HL7 FHIRハンズオン

Pythonを利用し FHIR REST APIを操作する

第27回医療情報学春季学術大会　チュートリアル9
日本Mテクノロジー学会主催

群馬大学医学部附属病院 システム統合センター　鳥飼　幸太
熊本大学病院 総合臨床研究部 研究データ管理センター　山ノ内　祥訓
東京大学医学部附属病院 企画情報運営部　土井　俊祐

ファイルのダウンロードついて

チュートリアル申し込みページからリンクを用意しました。
https://www.mta.gr.jp/tutorial/index.html

注意事項

- 会場Wi-fiを利用した大容量のデータのダウンロード、ポケットWi-fiのご利用はお控え下さい。
  - 配布プログラムはチュートリアルサイトで公開しておりますが、現地会場では受付でUSBメモリを貸し出しております。

- ZOOM Webinarは録画しております。
  - 当会会員及び参加者向けのオンデマンド配信のために利用します。

- 現地・オンラインともに会場のサポート要員が十分でない可能性があります。
  - 大会期間後もプログラムが正常に動作するまでサポートを行います。
  - 環境によりエラーが起きた場合は、修正プログラムを後日配布いたします。

このチュートリアルの目的

- **HL7 FHIRの構成要素について理解し、Pythonを利用してFHIRのJSONフォーマットを操るための基本的なコーディングスキルを身につける。**
  - HL7 FHIR・JSON、FHIR REST APIとは何か
  - Pythonを利用したコーディングの基本
- **Pythonライブラリを利用した簡単なWebサービスを作る。**
  - HL7 FHIRのJSONデータを、REST APIで利用される各メソッド（GET/POST/PUT/PATCH/DELETE）で操作するプログラムを参加者自身の環境で実装する。
  - 上記をWebサービスとして実装する。
- **上記の取り組みを通して、HL7 FHIRの活用について具体的な活用方法・メリットについて実感を得られること。**

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - ハンズオン2：患者情報（Patient）を操作する
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - ハンズオン4：文書情報（FHIR Document）の操作

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

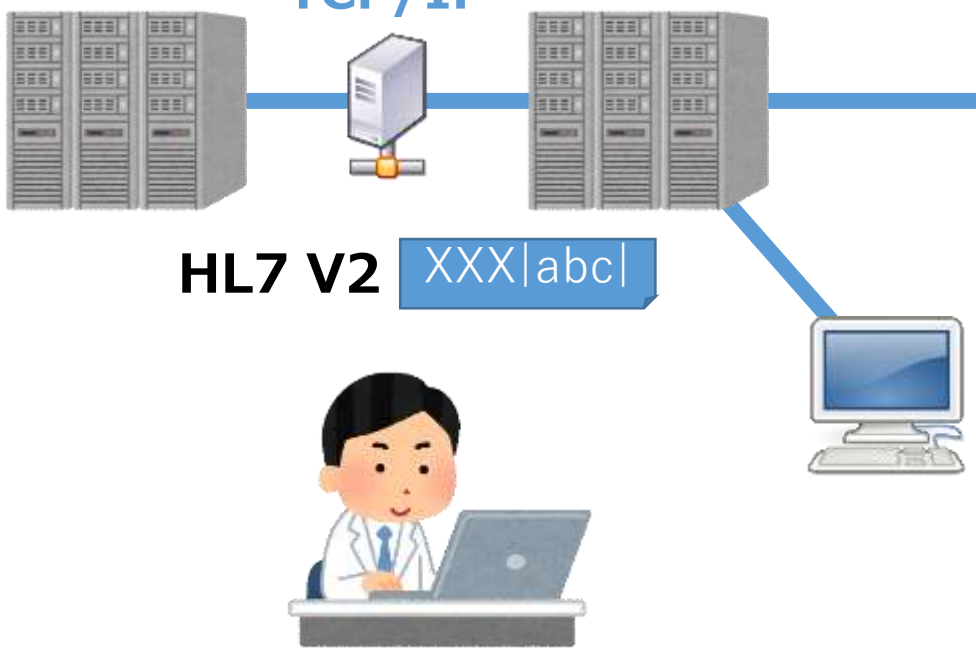
- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - ハンズオン2：患者情報（Patient）を操作する
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - ハンズオン4：文書情報（FHIR Document）の操作

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

医療情報通信のこれから

これまでの情報通信のありかた

CAT5/6/7
TCP/IP



HL7 V2 XXX|abc|

スタッフが情報の側に移動する
共通の端末を利用する
サーバ間だけ標準規格で通信できていれば良い

これからの情報通信のありかた

IEEE802.11
a/b/g/n/ac
Mobile Network
HTTP

スタッフが情報とともに移動する
個人用（または目的ごと）デバイスを利用
デバイスやアプリを問わない通信手段

HL7 FHIRとはどんな規格か

Fast（短期間で開発・導入が可能）

Health（健康・医療分野が対象）

Interoperable（相互運用が可能な）

Resources（リソース）

仕様が複雑で実装時に多様性が生じるHL７ver3に対して、
簡単な実装を重視した規格の策定が進んでいる。

これまでの規格と何が違うのか

- **FHIRが優れている理由**

誰でも参加しやすい、参入しやすい

  - **実装性に強力にフォーカス：速く、簡単に実装できる**
    （複数の開発者がたった1日で簡単なインターフェイスを構築できた例もある）
    クイックスタートを可能とする多くの実装ライブラリが準備されている

Web標準＝医療特化の脱却

  - **RESTfulアーキテクチャ**、メッセージとドキュメントを使用したシームレスな情報交換で技術開発社にとって学習障壁が低い

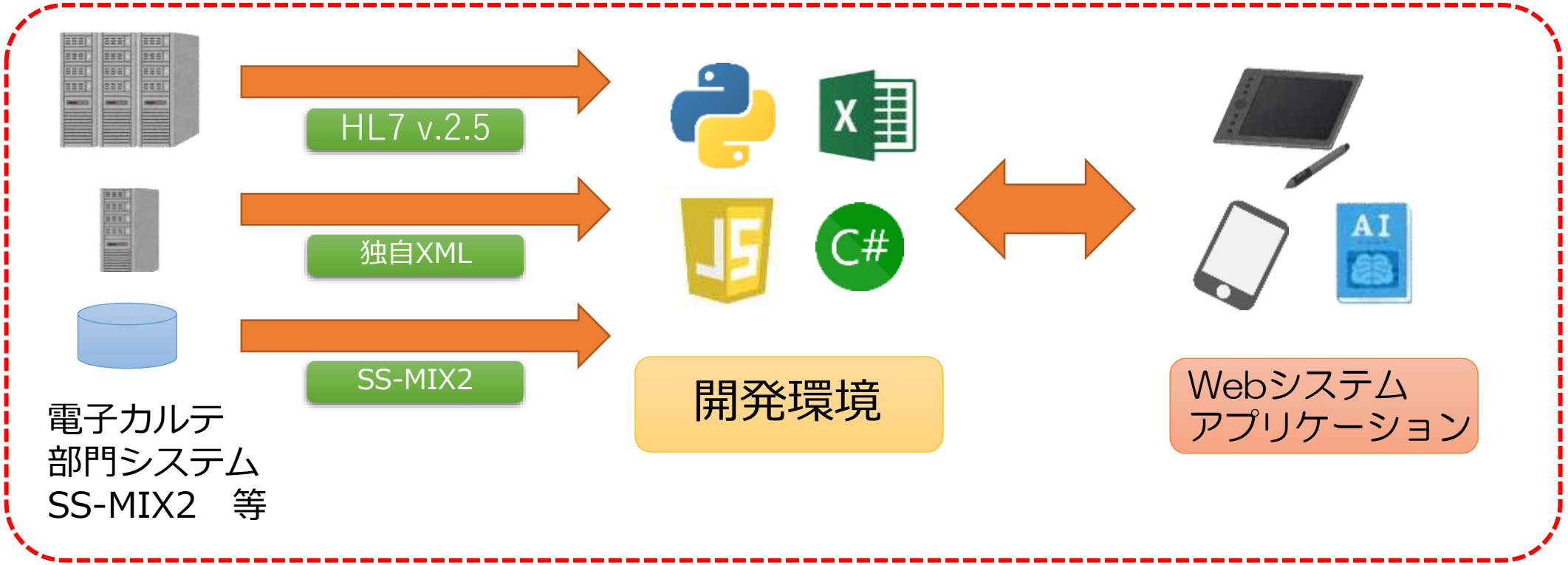
既存の資産も活用できる

  - **革新的な相互運用性**：ベースとなるリソースは用意されているが、既に利用されているプロファイルやエクステンション、用語集などを採用できる
  - **既存のHL7 v.2、CDAとは相互互換**があり、**両方から発展的に活用できる**

HL7 FHIR ポータルサイト「2.17.1 Why FHIR is better」を一部抜粋し翻訳（https://www.hl7.org/fhir/summary.html）

HL7 FHIRの導入が医療情報分野にもたらすもの

・サービスを現場に導入するまで（従来）

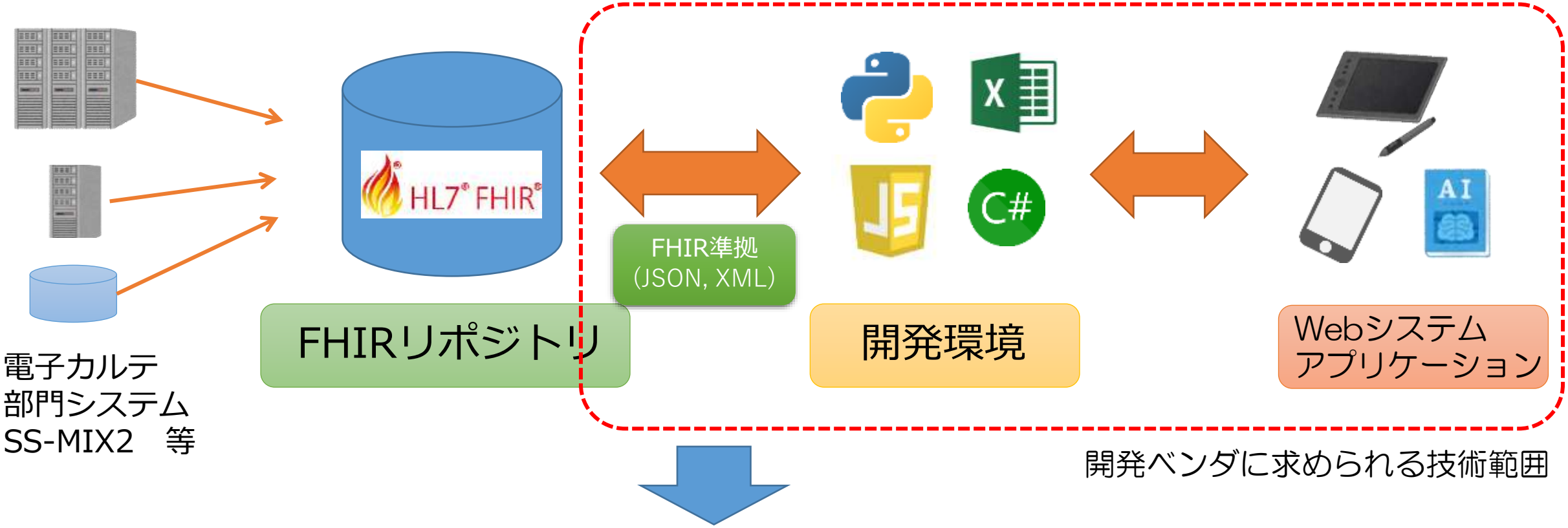


開発ベンダに求められる技術範囲

連携元のデータ構造や医療情報の通信規格も理解しなければならず、医療情報関係の開発経験の少ないベンダにとっては参入の障壁はとても高い。

HL7 FHIRの導入が医療情報分野にもたらすもの

・サービスを現場に導入するまで（FHIRを利用）

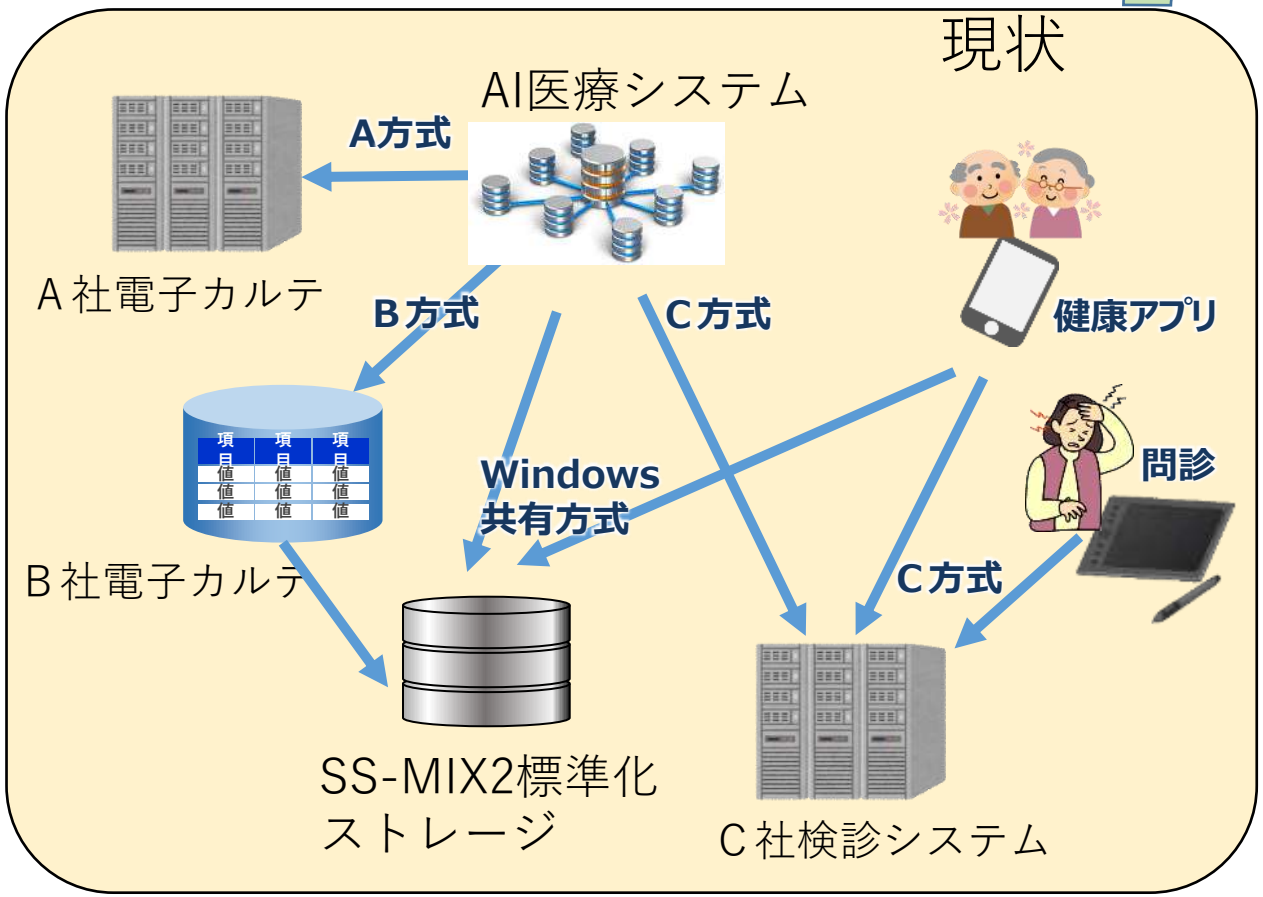


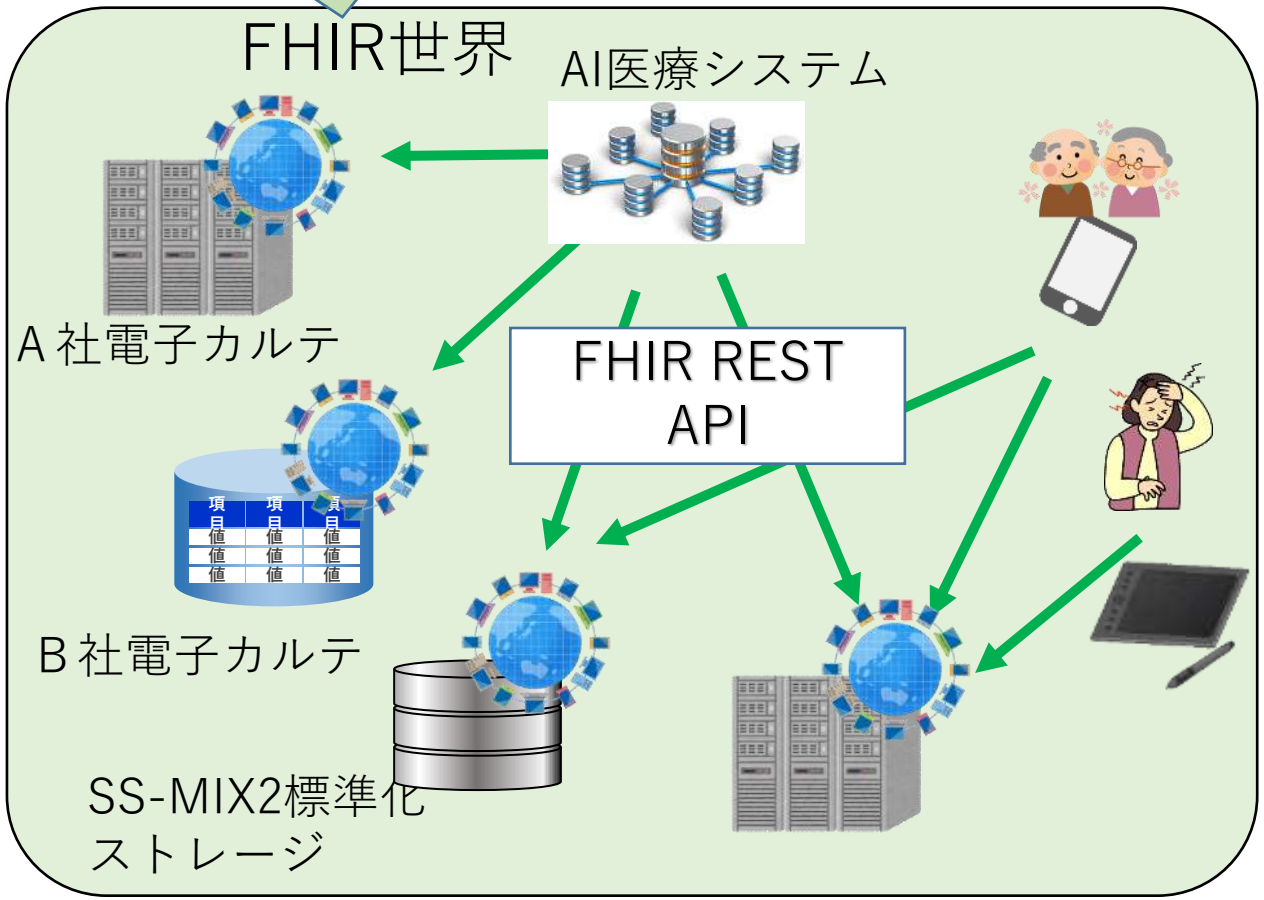
「医療情報のデータ交換には独特のアプローチが必要」という既成概念を覆す可能性
→より広範なベンダの参入を促し、ひいてはより良いサービス構築につながる！！

共通API（FHIR REST API）を利用する世界へ

データ型式や内容が標準化されている場合でも、
データの出し入れの方式（ＡＰＩ）はバラバラ
のため、各データシステムごとに開発が必要で、
そのための技術障壁が高い。

FHIR REST API を備えた
アダプタまたはレポジトリ
を装備すれば、利活用側は
既存のWebアクセスでＯＫ





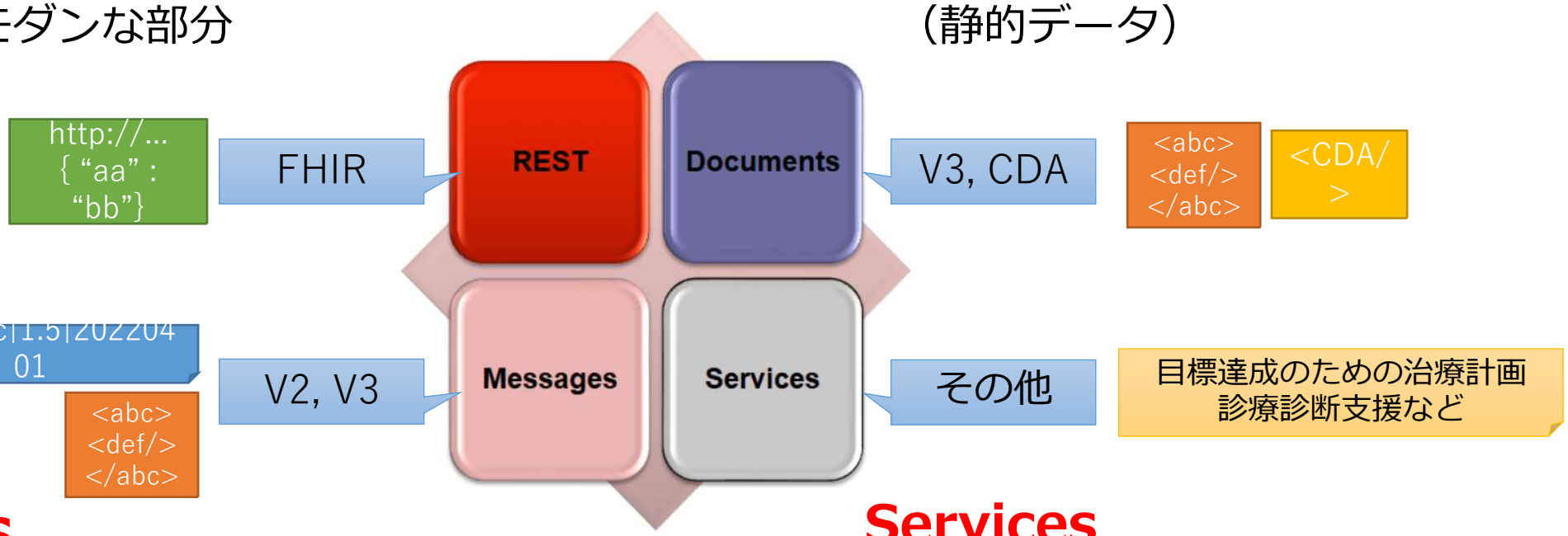
FHIRの4つのパラダイム

RESTfulインターフェースの利用

- Web技術を使用することの宣言
- RESTによりデータ交換をする、FHIRの最もモダンな部分

Documents

- HL7 CDAで実現したアーキテクチャ
- 1時点のリソースをパッケージ化（静的データ）



Messages

- HL7 Ver.2で実現したアーキテクチャ
- リソースをメッセージにパッケージ化し、システム間でプッシュする（動的データ）

Services

- FHIRからの新しい概念。メッセージで実現しているもの「以外」
- 診療診断支援など、医療スタッフの行動を1単位とする情報概念

RESTとは

- **REST(REpresentational State Transfer)**
  - Web開発において最も一般的な通信技術の1つ

RESTサービスの設計原則

1. **アドレス指定可能なURIで公開されていること**
2. **インターフェース(HTTPメソッドの利用)の統一がされていること**
3. **ステートレスであること**
4. **処理結果がHTTPステータスコードで通知されること**

FHIRのRESTで使用するURLの例

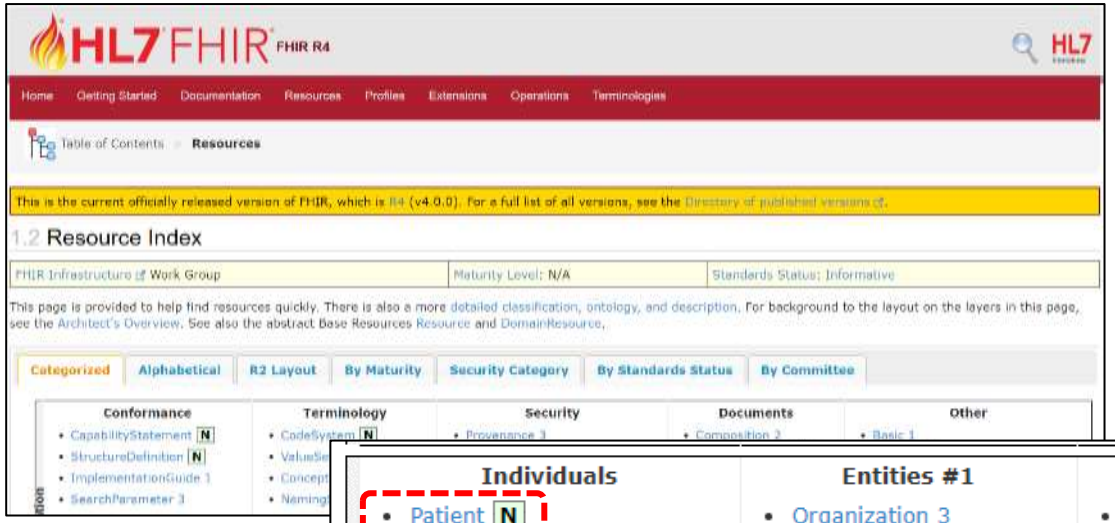
"http://hapi.fhir.org/baseDstu3/Patient/1646554/_history/1?_format=json"

エンドポイント（サーバごと固定される部分）　リソース名とID　引数（取得するデータの条件指定）

FHIRでは、**リソース**と呼ばれる同じ性質を持つ情報の断片（例：Patient, Observation）と、**リソースを一意に識別するURI**で表されるアドレスを持つ

FHIRリソースとは

- **FHIRリポジトリ（FHIR準拠のデータ格納庫（データベース））に保存される、一定のデータ構造をもった医療情報のかたまりそのもの**



Normativeとされたもの以外は、
FHIR Maturity Modelのレベルを表す。
値が大きいほど成熟度が高い。

Patient
（患者基本情報）
のリソース

Observation
（検査）
のリソース

Base

Individuals	Entities #1	Entities #2	Workflow	Management
• Patient N	• Organization 3	• Substance 2	• Task 2	• Encounter 2
• Practitioner 3	• OrganizationAffiliation 0	• BiologicallyDerivedProduct 0	• Appointment 3	• EpisodeOfCare 2
• PractitionerRole 2	• HealthcareService 2	• Device 2	• AppointmentResponse 3	• Flag 1
• RelatedPerson 2	• Endpoint 2	• DeviceMetric 1	• Schedule 3	• List 1
• Person 2	• Location 3		• Slot 3	• Library 2
• Group 1			• VerificationResult 0	

Clinical

Summary	Diagnostics	Medications	Care Provision	Request & Response
• AllergyIntolerance 3	• Observation N	• MedicationRequest 3	• CarePlan 2	• Communication 2
• AdverseEvent 0	• Media 1	• MedicationAdministration 2	• CareTeam 2	• CommunicationRequest 2
• Condition (Problem) 3	• DiagnosticReport 3	• MedicationDispense 2	• Goal 2	• DeviceRequest 1
• Procedure 3	• Specimen 2	• MedicationStatement 3	• ServiceRequest 2	• DeviceUseStatement 0
• FamilyMemberHistory 2	• BodyStructure 1	• Medication 3	• NutritionOrder 2	• GuidanceResponse 2
• ClinicalImpression 0	• ImagingStudy 3	• MedicationKnowledge 0	• VisionPrescription 2	• SupplyRequest 1

FHIRリソースとは

米国HL7協会がリソースの定義を行ったもの
→これとは別に日本版仕様（JP-CORE）が検討されている（後述）

Patientリソースの構造（HL7 FHIR公式）

エレメント名（Name）
データ構造は基本的に階層型になっており、json、XMLで表すのに適した構造になっている

多重度（Cardinality）
「0…1」は、データを格納しないか1つまで格納することができる。
「1…1」や「1…*」はであれば必ず1つ以上のデータを格納しなければならない。

データ型（Type）
データ型はそれぞれ定義されており、数値やテキストのみでなく、他のリソースを参照する「Reference」や、既存のコード体系を参照する「CodeableConcept」などもある

FHIRリソースのデータ構造

- **FHIRインスタンスの例**

患者IDを表す部分

患者氏名を表す部分

```
{
  "resourceType": "Patient",
  "id": "example",
  "text": {
    "status": "generated",
    "div": "text"
  },
  "identifier": [
    {
      "system": "urn:oid:1.2.392.999999.20.3.51.1[保険医療機関コード(10 桁)]",
      "value": "0001234567"
    }
  ],
  "active": true,
  "name": [
    {
      "extention": [
        {
          "url": "http://hl7.org/fhir/StructureDefinition/iso21090-ENrepresentation"
          "valueCode": "IDE"
        }
      ],
      "use": "official",
      "text": "東京太郎",
      "family": "東京",
      "given": [
        "太郎"
      ]
    },
```

Patientリソースの作成例。
先ほどのStructureに準じたJSON形式の階層構造で表されている。

JSONとは

- JavaScript Object Notation
- Web開発ではHTMLと並び非常に一般的なデータフォーマット
- 中カッコとコロンで要素名とインスタンス名を区切って表現する

例：

{“name”：“宮本武蔵”}

FHIRの4つのパラダイムと医療情報通信

モバイル端末
PHR

REST API

FHIRリポジトリ

Document

診療情報提供書
退院時サマリー

文書管理
システム

HL7 CDA

検査結果
実施入力

部門システム

HL7 V2

Message

電子カルテ
システム

Service

診療診断支援
システム

CDS HOOKS™

Bulk API

医学知識を提供

FHIR ゲートウェイ
FHIR アダプタ

AI医療システム

標準化まわりの政策的な動向（令和4年度診療報酬改定）

令和４年度診療報酬改定　Ⅱ－５　業務の効率化に資するICTの利活用の推進、その他長時間労働などの厳しい勤務環境の改善に向けての取組の評価－③

標準規格の導入に係る取組の推進

診療録管理体制加算の見直し

➢ 医療機関間等の情報共有及び連携が効率的・効果的に行われるよう、標準規格の導入に係る取組を推進する観点から、電子カルテの導入状況及びHL7 Internationalによって作成された医療情報交換の次世代標準フレームワークであるHL7 FHIR(Fast Healthcare Interoperability Resources)の導入状況について報告を求めることとする。

改定後

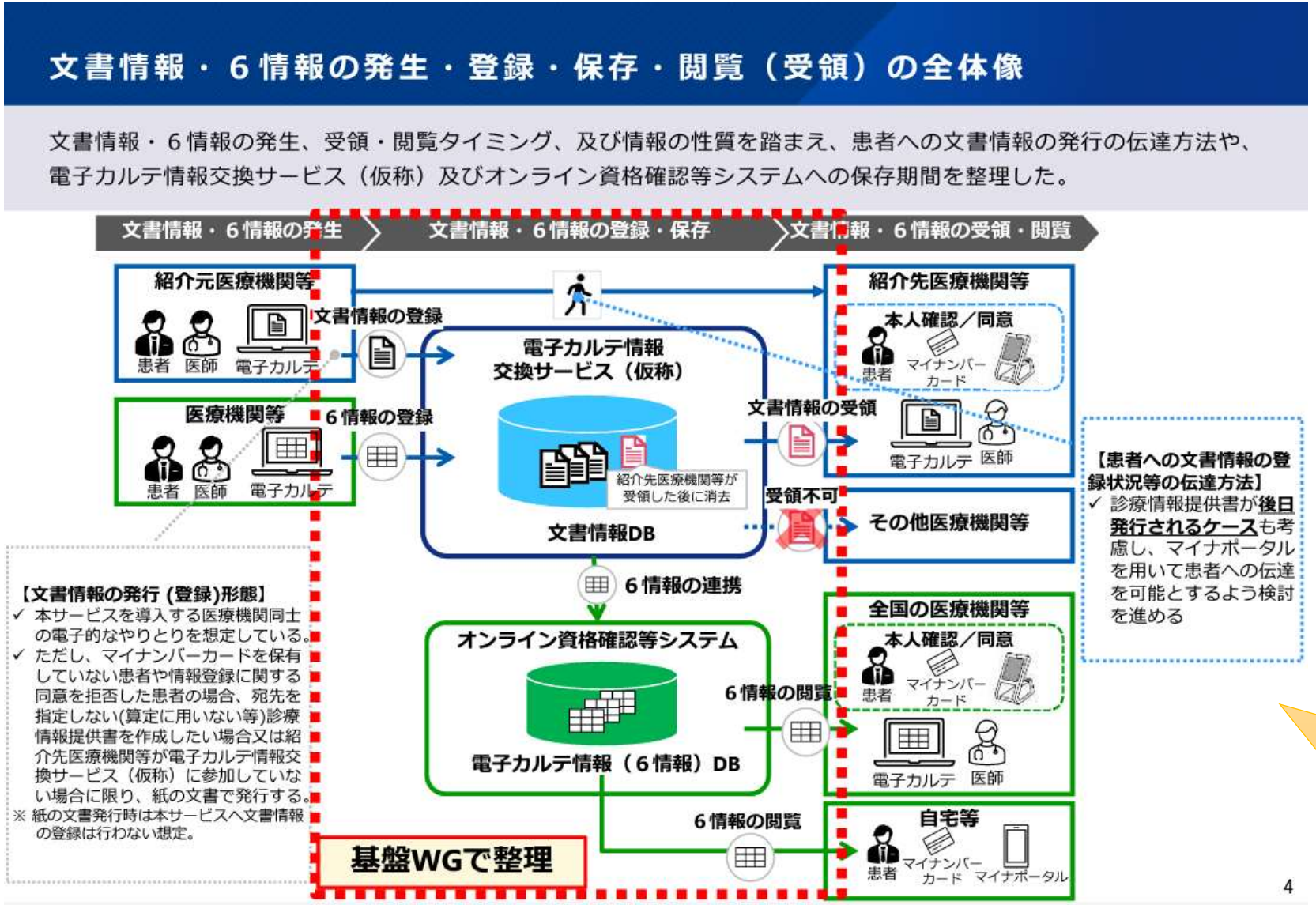
【診療録管理体制加算（入院初日）】
［施設基準］
3　届出に関する事項
(1) 診療録管理体制加算の施設基準に係る届出は、別添７の様式17を用いること。
(2) 毎年７月において、標準規格の導入に係る取組状況等について、別添様式により届け出ること。

HL7 FHIRの導入に係る取り組み状況について届け出ること（様式17の2）

⑥　標準規格（HL7 FHIR）への対応予定	診療情報提供書	□対応予定 （　　年　　月目途） □対応予定なし
	退院時要約	□対応予定 （　　年　　月目途） □対応予定なし

令和4年度診療報酬改定の概要　https://www.mhlw.go.jp/content/12400000/000920422.pdf

標準化まわりの政策的な動向（厚生労働省資料）



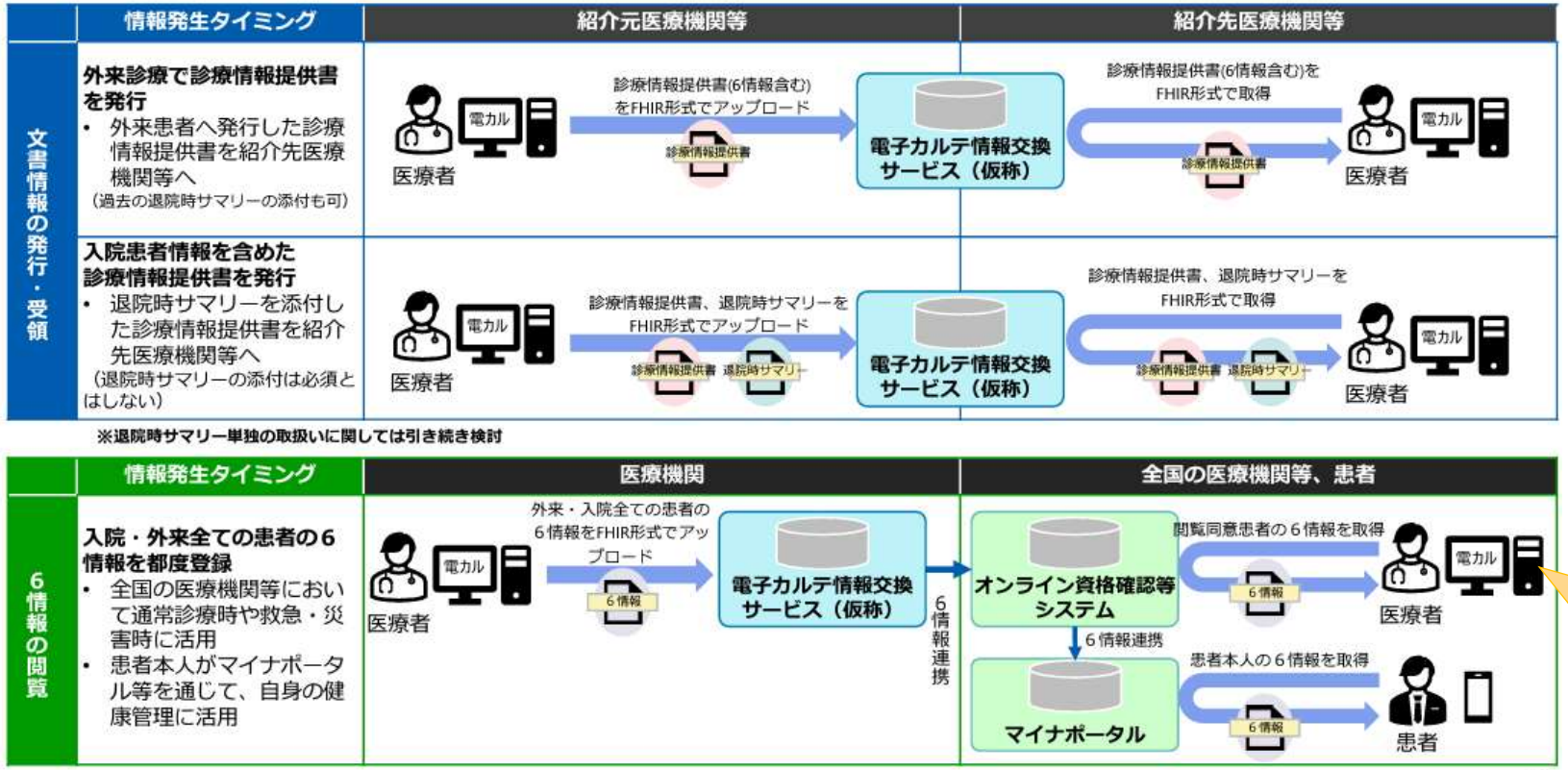
電子カルテ情報交換サービスにおける情報交換の仕様として、HL7 FHIRを利用する案が明記されている

第７回健康・医療・介護情報利活用検討会医療情報ネットワークの基盤に関するWG（令和５年３月９日）資料１－３参考資料
https://www.mhlw.go.jp/content/12600000/001090094.pdf

標準化まわりの政策的な動向（厚生労働省資料）

基盤を構築する上での情報の発生タイミング・登録の仕組みに関する想定

①電子カルテ情報を使用している全国の医療機関等との連携（医療機関同士での救急・災害時に有用となる情報や生活習慣病関連の情報の交換等）、②患者自身による自らの医療情報の活用等のための基盤となることを想定している。

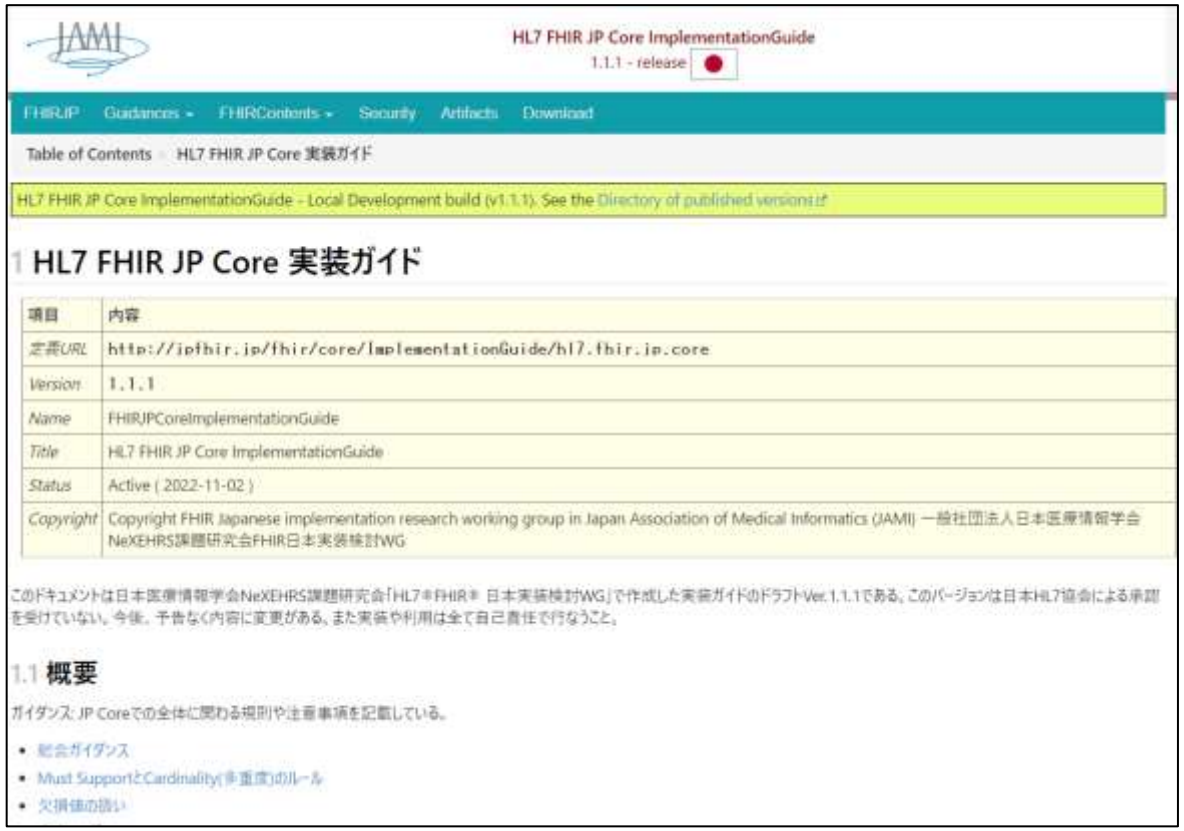


電子カルテ情報交換サービスにおける情報交換の仕様として、HL7 FHIRを利用する案が明記されている

第７回健康・医療・介護情報利活用検討会医療情報ネットワークの基盤に関するWG（令和５年３月９日）資料１－３参考資料
https://www.mhlw.go.jp/content/12600000/001090094.pdf

日本でFHIRを普及させるための「ルールづくり」

- **FHIRは実装性を主眼にして開発された規格である**
  - FHIRは**80%のシステム**が使用するであろう項目を標準で収載（80%ルール）
  - 逆に残りは自由に実装できる
- **各国が自国の制度に合ったFHIRの標準仕様を策定している**
  - それぞれの仕様を説明するために「実装ガイド」を策定している
  - 日本も**JP CORE**を策定中で、2021年末にDraft Ver.1が公開された
  - 現在は、**2022年11月2日**に公開された**V1.1.1**が最新版



https://jpfhir.jp/fhir/core/1.1.1/index.html

日本医療情報学会NeXEHRS研究会（正式名称：次世代健康医療記録システム共通プラットホーム課題研究会）

HL7®FHIR® 日本実装検討WG

■ FHIR WG情報

2022.04.07　2019.07.12

■ セミナー資料（2022.3.3）

PDF資料

■ 全体WG開催日程（当面ZOOMのみ）

第26回： 4/21(木) 10:00-12:00
第25回： 3/17(木) 13:00-15:00
第24回： 2/16(水) 10:00-12:00

ZOOMアクセス情報はSlack(hl7fhir-jp-wg.slack.com)
#generalを参照ください（会員限定：下記に参加登録申込み必要）

■ HL7FHIR® HL7 FHIR Jp Core 実装ガイド

▶ **JP CORE DRAFT V1 を公開しました。**
▶ **日本HL7協会により公認されました。**
▶ **FHIR公式レジストリ（https://registry.fhir.org/package/JP-CORE.Draft%7C1.0.5）にも登録されています。**

公式Webサイト： https://jpfhir.jp/jpcoreV1
Simplifier.net ： https://simplifier.net/guide/jpcorev1/fhirjp
GitHUB： https://github.com/jami-fhir-jp-wg/jp-core-draft.git

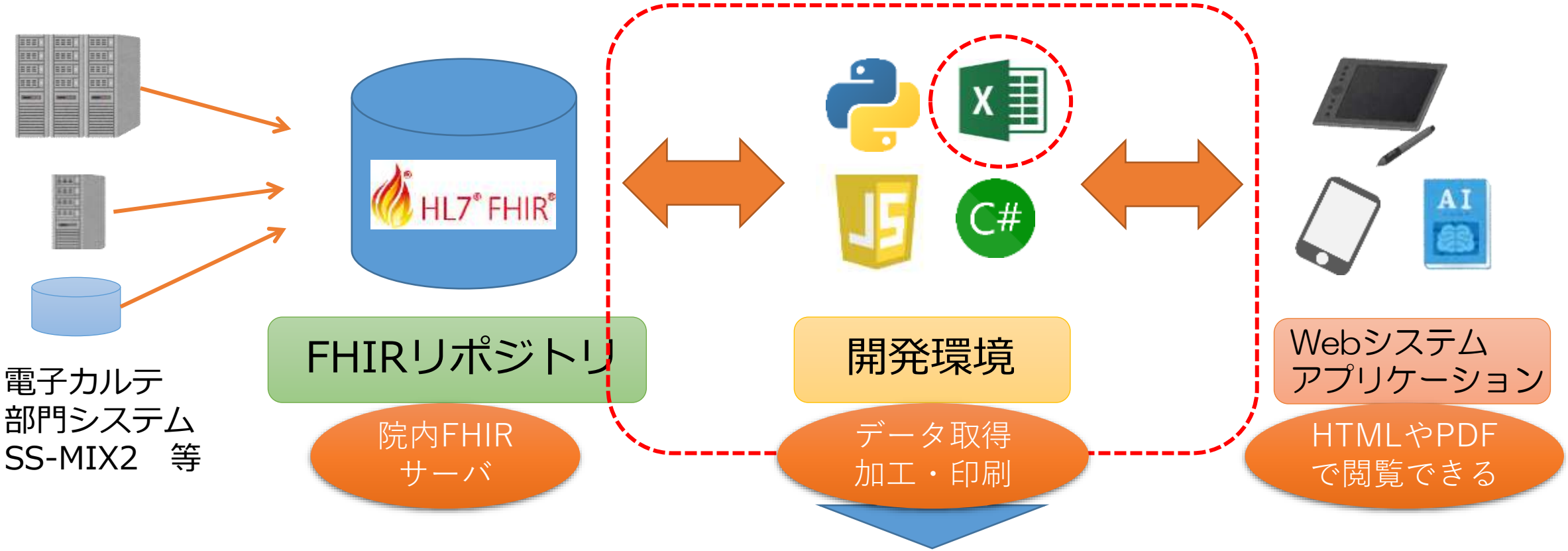
【情報一覧】

https://jpfhir.jp/

**100名超の有志が日本版の
HL7 FHIR仕様(JP CORE)を検討中
（随時メンバー募集中）**

本日のチュートリアルの内容

・FHIRデータの取得・表現・加工の方法をVBAで学ぶ！



今日やるのは主にココ
開発の敷居の低さを体感してもらおう！

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - **環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）**
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - ハンズオン2：患者情報（Patient）を操作する
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - ハンズオン4：文書情報（FHIR Document）の操作

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

Pythonのインストール（未インストールの方のみ）

- Python 3.7以降（推奨は3.9 or 3.10）をインストールして下さい

「Download」からOSを選択してクリックし、表示されたページを下の方にスクロール

https://www.python.org/

※Macの方はMac OS用を利用して下さい

- Python 3.10.11 - April 5, 2023
  - Download macOS 64-bit universal2 installer

M1チップ搭載機などApple Siliconを利用している方はuniversal2 installerを利用してください。

- Python 3.10.11 - April 5, 2023

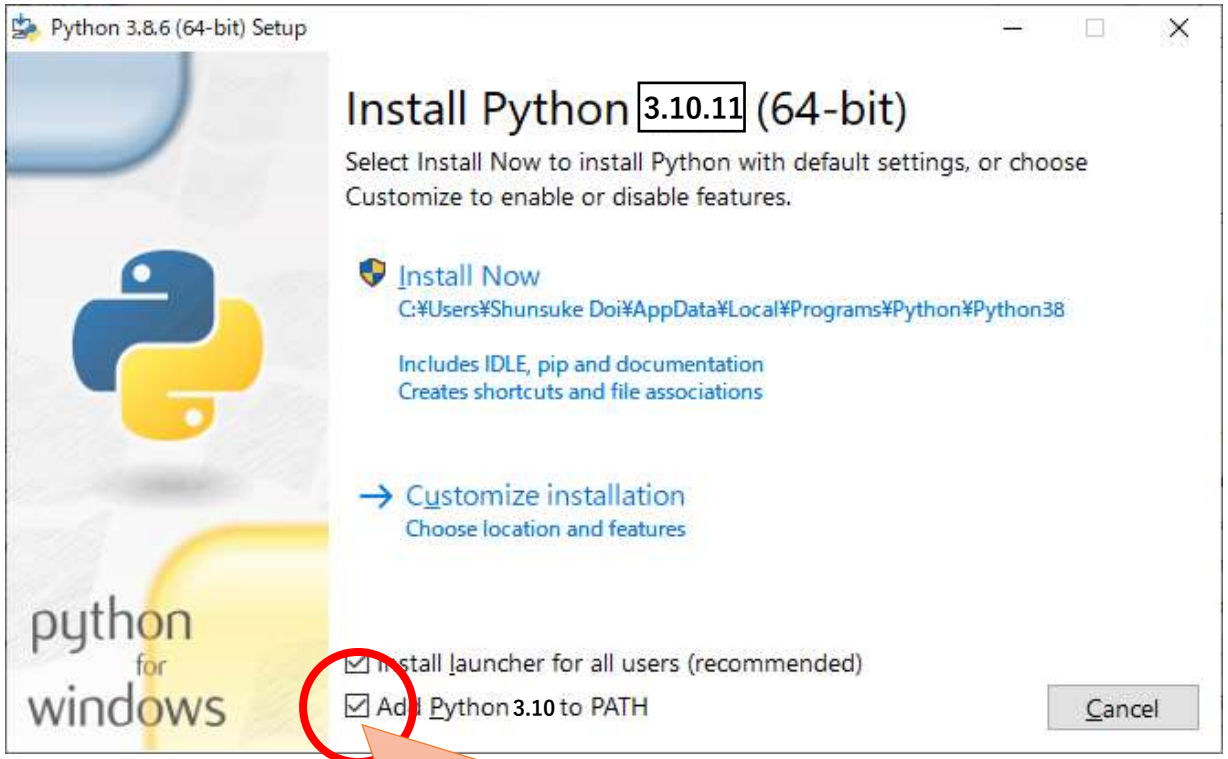
Note that Python 3.10.11 *cannot* be used on Windows 7 or earlier.

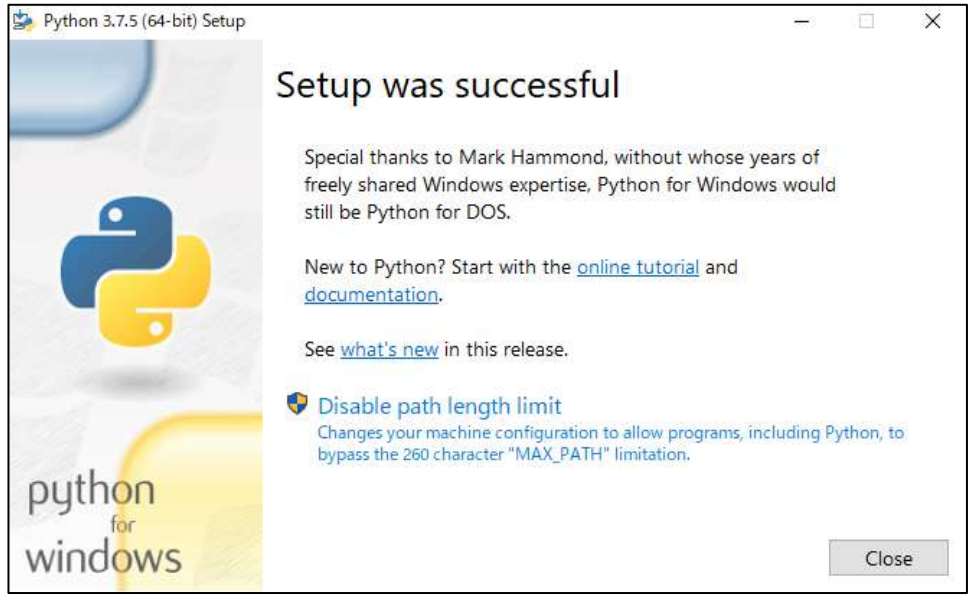
- Download Windows embeddable package (32-bit)
- Download Windows embeddable package (64-bit)
- Download Windows help file
- Download Windows installer (32-bit)
- Download Windows installer (64-bit)

- Download Windows installer (32-bit)
- Download Windows installer (64-bit)

Windowsの方は上記のインストーラをダウンロードいただき、ダブルクリックで実行して下さい。
（ご自身のOSが32bitか64bitかをご確認の上で実行してください。）

（事前ダウンロード）Pythonのインストール





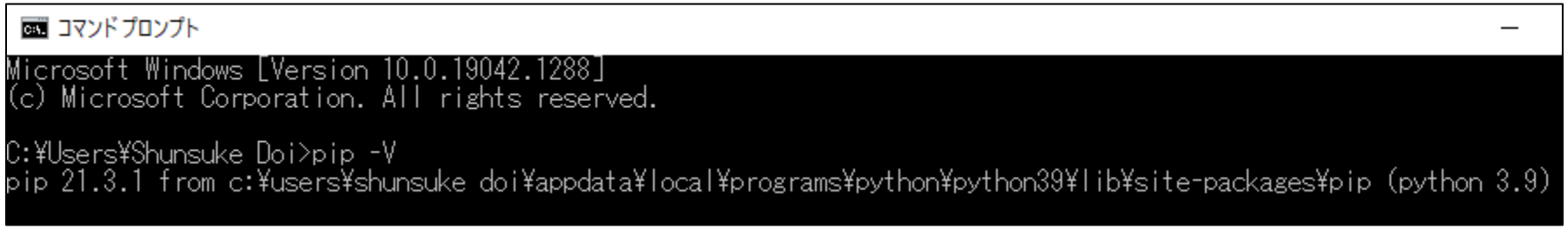
※重要※
必ずチェックを入れておいて下さい。

これでインストールは完了です。

- **他のバージョンでも動作するものと思いますが、全環境への動作保証はしておりません。**
- **また、最新版の3.12.0も動作未検証ですのでその点ご了承下さい。**

pipのインストール確認

- 本チュートリアルでは、Pythonのライブラリを利用するためpipを利用します。
- pipがインストールされているかは「pip -V」のコマンドで確認できます。

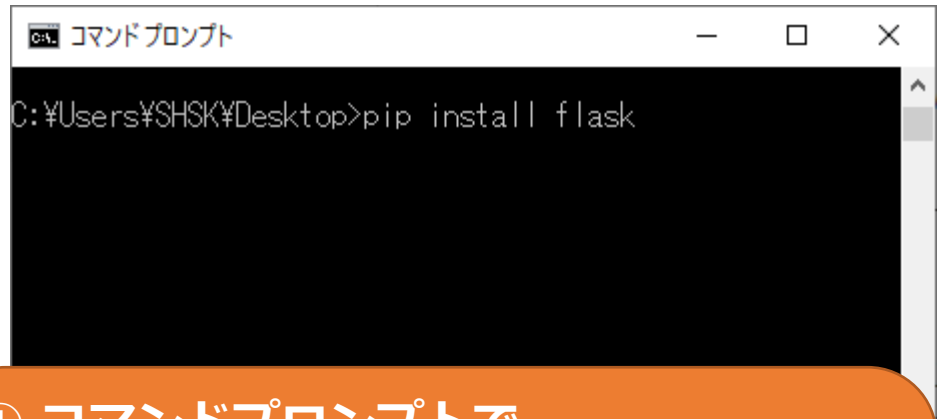


- pipのバージョンが古い場合は、「python –m pip install --upgrade」でアップデートすることができます。

※pipがインストールされていない方は、一度pythonをアンインストールし、3.10系のpythonをダウンロードしてインストールし直して下さい。（ver.3.4 以降は標準で内包されています）

ライブラリのインストール

- 本チュートリアルでは、PythonのWebフレームワークとして、「Flask」というライブラリを使用します。
- pipコマンドを使用することで簡単にインストール可能です。
- 同様に「requests」「datetime」もインストールします。

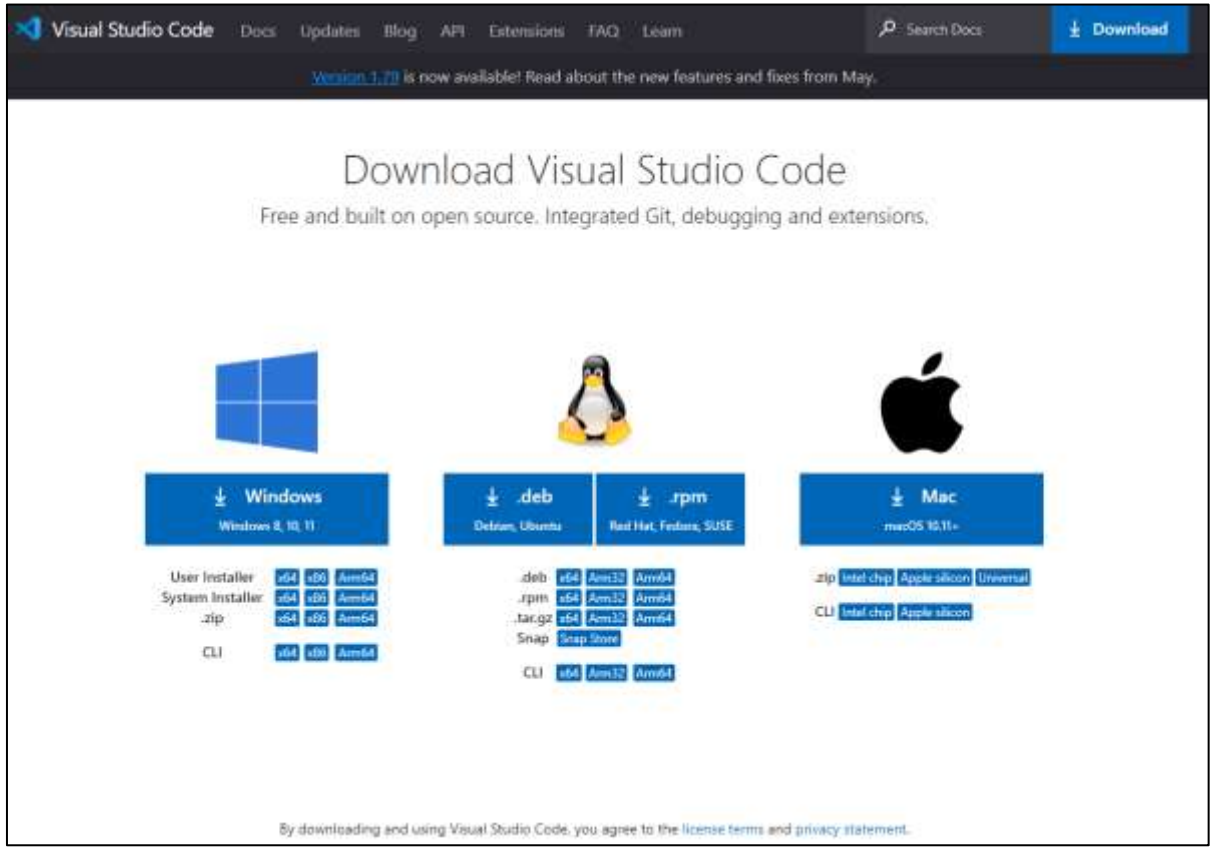


① コマンドプロンプトで
「pip install flask」と入力し実行します。
（ここはどのフォルダで実行してもOK）
※MACの方、2.x系もインストールしている方は、「pip3 install flask」と入力して実行して下さい。

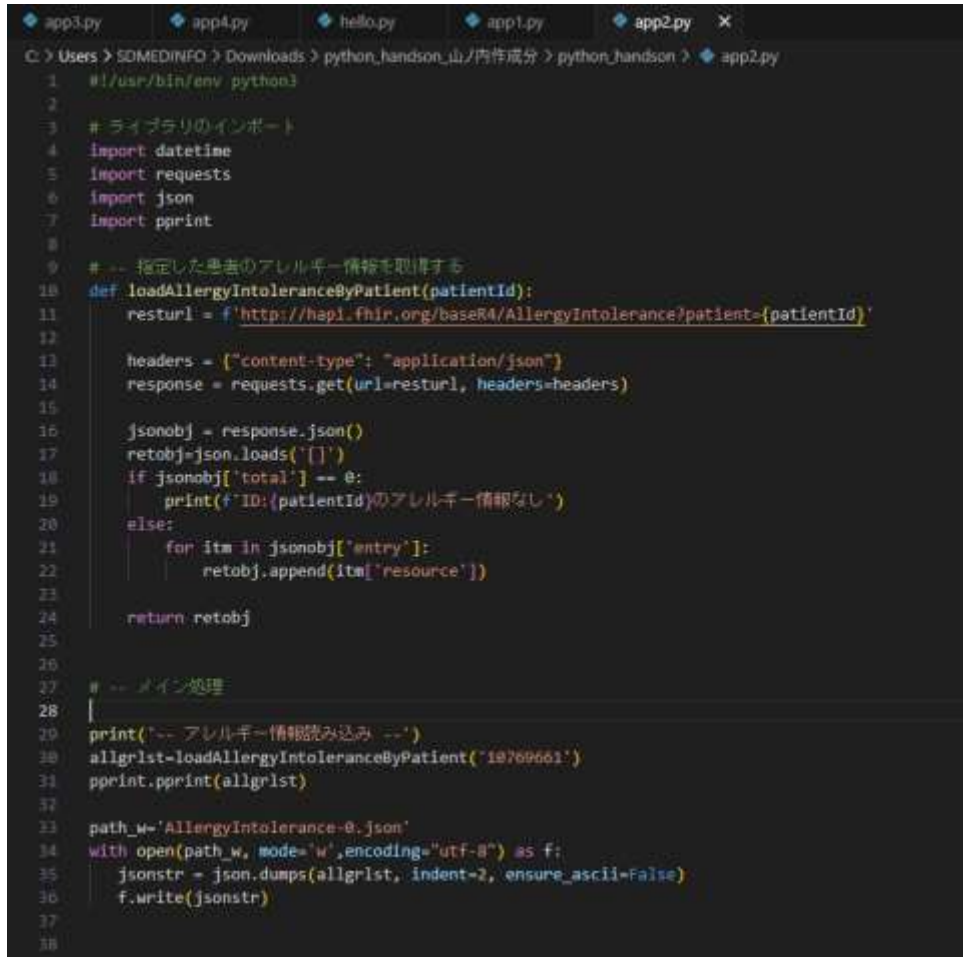
② メッセージが表示されインストールが進行します。
同様に「pip install requests」も実行します。

ハンズオン準備（VS codeの紹介）

- Pythonのエディタは多数ありますが、無料かつ予測入力などの機能が充実しているVisual Stadio Code(VS code)をお勧めします



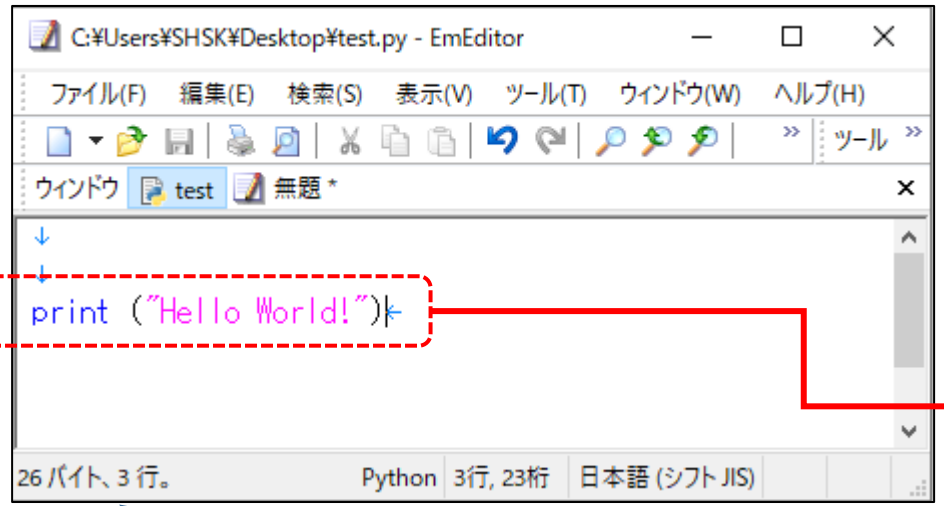
https://code.visualstudio.com/download

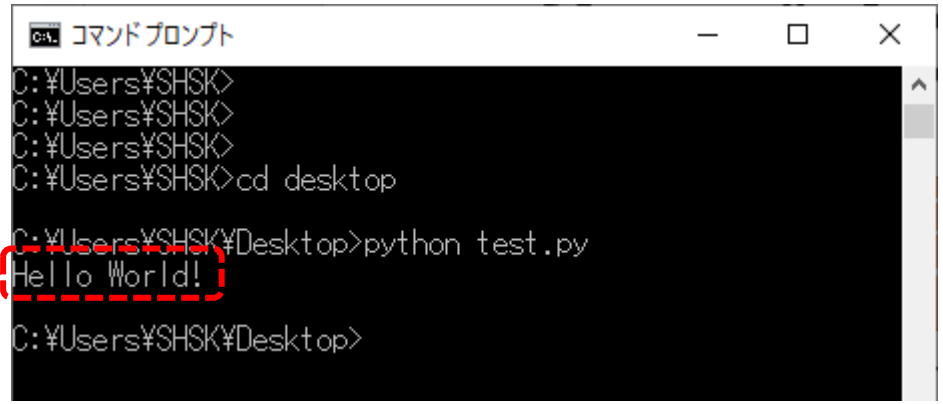


ハンズオン準備（Pythonの基本的な動作方法の説明）

① テキストエディタ等でコードを編集し、「○○.py」のファイル名で保存します

② 保存したPythonのファイルをコマンドから「python ○○.py」の形で実行します。
※MACの方、2.x系もインストールしている方は、「python -3 ○○.py」と入力して下さい。





エディタは各自の環境をご使用下さい。

Windowsの方はTeraPadが標準でインストールされています。

Windowsの方はスタートボタンをクリック後に「cmd」と入力しEnterを押下するとコマンドプロンプトが起動します。
(Macではターミナルというアプリ)

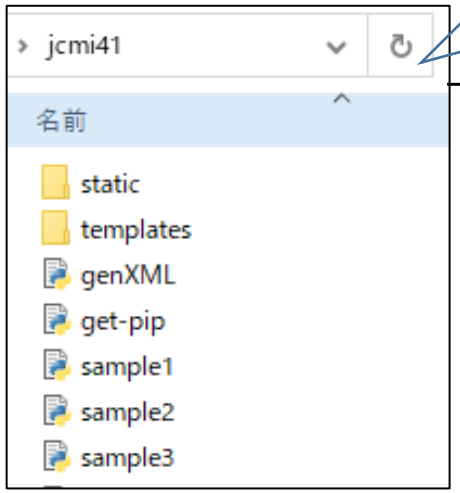
ハンズオン準備（作業用フォルダと環境の準備）

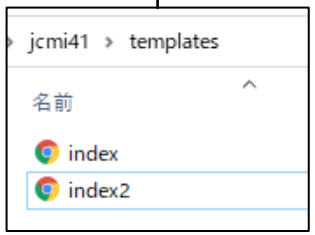
① 参加申し込みページから、今回使用するプログラムをダウンロードして下さい。
※6/28(水)までにアップロードします。

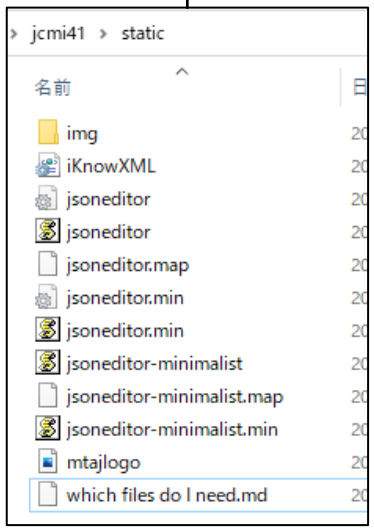
https://mta.gr.jp/tutorial/21jcmi/tutorial2021.html

② デスクトップに「41jcmi」等の名前の作業用フォルダを準備し、事前ダウンロードしたファイルを格納します。

作業フォルダの中にpythonの実行ファイルと「templates」「static」などのフォルダがあります。







ハンズオン準備（Flaskのインストールの確認）

- 作業フォルダ内にある「hello.py」をCMDで実行します。

hello.py - TeraPad

ファイル(F) 編集(E) 検索(S) 表示(V) ウィンドウ(W)

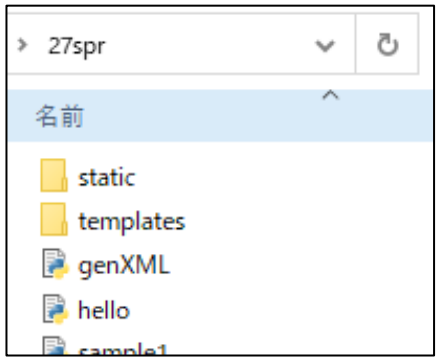
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    name = "Hello World"
    return name
@app.route('/jcmi41')
def good():
    name = "今日は学会初日です！"
    return name
if __name__ == "__main__":
    app.run(debug=True)
[EOF]
```

A

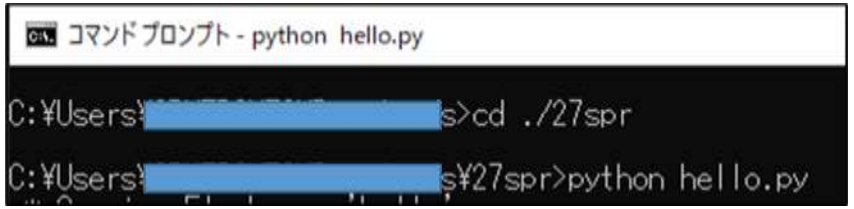
B

hello.pyのコード

① 作業フォルダにあるhello.pyを使用します。

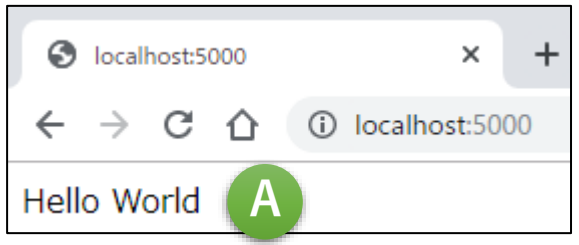


② コマンドプロンプト/ターミナルで作業フォルダに移動し、「python hello.py」を実行



③ Webブラウザを開くとPythonの出力をブラウザ上に表示できます。

http://localhost:5000/

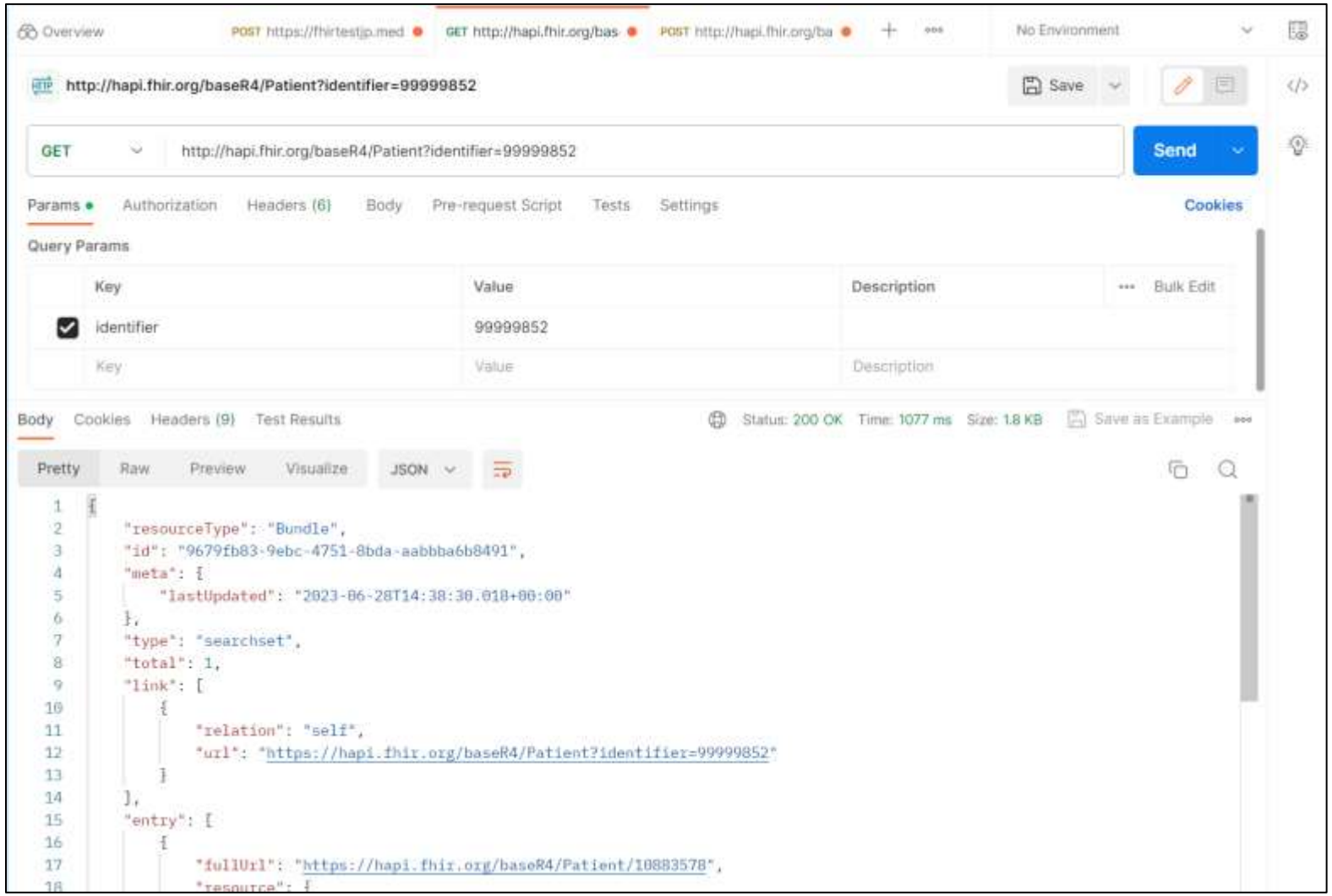


http://localhost:5000/sympo2022

④ Ctrl ＋ C で終了します

ハンズオン準備（Postmanの紹介）

・RESTのhttp要求・レスポンスのデータの流れを確認できるツール



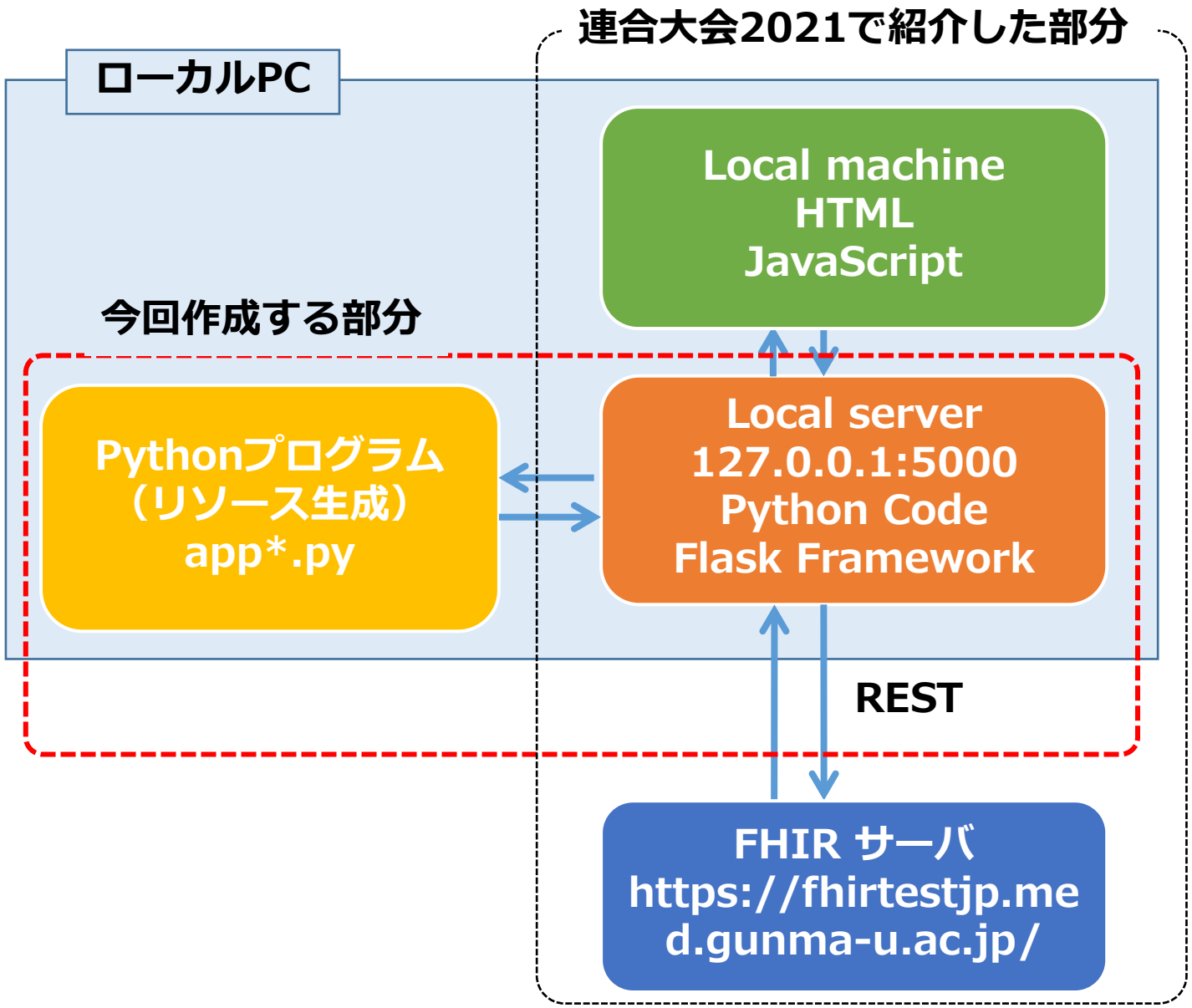
本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - **ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！**
  - ハンズオン2：患者情報（*Patient*）を操作する
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - ハンズオン4：文書情報（FHIR Document）の操作

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

完成イメージ



Python/Flaskが中心となり、

1）FHIRサーバとのREST通信
2）jsonデータの送受信
3）データの取得/登録/削除

各FHIRリソース・DocumentのREST通信における取り扱い方法を理解し、サーバへの送受信プログラムを作成します。
（表示系は一部JavaScriptを利用）

PythonにおけるREST呼び出しの最も単純な形

サンプルコード（app1.pyの一部）

```
#!/usr/bin/env python3
↓
# ライブラリのインポート
import urllib.request
import ssl
import json
import pprint
↓
ssl._create_default_https_context = ssl._create_unverified_context
↓
# RESTで取得するFHIRデータのURI
resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient/1'
↓
# 取得したデータを加工し表示
with urllib.request.urlopen(resturl) as response:
    fhir = response.read()
    json1 = json.loads(fhir.decode('utf-8')) #注意: json.loadはファイルから読む場合に使用する
↓
    #要素全てを取得
    print(json1['identifier'])
↓
    #リストの最初を読み出し
    print(json1['identifier'][0]['value'])
↓
    #リストの大きさを理解して読み出し
    print(json1['identifier'][len(json1['identifier'])-1]['value'])
↓
    #jsonデータ全てを表示
    pprint.pprint(json1)
```

RESTのrequest処理

JSON処理に必須

FHIRリソースのURL

JSONをdictionary型にして読み込む

特定の要素のみを出力

JSONデータをそのものをコンソール出力

ハンズオン1　PythonでRESTによりFHIRの患者データを取得

RESTで取得したデータを「コンソールに出力する」「ファイルに出力する」方法

```
#!/usr/bin/env python3
↓
# ライブラリのインポート
import urllib.request
import ssl
import json
import pprint
↓
ssl._create_default_https_context = ssl._create_unverified_context
↓
# RESTで取得するFHIRデータのURI
resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient/1'
↓
# 取得したデータを加工し表示
with urllib.request.urlopen(resturl) as response:
    fhir = response.read()
    json1 = json.loads(fhir.decode('utf-8')) #注意: json.loadはファイルから読む場合に使用する
↓
    #要素全てを取得
    print(json1['identifier'])
↓
    #リストの最初を読み出し
    print(json1['identifier'][0]['value'])
↓
    #リストの大きさを理解して読み出し
    print(json1['identifier'][len(json1['identifier'])-1]['value'])
↓
    #jsonデータ全てを表示
    pprint.pprint(json1)
    ↓
    #jsonデータをテキストファイルに出力する
    path_w = './applout.txt'  # ※ここのパスは好きな場所に書き換え可
    with open(path_w, mode='w') as f:
        json2 = json.dumps(json1, indent=2, ensure_ascii=False)
        f.write(json2) # 問題なく出力されることを確認します。
```

エディタにapp1.pyを表示

ファイル出力部

①CMDで「python app1.py」を実行

```
¥pythonfiles¥jcmi41>
¥pythonfiles¥jcmi41>app1.py
```

No module named ***
のようなエラーが出た場合は、モジュールのインストールが済んでいないので、「pip3 install ***」のように入力してインストールします。

```
Traceback (most recent call last):
  File "app.py", line 5, in <module>
    import matplotlib.pyplot as plt
ModuleNotFoundError: No module named 'matplotlib'
```

ハンズオン1　RESTで取得したFHIRデータの確認

RESTで取得したデータを「ブラウザに表示する」「ファイルに出力する」方法

② コンソールに取得したFHIRデータが表示される

```
[['value': '99999999']]
99999999
99999999
{'address': [['postalCode': '3718511', 'text': '群馬県前橋市昭和町3-39-15']],
 'birthDate': '1970-01-01',
 'gender': 'male',
 'id': '1',
 'identifier': [['value': '99999999']],
 'meta': {'lastUpdated': '2021-03-06T12:57:58Z', 'versionId': '1'},
 'name': [['extension': [['url': 'http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation',
                          'valueCode': 'IDE']],
           'family': '群馬',
           'given': ['太郎'],
           'text': '群馬 太郎',
           'use': 'official'],
          ['extension': [['url': 'http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation',
                          'valueCode': 'SYL']],
           'family': 'グンマ',
           'given': ['タロウ'],
           'text': 'グンマ タロウ',
           'use': 'official']],
 'resourceType': 'Patient',
 'telecom': [['value': '0272208773']]}
```

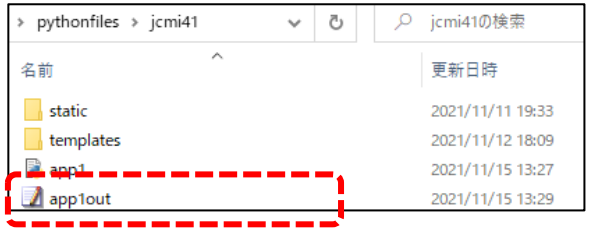
```
#要素全てを取得
print(json1['identifier'])
```
→ `[['value': '99999999']]`

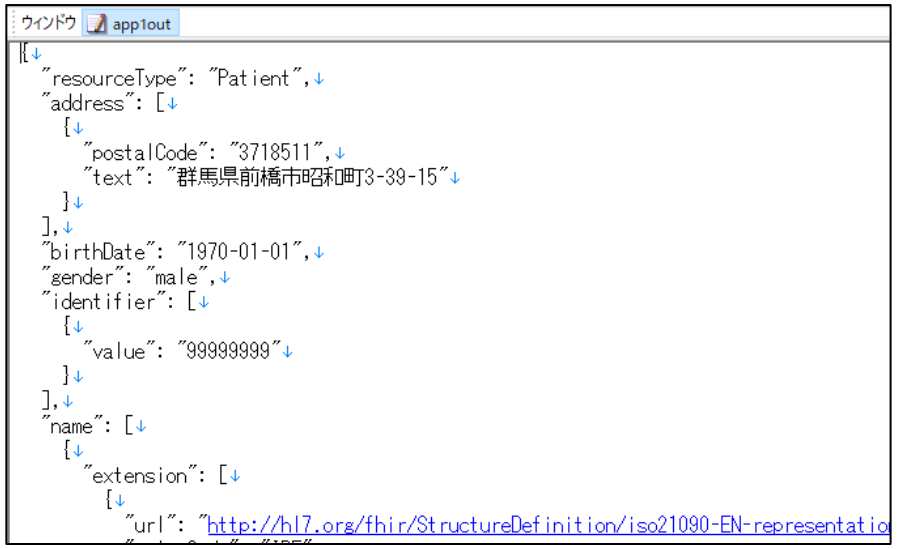
```
#リストの最初を読み出し
print(json1['identifier'][0]['value'])
```
→ `99999999`

```
#リストの大きさを理解して読み出し
print(json1['identifier'][len(json1['identifier'])-1]['value'])
```
→ `99999999`

```
#jsonデータ全てを表示
pprint.pprint(json1)
```
→ `{'address': [['postalCode':`
`'birthDate': '1970-01-01',`
`'gender': 'male',`

③作業フォルダにできた「app1out.txt」を開くと、②のjsonと同じ内容が表示されます。





ハンズオン1　FHIRデータの欲しい部分を取得するには

Patientリソースの構造（HL7 FHIR公式）

8.1.2 Resource Content

Structure | UML | XML | JSON | Turtle | R3 Diff | All

Structure

Name	Flags	Card.	Type	Description & Constraints
Patient	N		DomainResource	Information about an individual or animal rece Elements defined in Ancestors: id, meta, impl
identifier	Σ	0..*	Identifier	An identifier for this patient
active	?! Σ	0..1	boolean	Whether this patient's record is in active use
name	Σ	0..*	HumanName	A name associated with the patient
telecom	Σ	0..*	ContactPoint	A contact detail for the individual
gender	Σ	0..1	code	male | female | other | unknown AdministrativeGender (Required)
birthDate	Σ	0..1	date	The date of birth for the individual
deceased[x]	?! Σ	0..1		Indicates if the individual is deceased or not
deceasedBoolean			boolean	
deceasedDateTime			dateTime	
address	Σ	0..*	Address	An address for the individual
maritalStatus		0..1	CodeableConcept	Marital (civil) status of a patient MaritalStatus (Extensible)
multipleBirth[x]		0..1		Whether patient is part of a multiple birth
multipleBirthBoolean			boolean	
multipleBirthInteger			integer	
photo		0..*	Attachment	Image of the patient
contact	I	0..*	BackboneElement	A contact party (e.g. guardian, partner, friend

データ構造　多重度　データ型

FHIRリソースの設計を理解する

① データ構造（階層）の場所を確認する

リソースのプロファイルを確認し、Pythonの辞書型データのどこを指定するかを把握します。

② データの多重度(Cardinality)を確認する

(必須格納数)..(可能格納数)という表記。多重度によってプログラムを少し工夫する必要があります。

③ データ型(Type)を確認する

データ型によってはさらに深い階層にデータが格納されていることが（よく）あります。

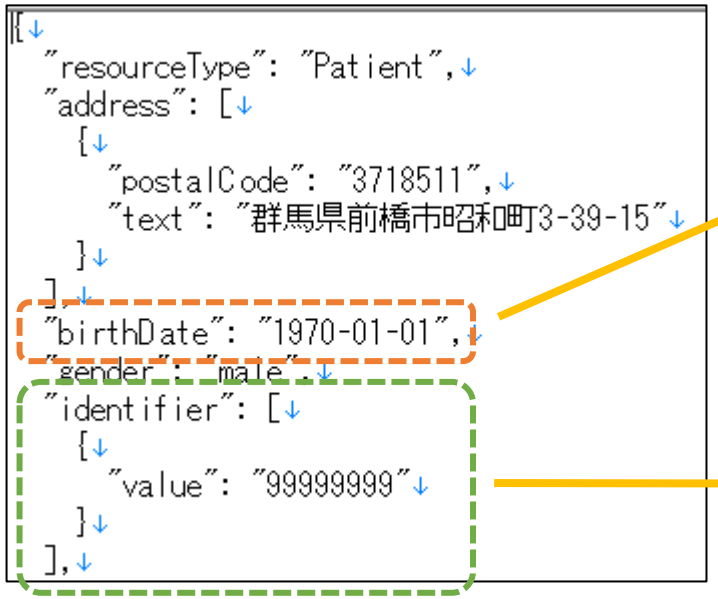
Structure

Name	Flags	Card.	Type
HumanName	Σ N		Element
use	?! Σ	0..1	code
text	Σ	0..1	string
family	Σ	0..1	string
given	Σ	0..*	string
prefix	Σ	0..*	string
suffix	Σ	0..*	string
period	Σ	0..1	Period

例えばHumanName型のデータは、左のような階層構造を持っており、**Patient.name.text**のように表現されます。

ハンズオン1　RESTで取得したFHIRデータの確認

改めて出力されたjsonを確認すると



“birthdate”は第1階層なので、

json1[‘birthdate’]

で取り出すことができます。

“identifier”の”value”は第2階層なので、

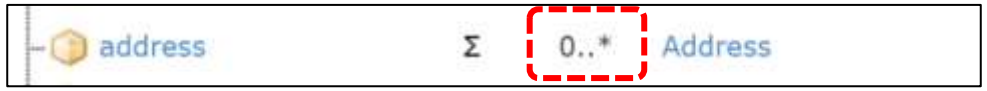
json1[‘identifier’][0][‘value’] となります。

この[0]は何でしょう？？

ハンズオン1　多重度が0...*または1...*の場合の対応方法

- Jsonではインスタンス数が2以上になる場合、[]←中カッコで囲われるので、python側でも読み取り方の工夫が必要になります。

【FHIRリソース】



Name	Flags	Card.	Type
Address	Σ N		Element
use	?! Σ	0..1	code
type	Σ	0..1	code
text	Σ	0..1	string
line	Σ	0..*	string
city	Σ	0..1	string
district	Σ	0..1	string
state	Σ	0..1	string
postalCode	Σ	0..1	string
country	Σ	0..1	string
period	Σ	0..1	Period

【jsonでの表現】

中カッコ[ ]が目印です

```
"address":[
  {
    "text": "神奈川県横浜市港区１－２－３",
    "postalCode": "123-4567",
    "country": "JP"
  }
]
```

【pythonのdictionary型での表現】

```
address_text = json1['address'][0]['text']
```

例え1つしか値が入っていない場合でも、pythonでは明示的に1つ目のデータを表す**[0]**が必要になります。
もしデータ数が不明の場合はインスタンス数のカウントや、存在しない場合のエラー処理が必要になります。(今回は割愛)

ハンズオン1　名前のデータを取得して出力してみましょう

app1.pyに**漢字氏名**を出力するプログラムを書いてみましょう。

```
#要素全てを取得
print(json1['identifier'])

#リストの最初を読み出し
print(json1['identifier'][0]['value'])

#リストの大きさを理解して読み出し
print(json1['identifier'][len(json1['identifier'])-1]['value'])

#【ハンズオン1】ここに取り出しすデータを記入してみましょう


#jsonデータ全てを表示
pprint.pprint(json1)
```

```
"name": [
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "IDE"
      }
    ],
    "use": "official",
    "text": "群馬 太郎",
    "family": "群馬",
    "given": [
      "太郎"
    ]
  },
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "SYL"
      }
    ],
    "use": "official",
    "text": "グンマ タロウ",
    "family": "グンマ",
    "given": [
      "タロウ"
    ]
  }
],
```

ヒント

すぐ上の**identifier**のコードをコピーすると便利です

print(json1　　)

この部分に指定するコードを書きます。

ハンズオン1　名前のデータを取得して出力してみましょう

（解答）app1.pyに漢字氏名を出力するプログラムを書いてみましょう。

```
#【ハンズオン1】ここに取り出しすデータを記入してみましょう
↓
print(json1['name'][0]['text'])
```

app1.pyを実行します

```
C:¥Users¥            ¥pythonfiles¥jcmi41>app1.py
[{'value': '99999999'}]
99999999
99999999
群馬 太郎
```

漢字氏名が表示されれば成功です

```
"name":[
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "IDE"
      }
    ],
    "use": "official",
    "text": "群馬 太郎",
    "family": "群馬",
    "given": [
      "太郎"
    ]
  },
  {
    "extension": [
      {
        "url": "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        "valueCode": "SYL"
      }
    ],
    "use": "official",
    "text": "グンマ タロウ",
    "family": "グンマ",
    "given": [
      "タロウ"
    ]
  }
],
```

この中カッコはtextにはかかっていないので注意

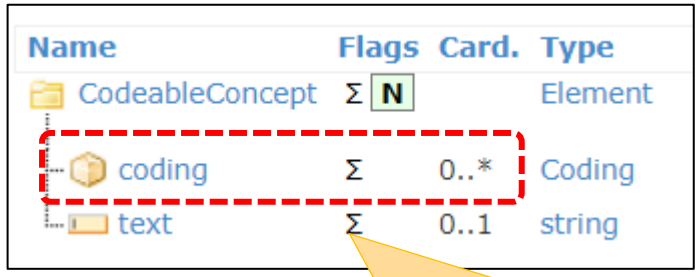
Tips　カナを取得したい場合は？？

JP Coreでは、漢字氏名の他にカナ氏名を入れる仕様になっています。漢字の方が先に格納されるので[0]を指定しましたが、実際のシステムでは”valueCode”のコードを一緒に読み込み、カナと漢字を判断します。

(参考)FHIRプロファイル（記述仕様）による仕様の違いに注意

- **電子処方箋FHIR記述仕様をもとにコーディングする場合に気をつけておきたいこと…電子処方箋独自の多重度に注意**

【FHIRの標準仕様】



CodeableConcept.codingの多重度は0...*

【電子処方箋仕様書案】

16	substitution				0..1	BackboneElement		後発医薬品への変更可否情報。詳細は「5.1 後発品変更可否」参照。
16.1		allowedCodeableConcept			1..1	CodeableConcept		
16.1.1			coding		1..1	Coding		後発品変更不可コード。
16.1.1.1				system	1..1	uri	"urn:oid:1.2.392.100495.20.2.41"	後発品変更不可コードを識別する URI。固定値。
16.1.1.2				code	1..1	code	“1”	後発品変更不可コード。値は例示。
16.1.1.3				display	0..1	string	“変更不可”	値は例示。

Substitution.allowedCodeableConcept.codingの多重度は1...1になっている

```
"substitution": [
  "allowedCodeableConcept": [
    "coding": [
      {
        "system": "urn:oid:1.2.392.100495.20.2.41",
        "code": "1",
        "display": "変更不可"
      }
    ]
```

ただしjsonは標準の0...*に合わせた[]中カッコが出現する仕様になっている

多重度でうまくいかない場合は、実際のjsonデータを見て確認するように

```
If jsonObj("entry")(i).Exists("substitution") = True Then
  Sheet2.Cells(row + 9 + j, 10) = jsonObj("entry")(i)("substitution")("allowedCodeableConcept")("coding")(1)("display")
  Sheet2.Cells(row + 9 + j, 11) = jsonObj("entry")(i)("substitution")("reason")("text")
End If
```

本日の流れ

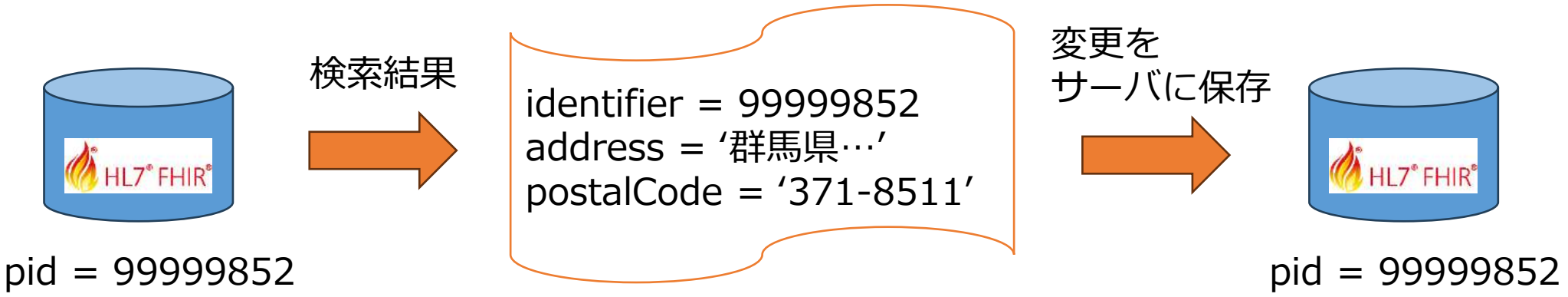
- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - **ハンズオン2：患者情報（Patient）を操作する**
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - ハンズオン4：文書情報（FHIR Document）の操作

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。



Tips　実際の利用シーンを考えると

Patientリソース（患者基本情報）いきなり情報を書き換えるのではなく、
既存データの有無を確認した上で
　データがない場合 → 新規作成
　データがある場合 → 更新
という手順で実行する操作をする。

ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。

app21.py

```python
@app.route("/", methods=['GET', 'POST'])
def index():
    #url = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient?params=/'
    url = 'http://hapi.fhir.org/baseR4/Patient?identifier='

    # 患者IDからPatientリソースを検索する
    if request.method == 'POST':
        # 入力されたPatient IDからURLを生成
        identifier_ = request.form['identifier_']
        url += identifier_

        # URLからJSONを取得
        with urllib.request.urlopen(url) as response:
            fhir = response.read()
            # jsonから辞書型に変換
            text = json.loads(fhir.decode('utf-8'))

        # 患者IDが見つからなかった場合は住所情報なし
        if text.get('total') > 0:
            note = '患者ID' + identifier_ + 'の住所情報が見つかりました'
            address = text['entry'][0]['resource']['address'][0]['text']
            postalCode = text['entry'][0]['resource']['address'][0]['postalCode']

        # 患者IDが見つかった場合は住所情報を取得する
        else:
            note = '患者ID' + identifier_ + 'の住所情報はありません'
            address = ''
            postalCode = ''

        # FHIR取得したリソースを「text」という変数に入れhtmlに送る
        return render_template("index_get.html", identifier_=identifier_, text=text, note=note, address=address, postalCode=postalCode)
```

入力された患者IDからURLを作成

サーバ側からPatientリソースを取得

住所と郵便番号のみ取得

ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。

index_get.html
index_post.html

```
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <h2>住所変更フォーム</h2>
        <a href="http://localhost:5000/"><input type="submit" value="クリア"></a>
        <h3>患者IDを入力して検索</h3>
        <form action="{{url_for('index')}}" method="POST" novalidate="novalidate">
            Patient ID:<input type="text" name="identifier_" placeholder="IDを指定する">
            <input type="submit" name="getid_" value="送信">
        </form>

        <p>{{note}}</p>
        <br>

        <hr>

        <h3>住所情報を編集して送信</h3>
        <form action="./post" method="POST" novalidate="novalidate">
            <table>
            <tr><td>Patient ID:</td><td><input type="text" size=50 name="identifier_" value={{identifier_}}></td></tr>
            <tr><td>郵便：</td><td><input type="text" size=50 name="postalCode" value={{postalCode}}></td></tr>
            <tr><td>住所：</td><td><input type="text" size=50 name="address" value={{address}}></td></tr>
            <tr><td><input type="submit" name="postid_" value="送信"></td></tr>
            </table>
        </form>

        <p>{{text}}</p>
    </body>
</html>
```

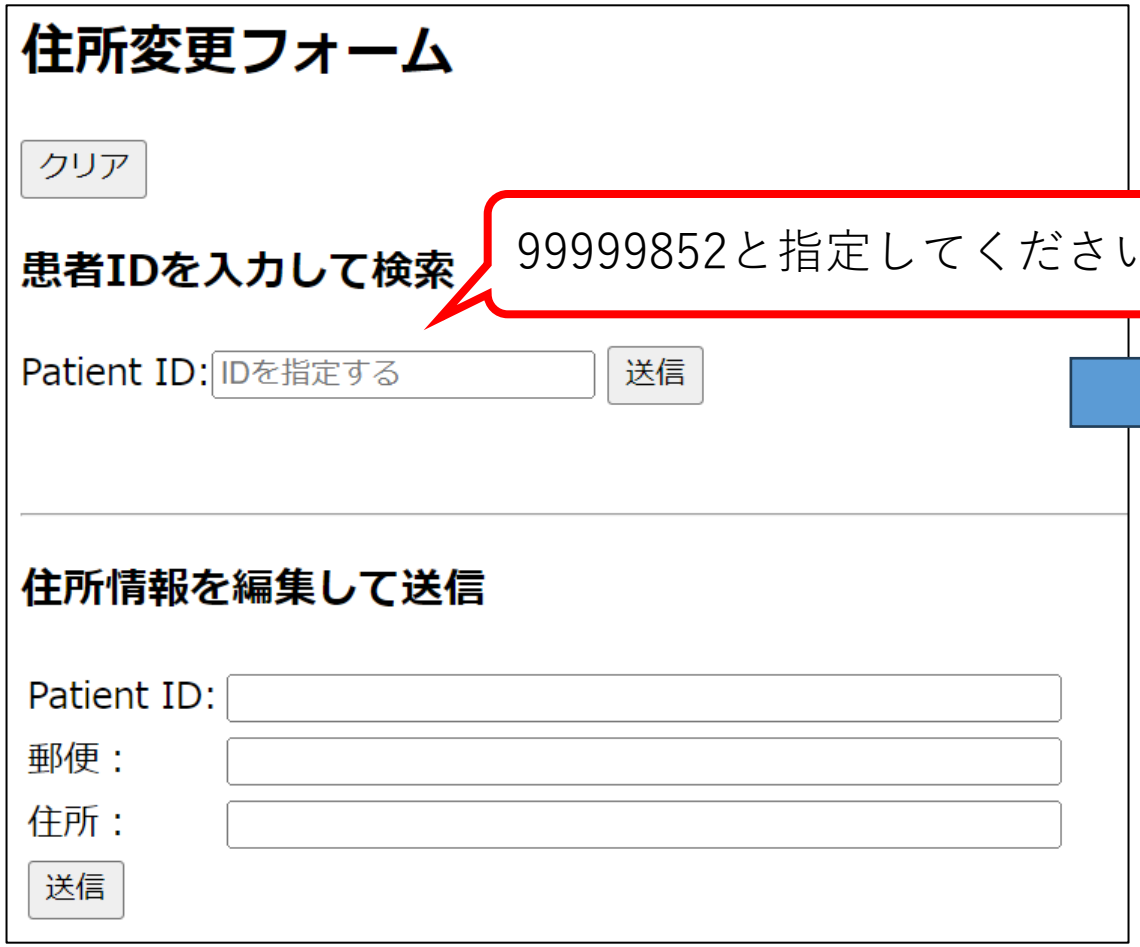
Python側で生成したURLを埋め込み

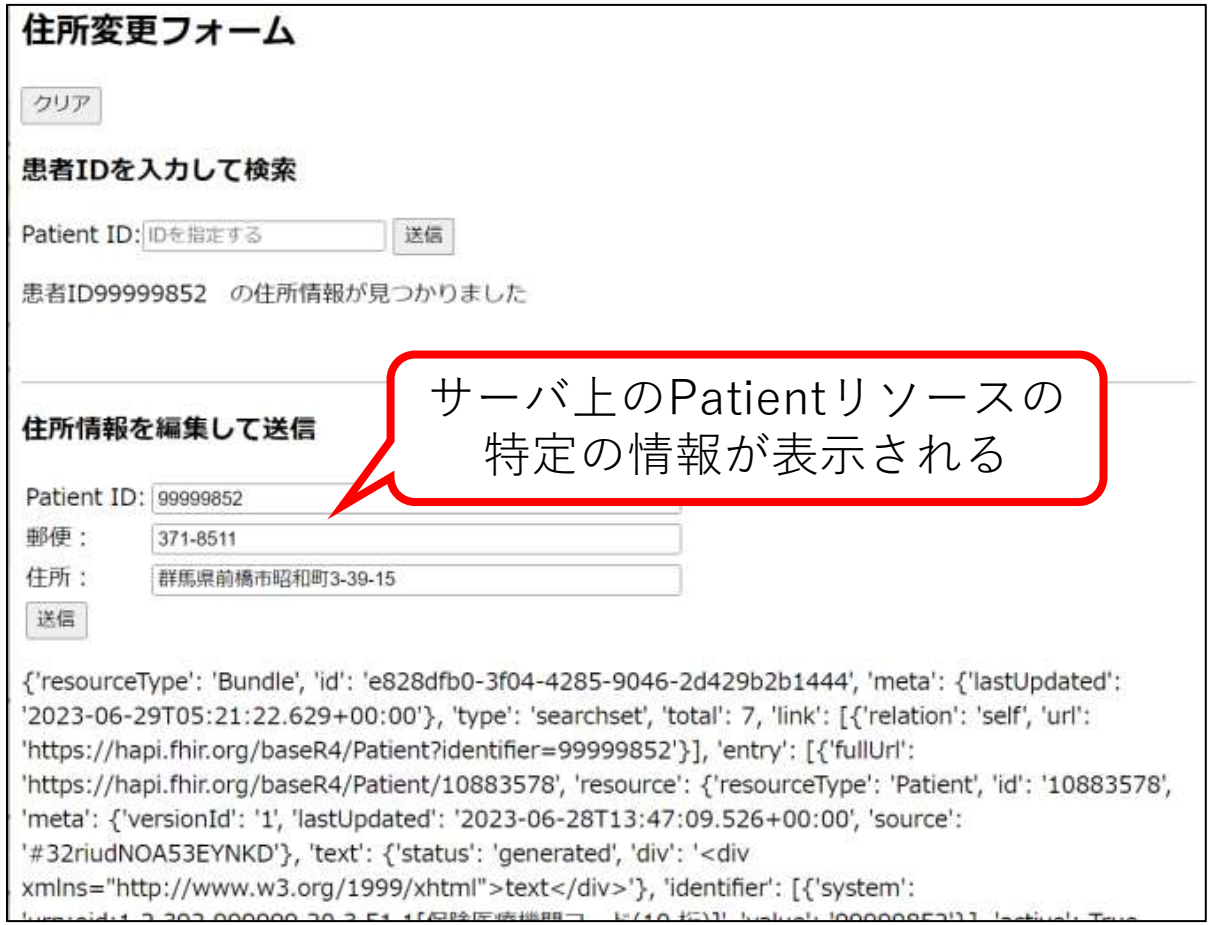
Getした住所と郵便番号をinputフォーム内に表示させ、編集できるようにする

ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。

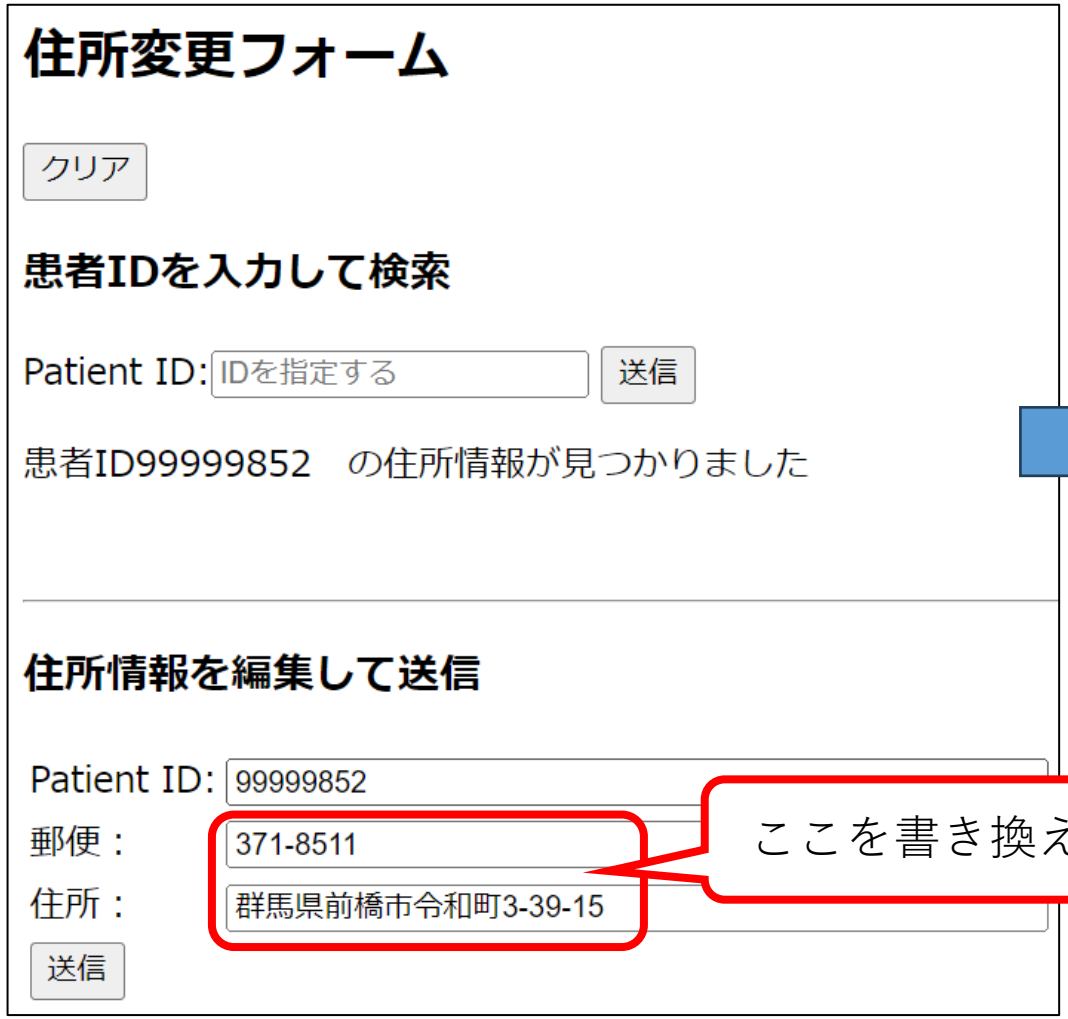
http://localhost:5000/　にアクセス

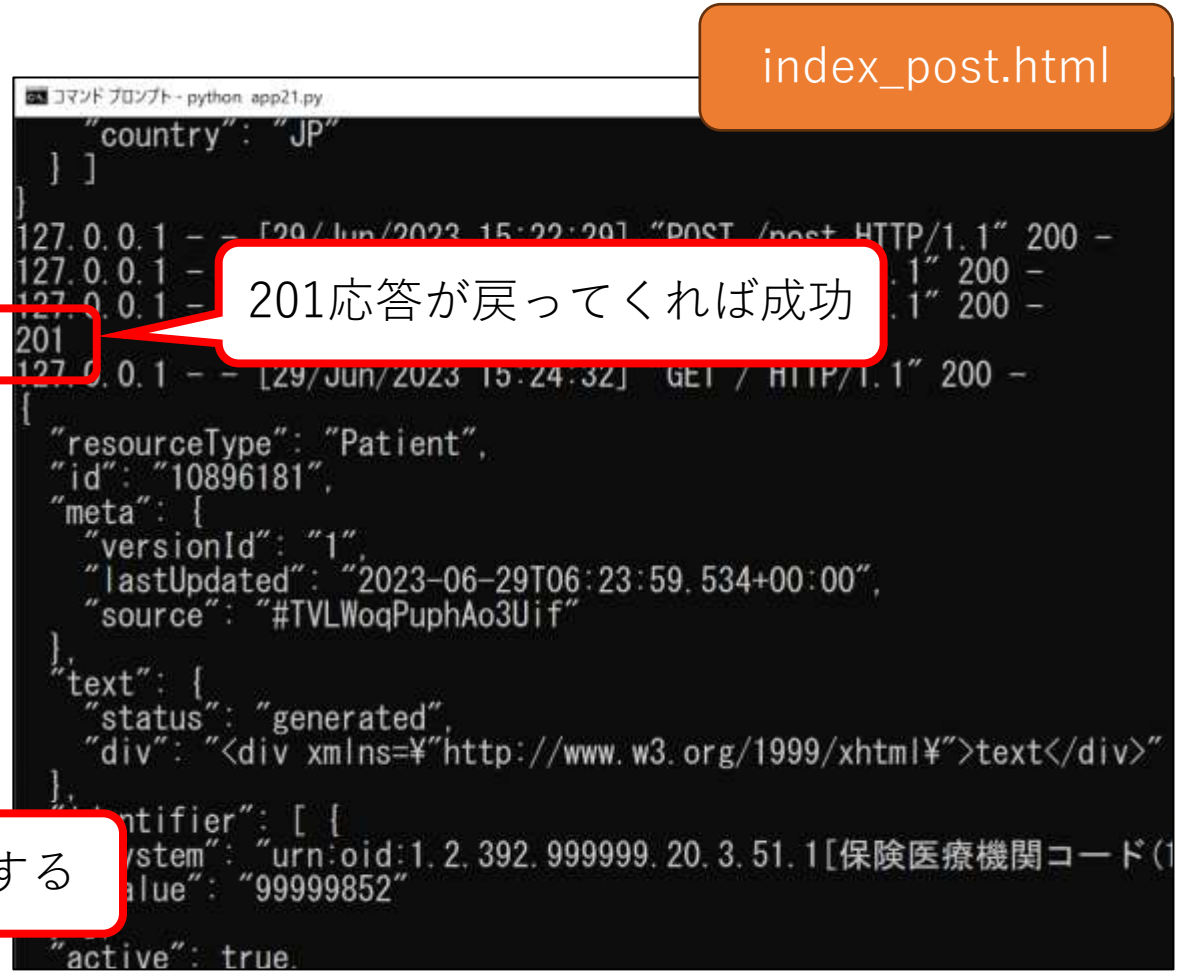




ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。





ここを書き換えて送信する

ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。

ハンズオン２　患者情報（Patient）を操作する

患者情報を検索し、情報を書き換えてサーバ側にPOSTする。

app21.py

```
# URLからJSONを取得
with urllib.request.urlopen(url) as response:
    fhir = response.read()
    # jsonから辞書型に変換
    text = json.loads(fhir.decode('utf-8'))

# 住所情報の部分をフォームの内容に書き換える
address = request.form['address']
postalCode = request.form['postalCode']
text['entry'][-1]['resource']['address'][0]['text'] = address
text['entry'][-1]['resource']['address'][0]['postalCode'] = postalCode
text = text['entry'][-1]['resource']
add_json = json.dumps(text)

# 書き換えられた情報をPOSTする
url = 'http://hapi.fhir.org/baseR4/Patient?identifier='
identifier_ = request.form['identifier_']
url += identifier_

headers = {"content-type": "application/json"}
response = requests.post(url=url, headers=headers, data=add_json)
response.encoding = response.apparent_encoding
print(response.status_code)
print(response.text)
```

formに入力された内容に確認

jsonの特定の要素をformで入力された内容に書き換える

POSTで送信

本日の流れ

- **HL7 FHIRについて（15分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (100分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - ハンズオン2：患者情報（Patient）を操作する
  - **ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する**
  - ハンズオン4：受診歴情報（Encounter）を操作する
  - ハンズオン5：文書情報（FHIR Document）を操作する

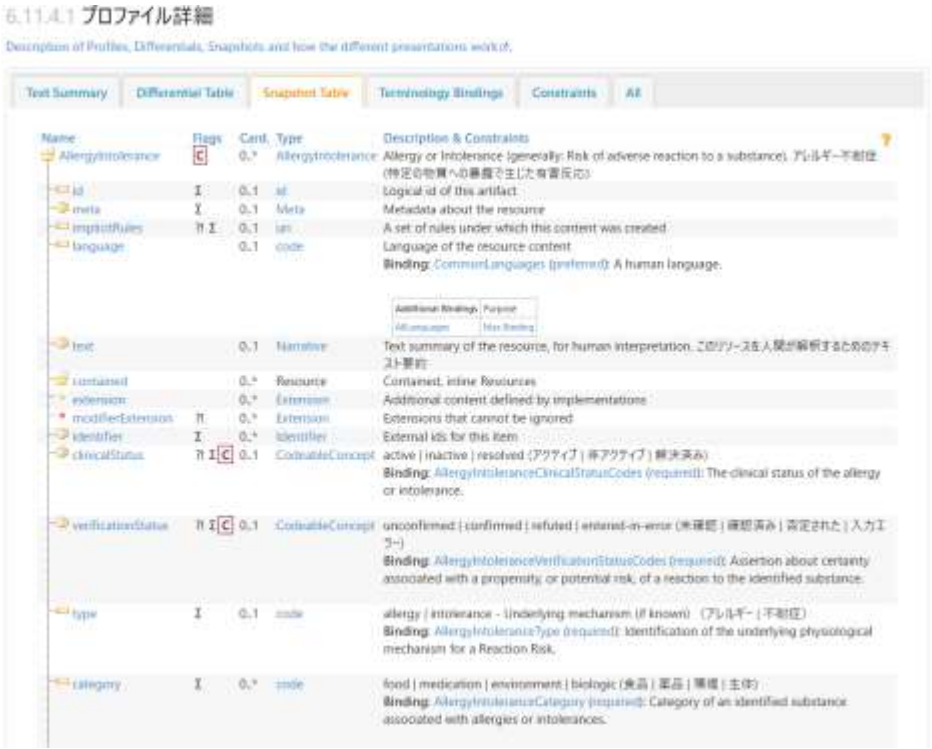
- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

ハンズオン３：アレルギー情報の取得と登録

HL7 FHIR のアレルギー情報は**AllergyIntolerance**というリソースで表現される。

JP coreのドキュメントは以下のURLを参照

https://jpfhir.jp/fhir/core/1.1.1/StructureDefinition-jp-allergyintolerance.html



AllergyIntoleranceリソースは原因となる物質の情報とそれによって引き起こされたアレルギー反応の情報の二つを格納する。

原因物質・・・分類(薬物、食物、環境、生体)
と原因物質
暴露時反応・・・重症度(軽度、中等度、重度)と症状

ハンズオン３：アレルギー情報の取得と登録

アレルギー原因物質に関する識別コードはJP core上次のValueSetとなっている。

- http://jpfhir.jp/fhir/core/CodeSystem/JP_JfagyFoodAllergen_CS
- http://jpfhir.jp/fhir/core/CodeSystem/JP_JfagyNonFoodNonMedicationAllergen_CS
- http://jpfhir.jp/fhir/core/CodeSystem/JP_JfagyMedicationAllergen_CS

コードのベースはJ-FAGY[1)]というアレルゲン用語集が使われている。このコードは階層構造となっており上位桁が一致すれば同じ分類の物質となる。
なお、本家FHIRのValueSetはSNOMED-CTが使われている。

コード	名称
J9FA15000000	小麦
J9FA15000016	小麦粉
J9FA15120000	国内産小麦
J9FA14000000	大麦

1) アレルギー情報の標準化を目指すJ-FAGY アレルゲン用語集とアレルゲンコードシステム. 河添 悦昌,永島 里美,大江 和彦. 第42回医療情報学連合大会論文集 2022.

ハンズオン３：アレルギー情報の取得と登録

アレルギー反応に関する症状コードはJP core上別途定義されておらず本家と同じくSNOMED-CTとなっている。

アレルギー反応を否定する表現に関しては個々のアレルギーではverificationStatusにrefutedを設定する。

本家ではアレルギー原因物質の識別コードに既知のアレルギーなしのコード(SNOMED-CT:716186003)を設定することが推奨されているがJPcoreでは該当するコードがないため、この方法は使えない。
なお、このコードを設定するときは患者全体のアレルギー情報と矛盾しないようにする必要がある。

ハンズオン３：アレルギー情報の取得と登録

3.1　事前準備

アレルギーを参照する患者リソースのidを取得します。ハンズオン２で登録した患者のPatientリソースを取得しidを控えておきます。Patientリソースの取得は**app31.py**を使用してください。

```
{'address': [{'postalCode': '1600023', 'text': '東京都新宿区'}],
 'birthDate': '1970-01-01',
 'extension': [{'url': 'http://hl7.org/fhir/StructureDefinition/patient-birthPlace',
                'valueAddress': {'state': '東京'}},
               {'url': 'http://jpfhir.jp/fhir/core/Extension/StructureDefinition/JP_Pa
                'valueCodeableConcept': {'coding': [{'code': '2039-6',
                                                     'display': 'Japanese',
                                                     'system': 'http://terminology.hl7
 'gender': 'male',
 'id': '10769660',
 'identifier': [{'system': 'urn:oid:1.2.392.100495.20.3.51.10000000000',
                 'value': '90000001'}],
 'meta': {'lastUpdated': '2023-06-26T10:37:23.833+00:00',
          'profile': ['http://jpfhir.jp/fhir/core/StructureDefinition/JP_Patient'],
```

```python
# 患者情報を取得するFHIRデータのURI
#resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Bundle?patientid=1'
#resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Patient/1'
resturl = 'http://hapi.fhir.org/baseR4/Patient/10769660'

headers = {"content-type": "application/json"}
response = requests.get(url=resturl, headers=headers)

jsonobj = response.json()

#jsonデータ全てを表示
pprint.pprint(jsonobj)

#jsonデータをテキストファイルに出力する
path_w = './Patient-1.json'
with open(path_w, mode='w',encoding="utf-8") as f:
    jsonstr = json.dumps(jsonobj, indent=2, ensure_ascii=False)
    f.write(jsonstr)
```

ハンズオン３：アレルギー情報の取得と登録

3.2　アレルギー情報取得

アレルギーの取得は**app32.py**を使用します。
app2.pyを開いてloadAllergyIntoleranceByPatientのパラメータを3.1で取得した患者idに変更して実行してください。

```
# -- メイン処理

print('-- アレルギー情報読み込み --')
allgrlst=loadAllergyIntoleranceByPatient('10769661')
pprint.pprint(allgrlst)

path_w='AllergyIntolerance-0.json'
with open(path_w, mode='w',encoding="utf-8") as f:
    jsonstr = json.dumps(allgrlst, indent=2, ensure_ascii=False)
    f.write(jsonstr)
```

ここを書き換えます

```
-- アレルギー情報読み込み --
ID:10769660のアレルギー情報なし
[]
```

先ほど新しく登録した患者なのでアレルギー情報は1件も存在しません。

ハンズオン３：アレルギー情報の取得と登録

3.3　アレルギー情報登録

アレルギーの登録は**app33.py**を使用します。
app3.pyを開いてcreateFoodAllergyIntoleranceとcreateMedicationAllergyIntoleranceの引数に3.1で取得した患者IDをセットします。

```
print('-- 食物アレルギー情報登録 --')
allgr=createFoodAllergyIntolerance('10769661')
pprint.pprint(allgr)

path_w='AllergyIntolerance-1.json'
with open(path_w, mode='w',encoding="utf-8") as f:
    jsonstr = json.dumps(allgr, indent=2, ensure_ascii=False)
    f.write(jsonstr)

saveAllergyIntolerance(allgr)

print('-- 薬剤アレルギー情報登録 --')
allgr=createMedicationAllergyIntolerance('10769661')
pprint.pprint(allgr)

path_w='AllergyIntolerance-2.json'
with open(path_w, mode='w',encoding="utf-8") as f:
    jsonstr = json.dumps(allgr, indent=2, ensure_ascii=False)
    f.write(jsonstr)

saveAllergyIntolerance(allgr)
```

ここを書き換えます

ここを書き換えます

ハンズオン３：アレルギー情報の取得と登録

3.3　アレルギー情報登録

食物アレルギーは既に実装済ですが薬剤アレルギーは未実装となっています。
食物アレルギー生成関数を参考に薬剤アレルギーの生成関数を実装してみましょう。

```python
# -- 薬剤アレルギー情報を生成する
def createMedicationAllergyIntolerance(patientId):
    allgr=dict()

    allgrcd='YCM1214401A1027'
    allgrname='キシロカイン注射液０．５％'
    reactcd='422400008'
    reactname='嘔吐'
    reactlevel='mild'
    critical='low'
    occrdatetime=datetime.datetime(2023,6,21,8,10,15)
    recdatetime=datetime.datetime(2023,6,21,10,30,45)
    recstaffid='1853'

    # ここに食物アレルギー情報のリソース作成を参考に薬剤アレルギー情報のリソース作成処理を追加しましょう
    # https://jpfhir.jp/fhir/core/1.1.1/StructureDefinition-jp-allergyintolerance.html

    return allgr
```

薬剤アレルギーの生成関数を実装したらapp3.pyを実行してみましょう。
正しく実装されていれば患者にアレルギー情報が登録されます。

ハンズオン３：アレルギー情報の取得と登録

3.4　アレルギー情報の確認

app32.pyで実装されているアレルギー情報の取得処理をもう一度実行します。
すると先ほど登録した2件のアレルギー情報が表示されるはずです。

```
-- アレルギー情報読み込み --
[{'category': ['food'],
  'clinicalStatus': {'coding': [{'code': 'active',
                                 'system': 'http://terminology.hl7.org/CodeSystem/allergyintolerance-clinical'}]},
  'code': {'coding': [{'code': 'J9FB13110000',
                       'display': '鶏卵',
                       'system': 'http://jpfhir.jp/fhir/core/CodeSystem/JP_JfagyFoodAllergen_CS'}],
           'text': '鶏卵'},
  'criticality': 'low',
  'id': '10769668',
  'meta': {'lastUpdated': '2023-06-26T10:57:35.781+00:00',
           'profile': ['http://jpfhir.jp/fhir/core/StructureDefinition/JP_AllergyIntolerance'],
           'source': '#c8enEiXV2O9AT1dB',
           'versionId': '1'},
  'onsetDateTime': '2023-06-20T08:10:15',
  'patient': {'reference': 'Patient/10769661'},
  'reaction': [{'manifestation': [{'coding': [{'code': '247472004',
```

ハンズオン３：アレルギー情報の取得と登録

3.4　アレルギー情報の確認

app34.pyではflaskを用いて簡単なアレルギー情報表示画面を実装しています。
実行してブラウザを開くと患者IDを入力する画面が表示されます。3.1で取得した患者IDを入力して「データ表示」ボタンをクリックすると先ほど登録したアレルギー情報が表示されます。

http://localhost:5000/

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - ハンズオン2：患者情報（Patient）を操作する
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - **ハンズオン4：文書情報（FHIR Document）の操作**

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

医療関連文書FHIR記述仕様が厚生労働省標準に

HELICS 一般社団法人 医療情報標準化推進協議会（HELICS協議会）
HEaLth Information and Communication Standards Organization

トップページ | 医療情報標準化指針一覧表 | 入会のご案内 | 標準規格・レポート等の申請 | お問い合わせ

Ⅰ．「医療情報標準化指針」一覧（採択されたもの）

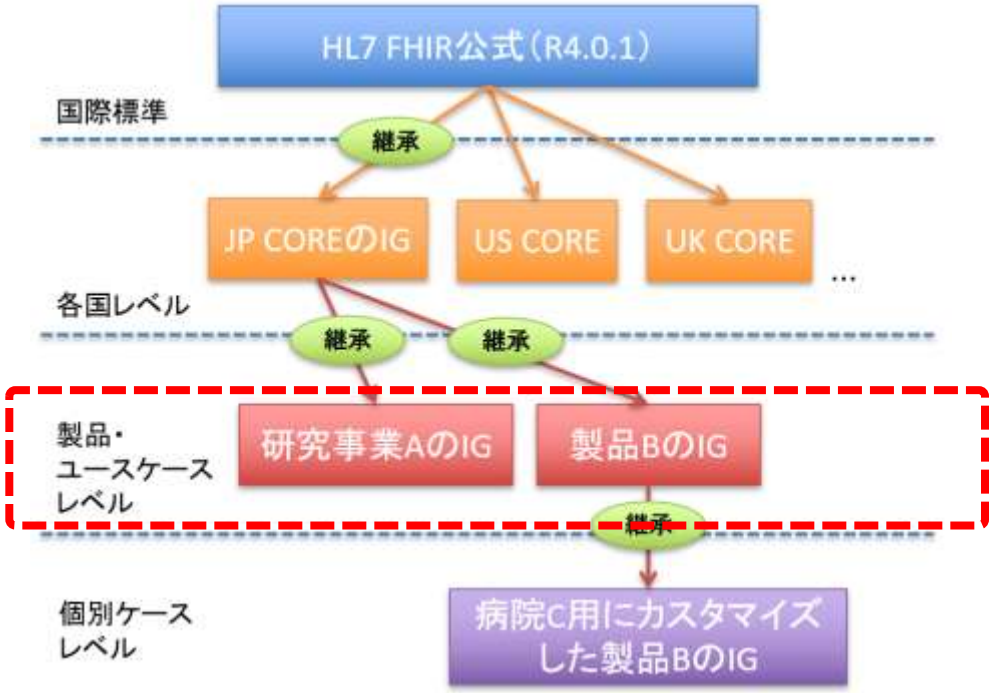
（申請受付番号は指針申請時に付番されます。改訂された場合には番号が変わりますので旧番号も合わせて記載します。改定履歴はⅣ．改定履歴一覧をご覧下さい。）

申請受付番号	提案規格名（［ ］内は提出団体名）	状況	申請日	採択日	厚生労働省標準規格	申請書	レポート	規格書等
HS036	処方情報HL7 FHIR記述仕様［日本医療情報学会］	採択	2021/09/21	2022/02/28	認定 2022/03/24 通知PDF	PDF	PDF	リンク
HS037	健康診断結果報告書HL7 FHIR記述仕様［日本医療情報学会］	採択	2021/09/21	2022/02/28	認定 2022/03/24 通知PDF	PDF	PDF	リンク
HS038	診療情報提供書HL7 FHIR記述仕様［日本HL7協会、日本医療情報学会］	採択	2021/10/10	2022/02/28	認定 2022/03/24 通知PDF	PDF	PDF	リンク
HS039	退院時サマリーHL7 FHIR記述仕様［日本HL7協会、日本医療情報学会］	採択	2021/10/10	2022/02/28	認定 2022/03/24 通知PDF	PDF	PDF	リンク

「3文書1情報」として2022年2月に承認された

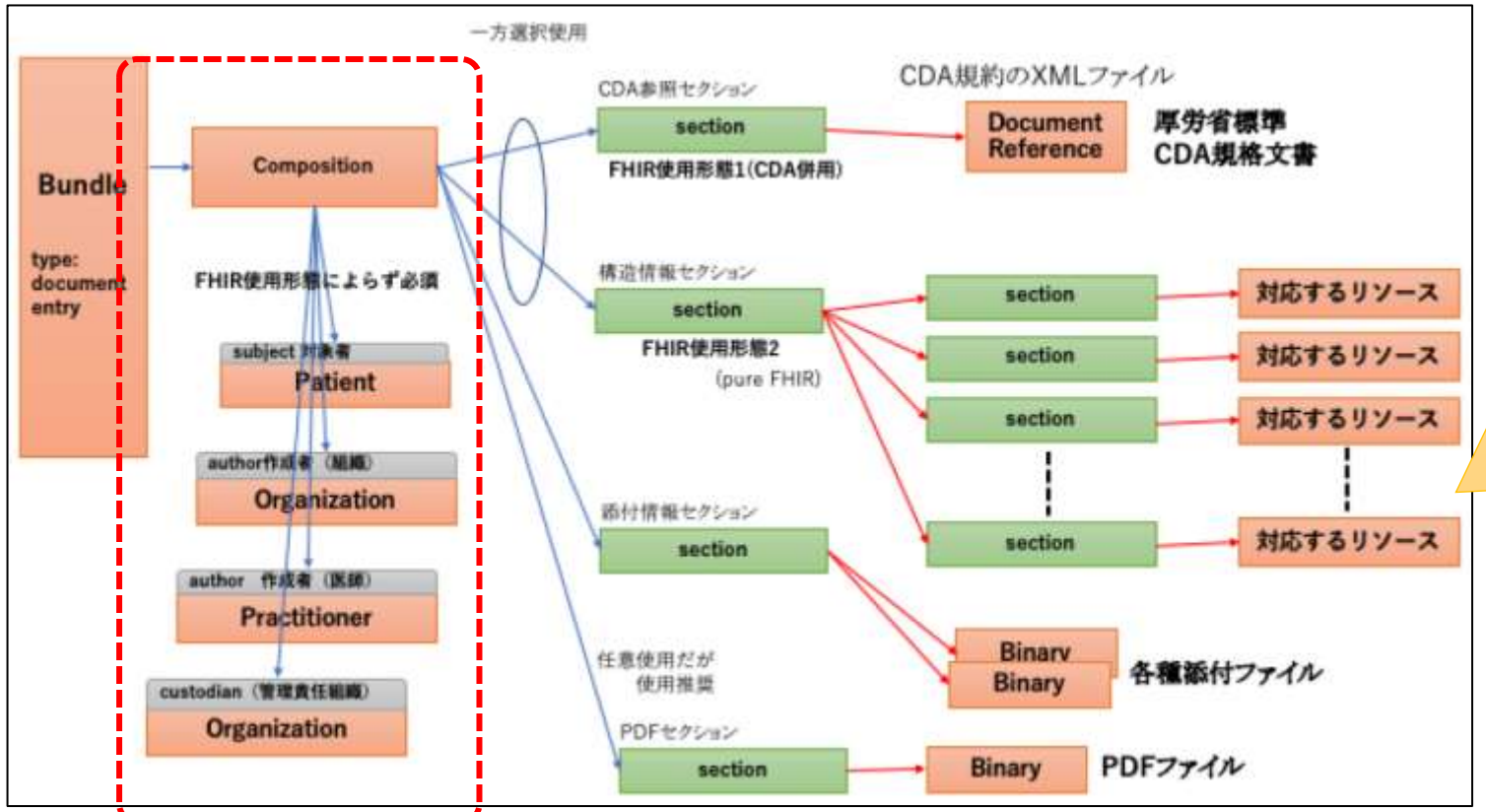
http://helics.umin.ac.jp/helicsStdList.html

- **令和２年度厚生労働科学特別研究事業の研究班により作成**
- **個別ユースケースごとにJP-COREを継承したプロファイルや実装ガイドが作成されることが見込まれる**



退院時サマリー・診療情報提供書記述仕様の全体構造

- 複数のリソースの組み合わせで構成されている
- 退院時サマリー・診療情報提供書の両者の基本的な構造は同じ
  - 文書全体はBundleリソースに包含されており、その下に全てのリソースがentryの形で登録される

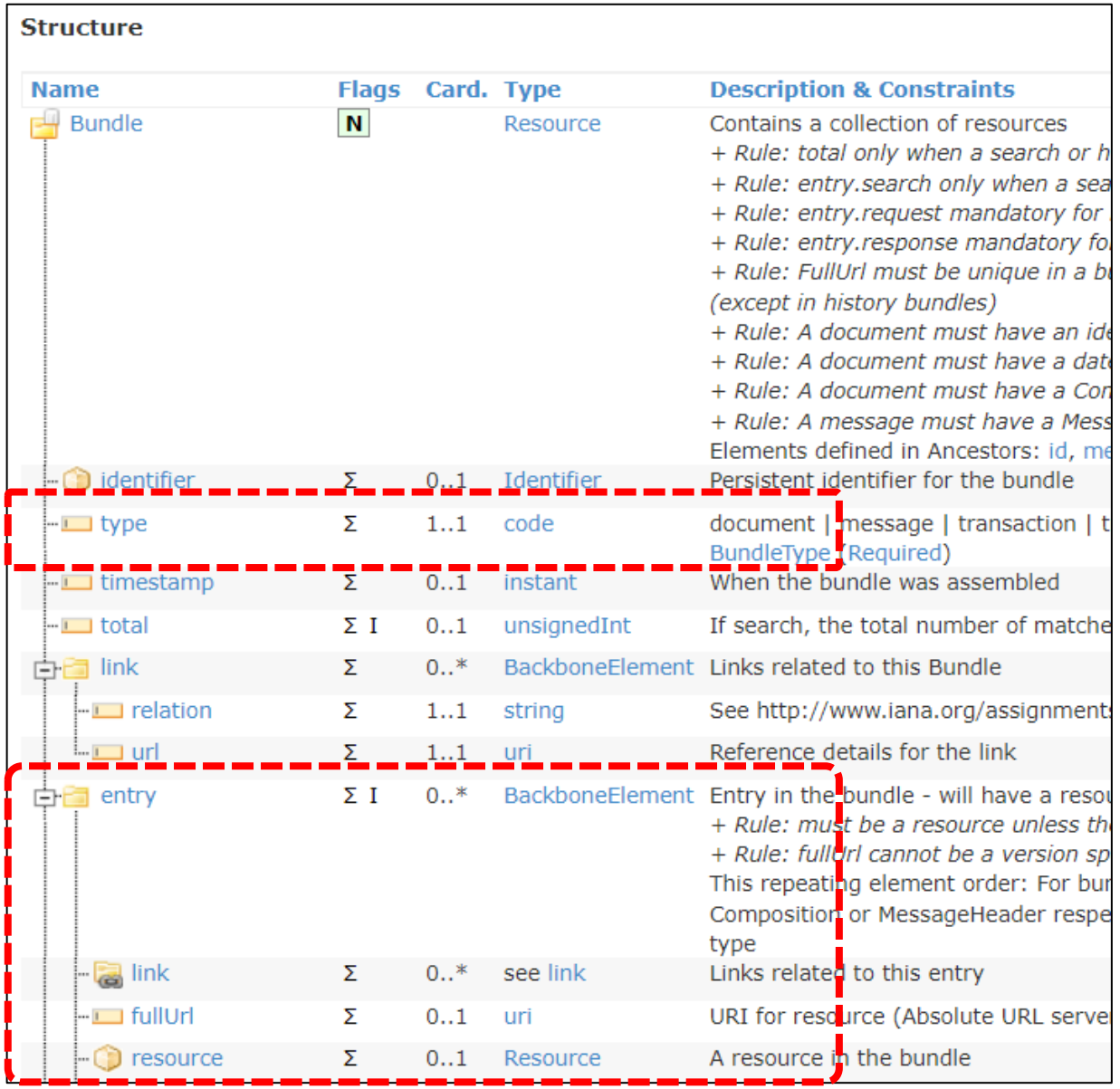


本チュートリアルでは、FHIR Documentで中心的な役割を持つCompositionを含む5つのリソースを含んだサンプルデータを使います。
「Patient」「Practitioner」「Organization」「Encounter」

退院時サマリー HL7 FHIR記述仕様（HS038）Ver. 1.0.1 p.6より引用

Bundleリソースとは（※とても重要※）

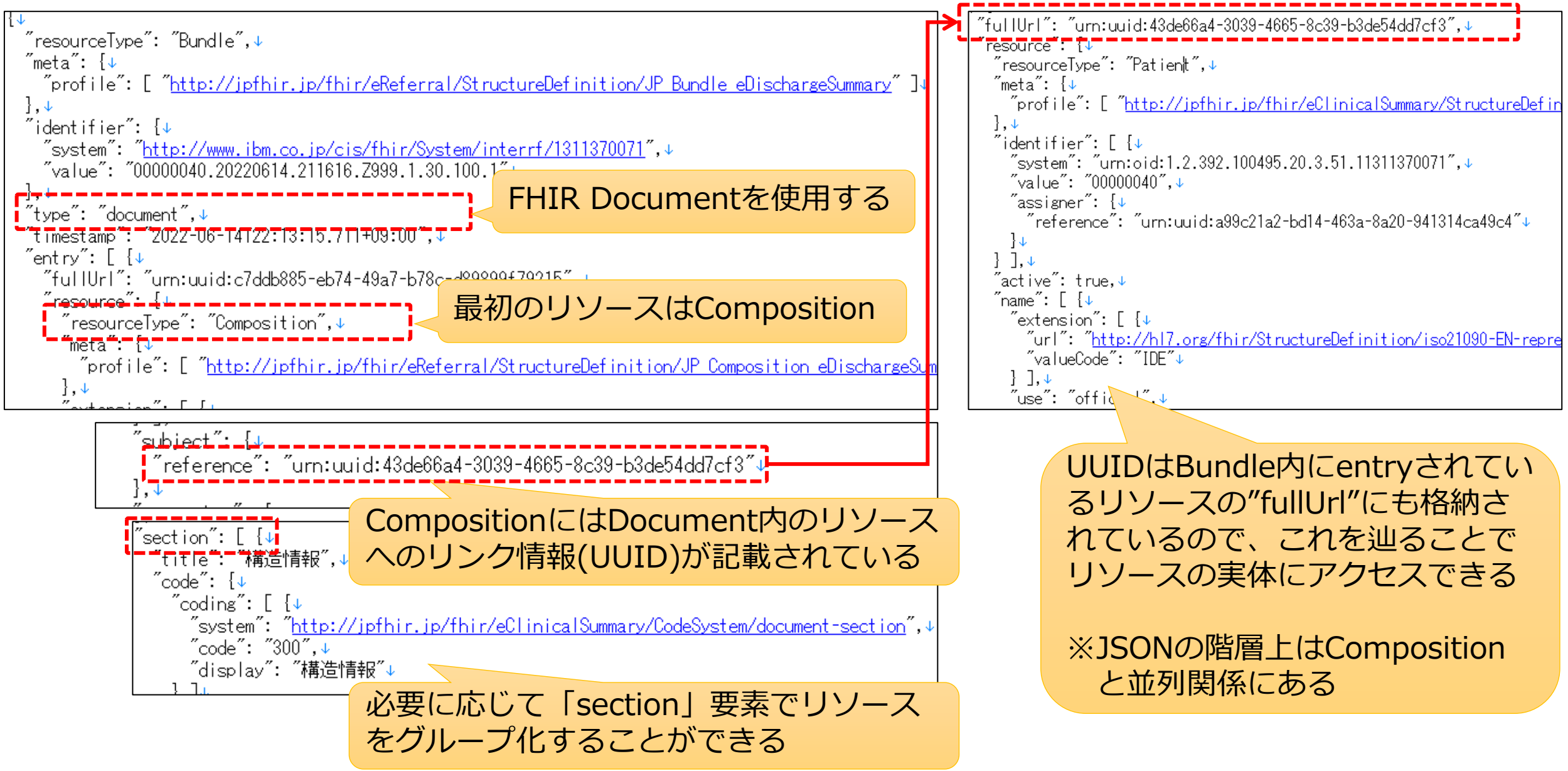
- 他のリソースを内包する「フォルダ」の役割をするリソース
  - Bundleの”type”=“document”にすることで、FHIR Documentという文書形式で記述できる
  - Compositionリソース･･･特定の文書情報を表すためのリソースの基本セットと、文書内のリソースの「目次」の役割を提供する
  - fullUrlエレメントにリソースのUUIDを格納することで、該当のリソースを参照することができる
- 厚労省標準となった3文書1情報は全てFHIR Documentを使用



Structure

Name	Flags	Card.	Type	Description & Constraints
Bundle	N		Resource	Contains a collection of resources + Rule: total only when a search or h + Rule: entry.search only when a sea + Rule: entry.request mandatory for + Rule: entry.response mandatory fo + Rule: FullUrl must be unique in a b (except in history bundles) + Rule: A document must have an ide + Rule: A document must have a dat + Rule: A document must have a Cor + Rule: A message must have a Mess Elements defined in Ancestors: id, me
identifier	Σ	0..1	Identifier	Persistent identifier for the bundle
type	Σ	1..1	code	document | message | transaction | t BundleType (Required)
timestamp	Σ	0..1	instant	When the bundle was assembled
total	Σ I	0..1	unsignedInt	If search, the total number of matche
link	Σ	0..*	BackboneElement	Links related to this Bundle
relation	Σ	1..1	string	See http://www.iana.org/assignment
url	Σ	1..1	uri	Reference details for the link
entry	Σ I	0..*	BackboneElement	Entry in the bundle - will have a resou + Rule: must be a resource unless th + Rule: fullUrl cannot be a version sp This repeating element order: For bur Composition or MessageHeader respe type
link	Σ	0..*	see link	Links related to this entry
fullUrl	Σ	0..1	uri	URI for resource (Absolute URL serve
resource	Σ	0..1	Resource	A resource in the bundle

Bundleリソースとは（※とても重要※）



診療情報提供書・退院時サマリー記述仕様の全体構造

- **どのリソースをentryするか、sectionに含めるかは、記述仕様書内で定義されている**

エントリ(entry)で表現する情報	使用される FHIR リソース	リソースの多重度
Bundle に含まれる全リソースエントリの参照リスト	Composition リソース	1..1
患者情報エントリ	Patient リソース	1..1
文書作成責任者情報エントリ	Practitioner リソース	1
文書法的責任者情報と同じ		
文書作成機		
文書管理責報と同じ場合		
文書作成責任者の所属診療科情報エントリ(文書作成機関情報と同じ場合省略)	Organization リソース	0..1
文書法的責任者の所属診療科情報エントリ(文書作成機関情報と同じ場合省略)	Organization リソース	0..1
入院詳細情報エントリ(退院時詳細情報、入院理由を含む)	Encounter リソース	1..1
入院期間中の診断情報エントリ(入院詳細情報エントリから参照される)	Condition リソース	1..*
入院前の所在施設(入院詳細情報エントリから参照される)	Observation リソース	0..*

リソースそのもの多重度も規定されている。
この例では、患者情報エントリである**Patientリソース**は必ず1つだけ作成することになる。

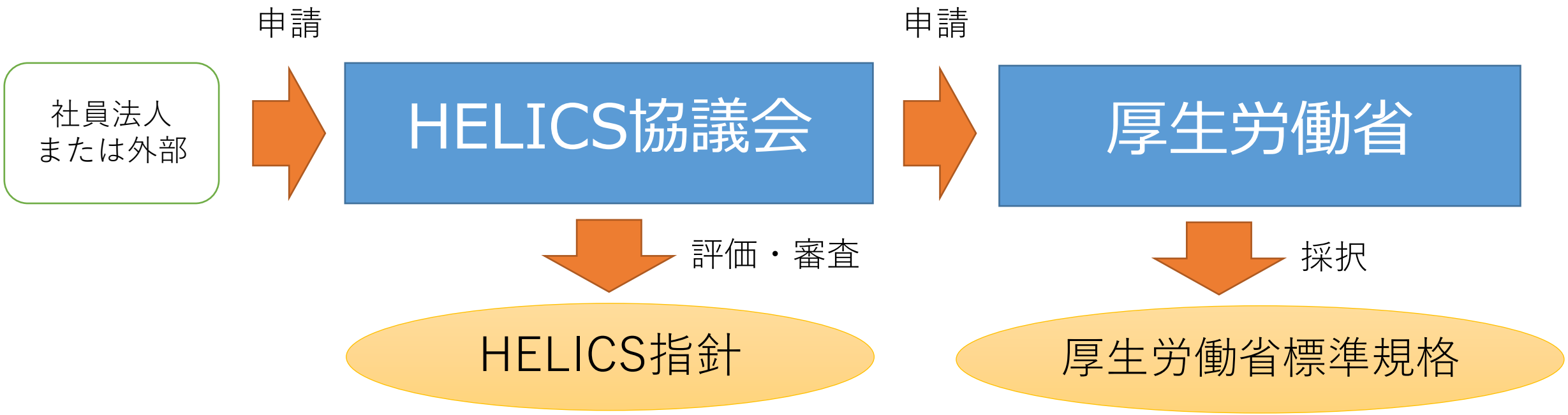
退院時サマリー HL7 FHIR記述仕様（HS038） Ver. 1.0.1 p.6より引用

サマリー本体(ボディー部)でのセクション構成(構造情報セクションの下のセクション)

セクションコード	セクション名(日本語) セクション名(英語)	必須/任意	対応する CDA 退院時サマリー規約の要素名	section.entry 参照先の FHIR リソース種別	section.entry の多重度
200	CDA 参照セクション cdaSection	任意	該当なし	DocumentReference (CDA 規約文書ファイルに限る)	1..1
300	構造情報セクション structuredSection	任意	該当なし	–	
322	入院詳細セクション detailsOnAdmissionSection	必須	入院時情報、入院時補足情報、入院時主訴・入院理由	Encounter 本表の他の Encounter と同一インスタンスを参照。	1..1
342	入院時診断セクション diagnosesOnAdmissionSection	必須	入院時情報	Condition	1..*
510	アレルギー・不耐性反応セクション allergiesIntoleranceSection	必須	アレルギー・不適応反応	AllergyIntolerance	0..*

我が国における標準規格の認定プロセス

- **一般社団法人医療情報標準化推進協議会（HELICS協議会）**
  - ➢**日本HL7協会、日本医療情報学会、医療情報システム開発センター等が社員となり、わが国の医療情報分野に適用し利用することが望ましい標準規格を評価・審査する機関**



Pythonを使用して退院時サマリーのFHIRデータをWebビューアに表示した後、PDFに印刷する

- **Pythonの環境を準備します。**

  ✓ Python 3.7以降(最新でOK)をインストールしてください。

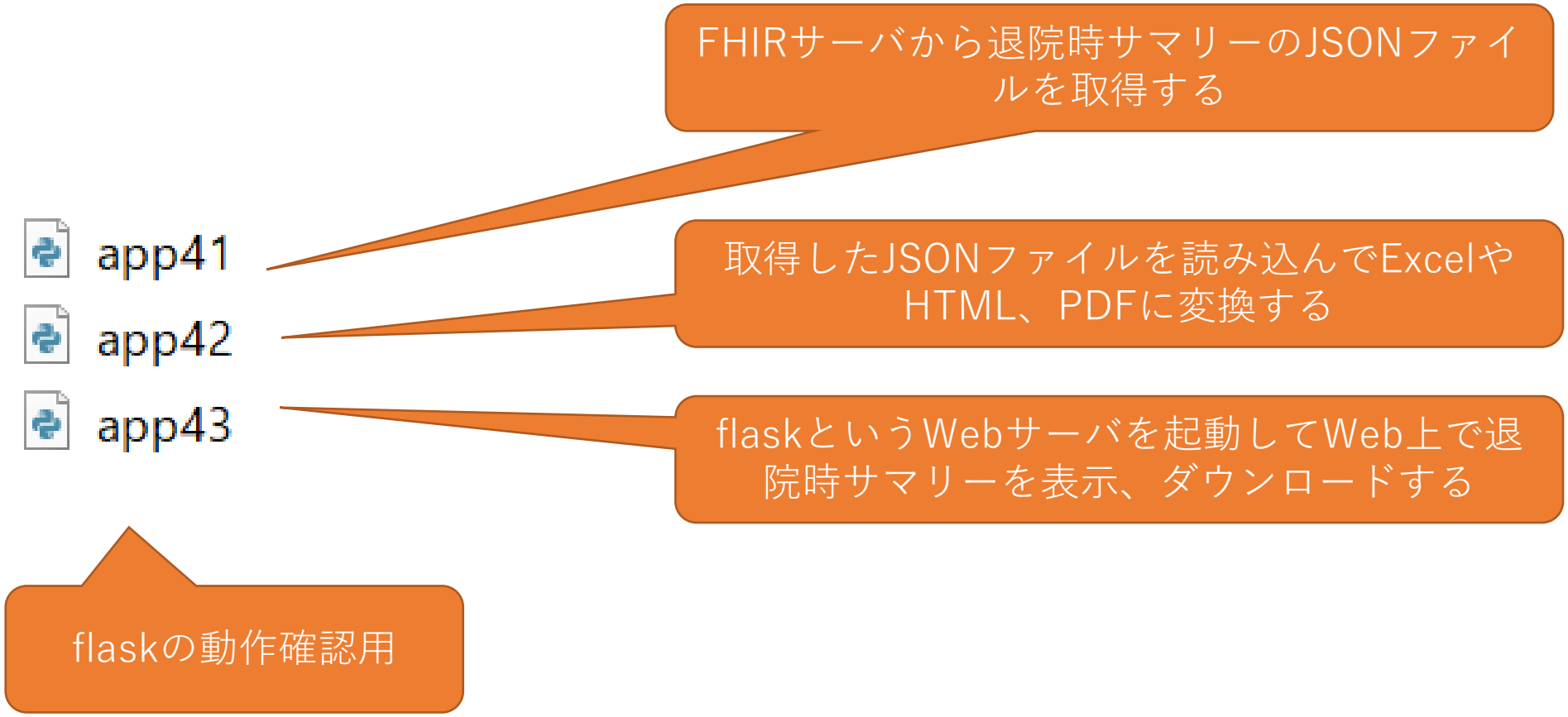
  python.jpの環境構築ガイドが参考になります。
  https://www.python.jp/install/install.html

  ✓ pip(Pythonのライブラリ管理ツール)で以下のライブラリをインストールしてください。

  flask
  pandas
  openpyxl
  xhtml2pdf

プログラムの全体構成

app41.py, app42.py, app43.pyを順に実行してください。



App41.pyFHIRサーバから退院時サマリーを取得

```
# RESTで取得するFHIRデータのURI
resturl = 'https://fhirtestjp.med.gunma-u.ac.jp/Bundle?patientid=1'

# 取得したデータを加工し表示
with urllib.request.urlopen(resturl) as response:
    fhir = response.read()
    jsonobj = json.loads(fhir.decode('utf-8')) #注意: json.loadはファイルから読む場合に使用する
```

FHIRサーバはRESTで構成されているため一般的なWebサービスの呼び出し処理と同様に実装できます。

取得した退院時サマリーデータはJSON形式のまま保存します。

app42.py退院時サマリーデータのExcelへの書き込み

ダウンロードしたJSONファイルを読み込みExcelやPDFに変換します。

```
jsondoc=None
json_path = './summary.json'
with open(json_path, mode='r') as json_file:
    jsonstr=json_file.read()
    jsondoc = json.loads(jsonstr)
```

JSONデータはpythonの辞書型に変換しておく

```
lst=[p for p in jsondoc['entry'] if p['resource']['resourceType']=='Composition']
subjectid=''
encounterid=''
autheridlst=[]
if len(lst)>0 :
    composi=lst[0]['resource']

    xlsx_ws['A1'] = '文書ID'
    print(composi['identifier']['value'])
    xlsx_ws['B1'] = composi['identifier']['value']
```

FHIR Bundleのentryリストから Compositionを検索して取得

FHIRデータはこのように辞書型のキーに名前をセットしてアクセスする

app42.py:退院時サマリーデータのExcelへの書き込み

```
lst=[p for p in jsondoc['entry'] if p.get('fullURL')==subjectid]
if len(lst)>0 :
    pat=lst[0]['resource']

    xlsx_ws['A8'] = '患者ID'
    print(pat['identifier'][0]['value'])
    xlsx_ws['B8'] = pat['identifier'][0]['value']

    xlsx_ws['A9'] = '患者氏名'
    print(pat['name'][0]['text'])
    xlsx_ws['B9'] = pat['name'][0]['text']

    xlsx_ws['A10'] = '患者カナ'
    print(pat['name'][1]['text'])
    xlsx_ws['B10'] = pat['name'][1]['text']
```

FHIR BundleのentryリストからFullURLで検索するとリソースを取得できる

リスト形式のデータは配列の添え字でアクセスする
配列なのでデータが本当にあるかどうか、順序よく格納されているかは保証しないのでチェックしておいたほうがよい

app42.py:退院時サマリーデータのHTMLへの変換

```
excel_df = pd.read_excel("summary.xlsx", engine='openpyxl',header=None)
```

engineにopenpyxlを指定するとpandasでExcelデータを読み込むことができる

```
htmllines.append('<table border="1">\n')
for i in range(len(excel_df)):
    htmllines.append("\t<tr>\n")
    for k in range(len(excel_df.columns)):
        val = excel_df.iloc[i, k]
        rowspan = 1
        if isinstance(val, str):
            val = re.sub("(\r\n)|(\n)", "<br />", val)
            l = 1
            while i+l < len(excel_df):
                _val = excel_df.iloc[i+l, k]
                if isinstance(_val, str):
                    break
                l += 1
            rowspan = l
            htmllines.append(f'\t\t<td colspan="1" rowspan="{rowspan}">{val}</td>\n')
    htmllines.append("\t</tr>\n")
htmllines.append('</table>\n')
```

Pandas data frameはインデックスを使用してデータを取得する

app42.py:退院時サマリーデータのPDFへの変換

```
pdf_path = 'summary.pdf'
with open(pdf_path, "w+b") as pdf_file:
        pisa_status = pisa.CreatePDF(src=htmlstr, dest=pdf_file)
        print(f'Pdf Convert Status:{pisa_status}')
```

HTMLデータをPDFファイルに変換する

app43.py:退院時サマリー表示Webサーバ

app3.pyはapp1,app2の内容を関数化してあります。これをWebページの処理に合わせて呼び出しています。

```
@app.route('/')
def index():…

@app.route('/SummaryViewer', methods=["POST"])
def viewSummary():…

@app.route('/SummaryExcel', methods=["GET"])
def downloadSummaryExcel():…

@app.route('/SummaryPDF', methods=["GET"])
def downloadSummaryPDF():…

if __name__ == "__main__":
    app.run(debug=True)
```

ホーム画面(サマリ検索画面)の表示

サマリデータの画面表示

サマリデータをExcelでダウンロード

サマリデータをPDFでダウンロード

app43.py:退院時サマリー表示

```python
@app.route('/SummaryViewer', methods=["POST"])
def viewSummary():
    '''
    指定したパラメータの条件でFHIRサーバの退院時サマリーを検索し画面に表示する
    '''
    qt=request.form["querytext"]

    jsondoc=downloadSummary(qt)
    excel_file=WriteSummaryExcel(jsondoc)

    sum_tbl_html=Markup(convertHtmlString(excel_file))
    return render_template('summary.html',query_text=qt,sum_tbl_html=sum_tbl_html)
```

POSTパラメータで渡された検索条件から退院時サマリーを取得
取得した退院時サマリデータをExcelに変換
Excelの内容をHTMLに変換してページ表示

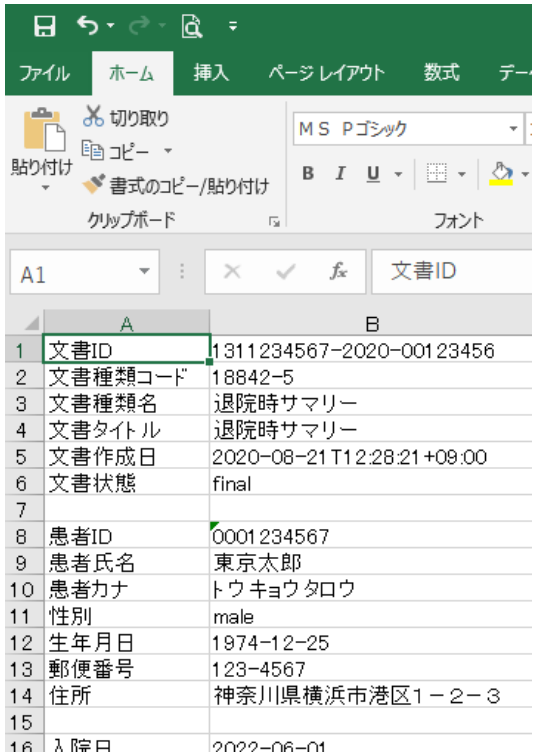
app43.py:退院時サマリーExcelダウンロード

```python
@app.route('/SummaryExcel', methods=["GET"])
def downloadSummaryExcel():
    '''
    指定したパラメータの条件でFHIRサーバの退院時サマリーを検索しExcelファイルにしてダウンロードする
    '''
    qt=request.query_string

    jsondoc=downloadSummary(qt)
    excel_file=WriteSummaryExcel(jsondoc)

    return send_file(excel_file, as_attachment=True,
                     attachment_filename=excel_file,
                     mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet')
```

GETパラメータで渡された検索条件から退院時サマリーを取得
取得した退院時サマリデータをExcelに変換
Excelデータをファイルダウンロードのレスポンスとして返す



	A	B
1	文書ID	1311234567-2020-00123456
2	文書種類コード	18842-5
3	文書種類名	退院時サマリー
4	文書タイトル	退院時サマリー
5	文書作成日	2020-08-21T12:28:21+09:00
6	文書状態	final
7		
8	患者ID	0001234567
9	患者氏名	東京太郎
10	患者カナ	トウキョウタロウ
11	性別	male
12	生年月日	1974-12-25
13	郵便番号	123-4567
14	住所	神奈川県横浜市港区１－２－３
15		
16	入院日	2022-06-01

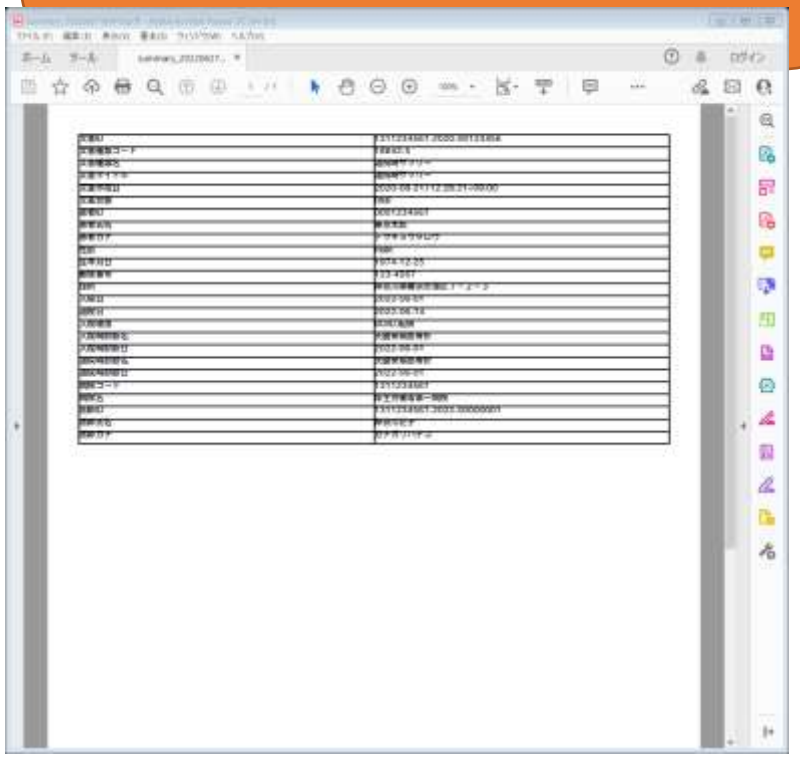
app43.py:退院時サマリーPDFダウンロード

```python
@app.route('/SummaryPDF', methods=["GET"])
def downloadSummaryPDF():
    '''
    指定したパラメータの条件でFHIRサーバの退院時サマリーを検索しPDFファイルにしてダウンロードする
    '''
    qt=request.query_string

    jsondoc=downloadSummary(qt)
    excel_file=WriteSummaryExcel(jsondoc)
    pdf_file=writeSummaryPDF(excel_file)

    return send_file(pdf_file, as_attachment=True,
                     attachment_filename=pdf_file,
                     mimetype='application/pdf')
```

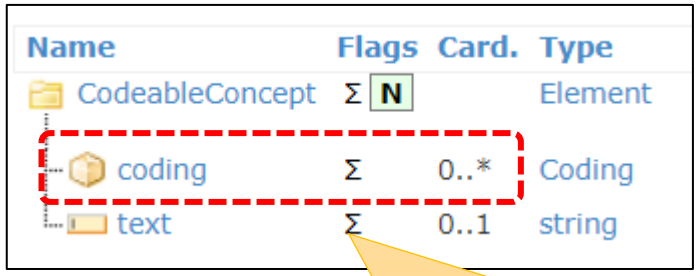
GETパラメータで渡された検索条件から退院時サマリーを取得
取得した退院時サマリデータをExcelに変換
ExcelデータをPDFファイルに変換
PDFデータをファイルダウンロードのレスポンスとして返す



【tips】ユースケースごとの仕様の違いに注意

- **FHIR記述仕様をもとにコーディングする場合に気をつけておきたいこと…ユースケース独自の多重度に注意**

【FHIRの標準仕様】



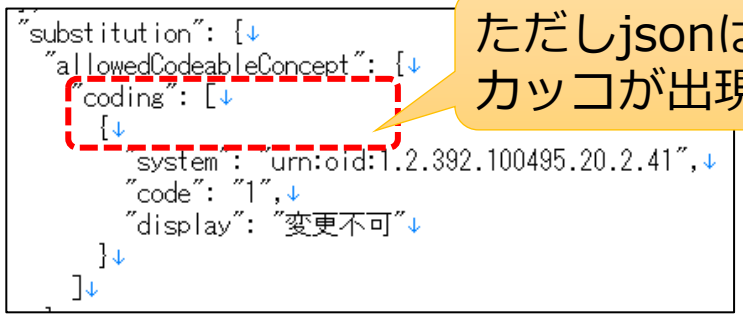
Name	Flags	Card.	Type
CodeableConcept	Σ N		Element
coding	Σ	0..*	Coding
text	Σ	0..1	string

CodeableConcept.codingの多重度は0…*

【例：退院時サマリーHL7 FHIR記述仕様書】

type				1..1	CodeableConcept		
	coding			1..1 *	Coding		
		system		1..1	uri		
		code		1..1	code	"18842-5" | "57133-1"	文書区分コード。 退院時サマリー:"18842-5"、診療情報提供書:"57133-1"を指定。 固定値。
		display		0..1	string	"退院時サマリー"	文書区分コードの表示名。

Composition.type.codingの多重度は**1…1**になっている（「1..1*」と記載することで、標準仕様と異なっていることに気づけるように配慮されている）



```
"substitution": {
  "allowedCodeableConcept": {
    "coding": [
      {
        "system": "urn:oid:1.2.392.100495.20.2.41",
        "code": "1",
        "display": "変更不可"
      }
    ]
```

ただしjsonは標準の0…*に合わせた[]中カッコが出現する仕様になっている

Point

多重度でうまくいかない場合は、**実際のjsonデータを見て確認**するように

本日の流れ

- **HL7 FHIRについて（20分）**
  - 医療情報ユーザーから見たFHIRアプリケーションの利点
  - FHIR紹介、REST, JSONについて

- **ハンズオン ～Pythonを利用しFHIR REST APIを操作する～ (90分)**
  - 環境準備・確認（Pythonバージョンの確認、各種ライブラリのインストール）
  - ハンズオン1：PythonでRESTによりFHIRデータを取得しよう！
  - ハンズオン2：患者情報（Patient）を操作する
  - ハンズオン3：アレルギー情報（AllergyIntolerance）を操作する
  - ハンズオン4：文書情報（FHIR Document）の操作

- **オブジェクト指向プログラミングの紹介、イベントのご案内（5分）**

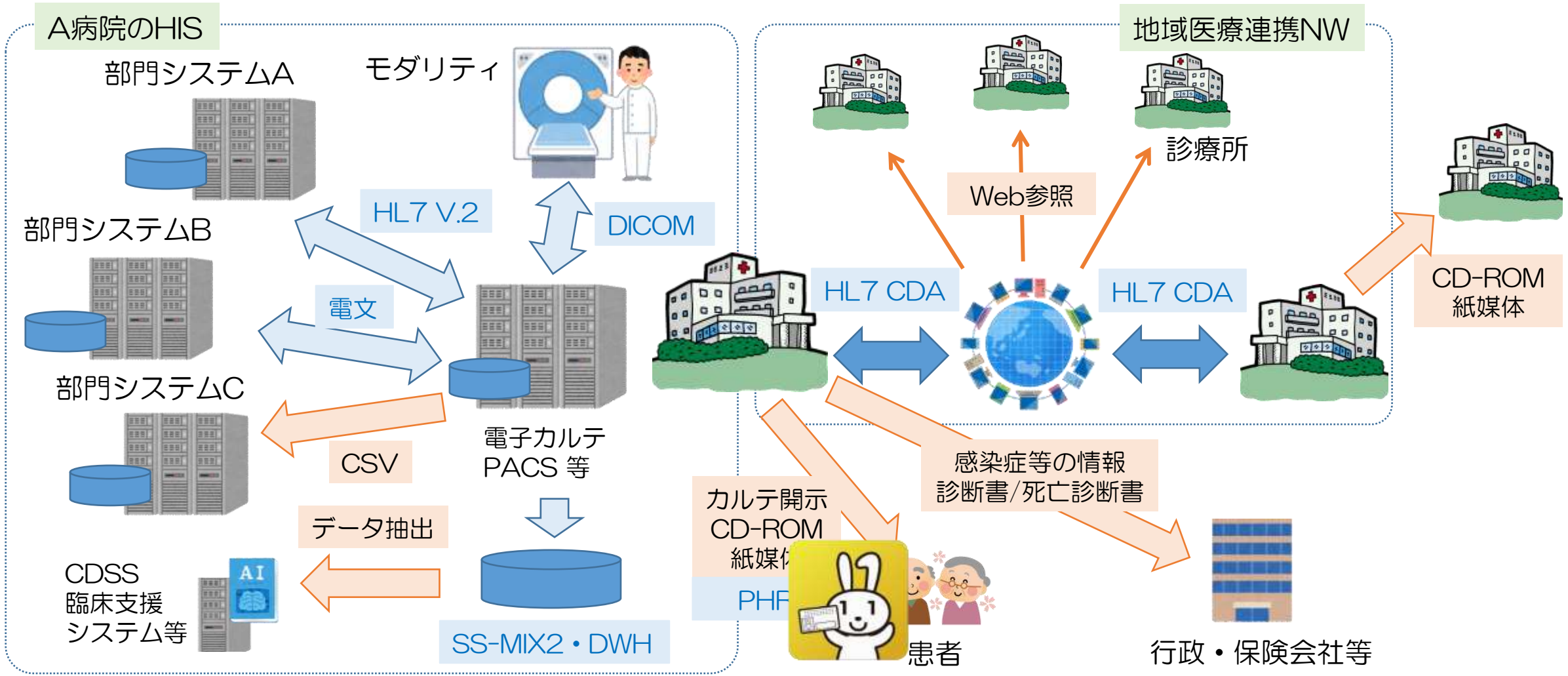
さらに使いこなしたい方へ

- 本チュートリアルでは、FHIRデータの取り扱いに関する概略と、Pythonを使用してリソースの取得・加工ができるよう、中心となるリソースを使用して基本的なコーディングを扱いました。

- 現時点ではJP Coreも一部のリソースの公開に留まっており、文書についても今後仕様変更や修正があるかもしれません。

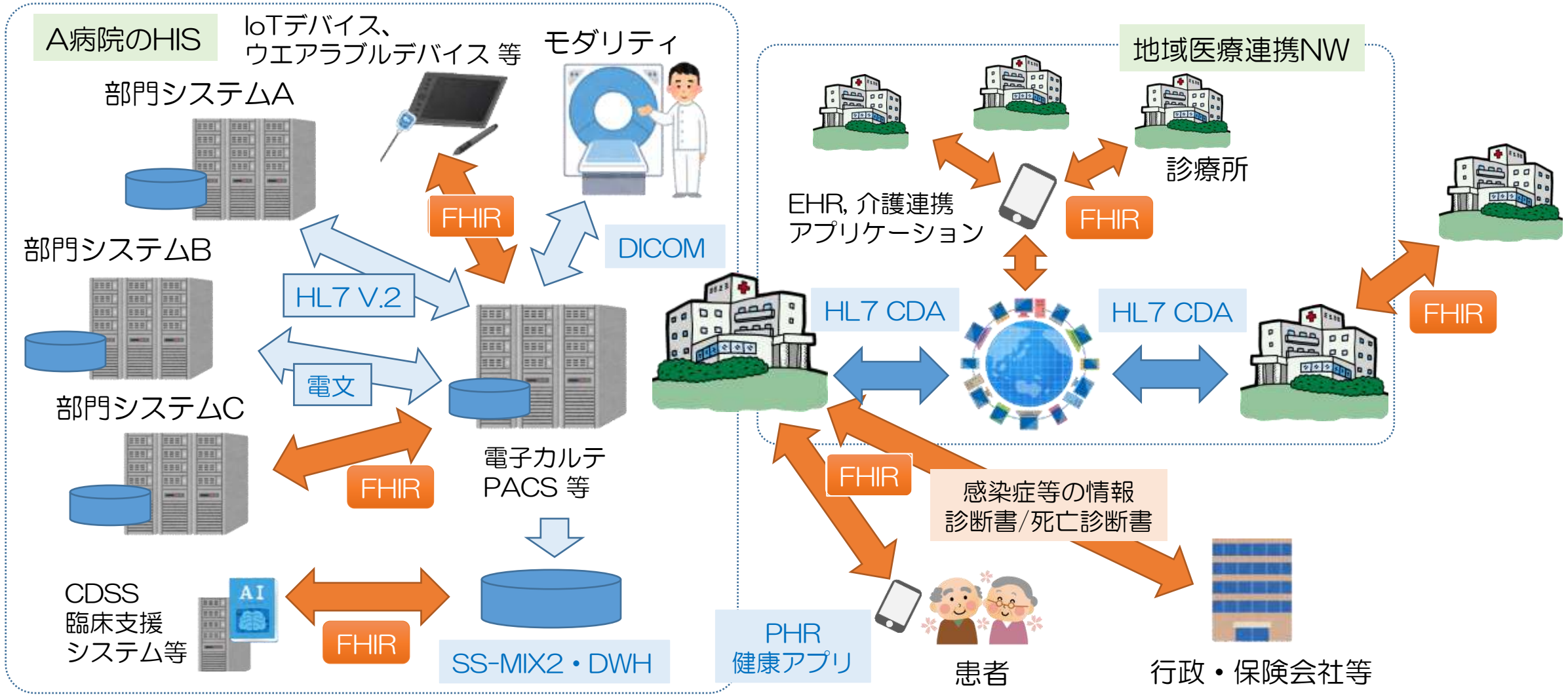
- 政府の政策はFHIRを普及させる枠組みが整備されつつあり、今後加速的な変化があるものと思います。FHIRの利用について、コーディングのレベルから一緒に考えてみましょう！

日本の電子カルテの構成とデータ交換



HL7 v2やCDA等を利用した双方向通信は、標準規格に対応できる「開発力のあるベンダ」や「規模や資金のある施設」は利用できるが、それ以外では個別開発や一方向送信が主体。

今後のシステム構築の形（REST/FHIRで広がる通信）

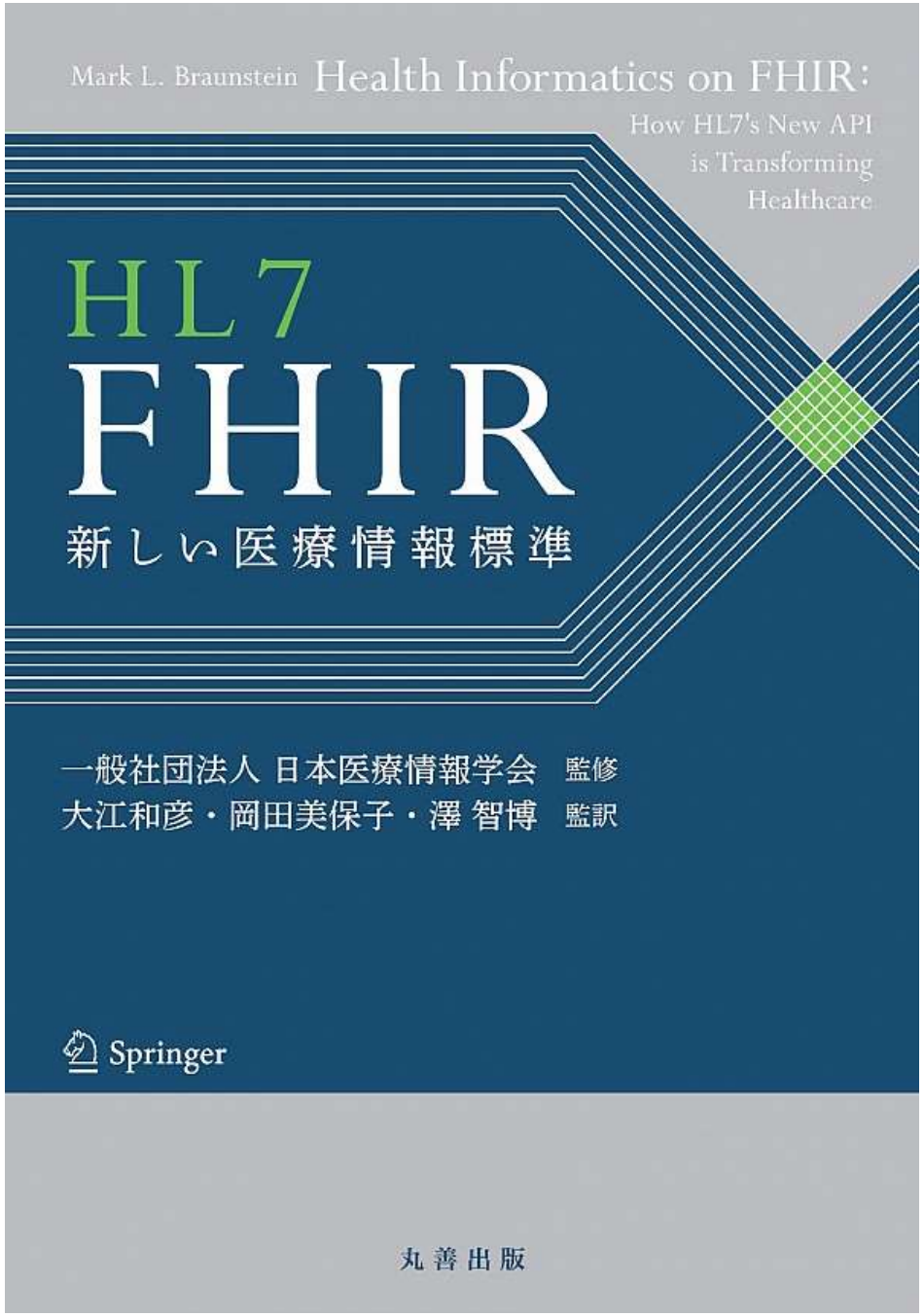


現在の標準規格を利用しながら、これまで連携が難しかったデバイスや利用者との接続を補完し、さらにより広いベンダの参入を可能に。

日本Mテクノロジー学会について

- (主に)医療データベース、プログラミング等に関連する領域の利用、応用、改良、及び普及を行うことを目的とした団体です。
  - 現代のMテクノロジーは関数型のプログラム、ツリー型とテーブル型の両方式のDBを統合できるオブジェクト指向型の開発環境です。
  - より良い医療システムアーキテクチャを探究しています。

- プログラミングやデータベースの技能・知識を持った情報部門担当者の育成を目的としたチュートリアルを年3回実施します
  - 学術部会（主に大学・病院関係）と技術部会（主にベンダ関係）が中心
  - 年次大会ではユーザとベンダのそれぞれの立場からの学術発表、技術討論、チュートリアル等を行い、会員のレベルアップを図っています。



一般社団法人　日本医療情報学会(監修)

大江　和彦(監修 | 翻訳)

岡田　美保子(監修 | 翻訳)

澤 智博(監修 | 翻訳)

医療情報学会監修の
HL7 FHIRの解説書（翻訳）

FHIR®研究会は医療情報学会内でFHIR®の実装、活用を主として検討、提案する研究会です。

FHIR®研究会

Since 2019

トップページ | 新着情報・FAQ | お問い合わせ | アクセス | 団体情報 | その他情報 | 活動報告 | FHIR®研究会活動報告書

トップページ

相互運用性を確保しつつ、医療現場のニーズに真に対応できるHL7 FHIR®の実装、活用を中心とした研究ならびに提案を行っています。

トピックス

- FHIR®研究会は日本HL7協会、NeXEHRS研究会と協力してHL7 FHIR®の普及推進に努めてまいります。
- FHIR®研究会は日本医療情報学会の課題研究会として組織されました。

ニュース

2020年12月05日 2020/12/5FHIR®研究会(Web)を開催しました

2020年10月23日 第40回医療情報学会チュートリアルの申し込みを開始しました

ブログ

活動ブログを開設しました

2019/9/18FHIR®研究会（予定）

2019/10/27岩手医療情報研究会と共催でFHIR®研究会を開催しました

2019年度FHIR®研究会活動報告書を提出しました

第40回医療情報学会チュートリアル申し込み

FHIRの実装・活用について
ユースケースに焦点を当て、
普及推進に努めいます!

第51回日本Mテクノロジー学会大会

MTA2023 in 北の国から 富良野（ハイブリッド開催）

【会期】　2023年9月1日(金)～2日(土)
【大会長】旭川医科大学病院 経営企画部部
准教授・副部長　谷　祐児
【テーマ】「利便性を考えたデータベース
～だれでもデータベース利用へ～」

当会技術委員会では、データベースとプログラムの構造について、真剣に「手を動かす」企業技術者、医療機関の情報部門担当者、大学等の研究者のご参加をお待ちしております！

アンケートにご協力をお願いいたします！