
   }
   finally{
       session.close();
   }
```

取得用であることを示す

Patientオブジェクトを返す

リソースIDが引数として渡る

DBに接続する

リソースIDを患者IDとしてMY_PATIENTを検索する

取得したレコードの内容をPatientオブジェクトに変換して返す

DBとの接続を閉じる

Converterクラスの実装

- 対応するFHIRリソースやデータソースごとにクラスとして実装する（推奨）
- FHIRのオブジェクトモデルを操作して、データベースのレコード情報から対応するFHIRリソースのオブジェクトに変換する
  - 今回は患者マスターからPatientオブジェクトを生成
- 必要に応じて、他のマスターを検索して、コードに対する名称を取得したり、ローカルコードから標準コードへの変換を行ったりする（例：薬剤コード）

Converterクラスの実装例①

```
static public Patient CreatePatientResource(SqlSession session, MyPatient table){
   Patient resource = new Patient();
   // id
   resource.setId(table.getPt_id());
   //  identifier
   resource.addIdentifier()
      .setValue(table.getPt_id())
      .setSystem(Common.GetNSPatient());
   //  name
   resource.addName(Common.CreateHumanName(table.getPt_family_ide(),
     table.getPt_given_ide(), "IDE"));
   resource.addName(Common.CreateHumanName(table.getPt_family_syl(),
     table.getPt_given_syl(), "SYL"));
   //telecom
   resource.addTelecom()
       .setUse(ContactPointUse.HOME)
       .setSystem(ContactPointSystem.PHONE)
       .setValue(table.getPt_phone());
```

Patientオブジェクトを生成する

MyBatis用の患者マスターオブジェクト

患者IDをリソースIDとしてセットする

患者IDをセットする

患者漢字氏名をセットする

患者カナ氏名をセットする

電話番号をセットする

Converterクラスの実装例②

```
   //gender
   switch(table.getPt_gender()) {
   case "M" : resource.setGender(AdministrativeGender.MALE); break;
   case "F" : resource.setGender(AdministrativeGender.FEMALE); break;
   case "O" : resource.setGender(AdministrativeGender.OTHER); break;
   default  : resource.setGender(AdministrativeGender.UNKNOWN); break;
   }
   //birthDate
   resource.setBirthDateElement(new DateType(table.getPt_birthdate()));
   //address
   resource.addAddress()
      .setText(table.getPt_address())
      .setPostalCode(table.getPt_zip());

  return resource;
}
```

性別をセットする
（例：female）

生年月日をセットする
（例：1990-10-10）

住所、郵便番号をセットする

生成したPatientオブジェクトを返す

HumanNameの生成

```
public static HumanName CreateHumanName(String family, String given, String
style) {
  HumanName humanName = new HumanName()
    .setText(family + " " + given)
    .setUse(NameUse.OFFICIAL)
    .setFamily(family)
    .addGiven(given);
  // Create an extension
  if (style != null) {
    humanName.addExtension()
      // "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation"
      .setUrl(GetSDIso21090ENRepresentation())
      .setValue(new CodeType(style));
  }

  return humanName;
}
```

textをセットする
（例：福岡 千尋）

familyをセットする
（例：福岡）

givenをセットする
（例：千尋）

EN Representation
拡張を追加する
（例：IDE）

生成したPatientリソースの例

```
{
  "resourceType": "Patient",
  "id": "1234567890",
  "meta": {
    "profile":
[ "http://jpfhir.jp/fhir/core/StructureDefinition/JP
_Patient" ]
  },
  "identifier": [ {
    "system":
"http://terminology.sample.com/IdSystem/patient",
    "value": "1234567890"
  } ],
  "active": true,
  "name": [ {
    "extension": [ {
      "url":
"http://hl7.org/fhir/StructureDefinition/iso21090-
EN-representation",
      "valueCode": "IDE"
    } ],
    "use": "official",
    "text": "福岡 千尋",
    "family": "福岡",
    "given": [ "千尋" ]
  }, {
    "extension": [ {
      "url":
"http://hl7.org/fhir/StructureDefinition/iso21090-
EN-representation",
      "valueCode": "SYL"
    } ],
    "use": "official",
    "text": "フクオカ チヒロ",
    "family": "フクオカ",
    "given": [ "チヒロ" ]
  } ],
  "telecom": [ {
    "system": "phone",
    "value": "03-6252-2220",
    "use": "home"
  } ],
  "gender": "female",
  "birthDate": "1990-10-10",
  "address": [ {
    "text": "東京都港区東新橋1－5－2",
    "postalCode": "105-7123",
    "country": "JP"
  } ]
}
```

リソースID

患者ID

患者漢字氏名

患者カナ氏名

電話番号

性別・生年月日

住所

郵便番号

まとめ

- **PDQmを題材にHAPI FHIRによるFHIRサーバーの実装の概要**について紹介した
- PDQmでは、FHIR R4での**Patientリソースの各種検索と取得**に対応する必要がある
- HAPI FHIRでは、リソース等の**オブジェクトモデルやRESTサーバー機能、検索パラメータの解析、**オブジェクトから**XML/JSONへの変換**、などの機能が用意されている
- アプリ側では、**DBの検索とリソースモデルへの変換**という、**本質的な処理だけ**実装すればよい

ご清聴ありがとうございました。

ご質問は、
日本IHE協会ホームページまで。