InterSystems
Creative data technology

FHIR新機能のご紹介 ～2024.1～

インターシステムズジャパン株式会社
セールスエンジニア
古薗　知子

2024年6月25日

説明内容

FHIR Object Model

FHIR Object Modelとは

FHIRリソースをオブジェクトモデル化したクラス群 (HS.FHIRModel.R4パッケージ)

- ✓IRIS for Health 2024.1~
- ✓FHIRデータを、より機能的且つ効率的に取り扱うことを可能に

Base Categories	Clinical Categories
HS.FHIRModel.R4.Patient	HS.FHIRModel.R4.AllergyIntolerance
HS.FHIRModel.R4.Practitioner	HS.FHIRModel.R4.AdverseEvent
HS.FHIRModel.R4.PractitionerRole	HS.FHIRModel.R4.Condition
HS.FHIRModel.R4.Organization	HS.FHIRModel.R4.Procedure
HS.FHIRModel.R4.Location	HS.FHIRModel.R4.FamilyMemberHistory
HS.FHIRModel.R4.Encounter	HS.FHIRModel.R4.Observation
HS.FHIRModel.R4.EpisodeOfCare	HS.FHIRModel.R4.DiagnosticReport
HS.FHIRModel.R4.Appointment	HS.FHIRModel.R4.CarePlan
HS.FHIRModel.R4.VerificationResult	HS.FHIRModel.R4.RiskAssessment
…	…

1079個のクラス群

FHIR Object Model
ー開発の背景と目的ー

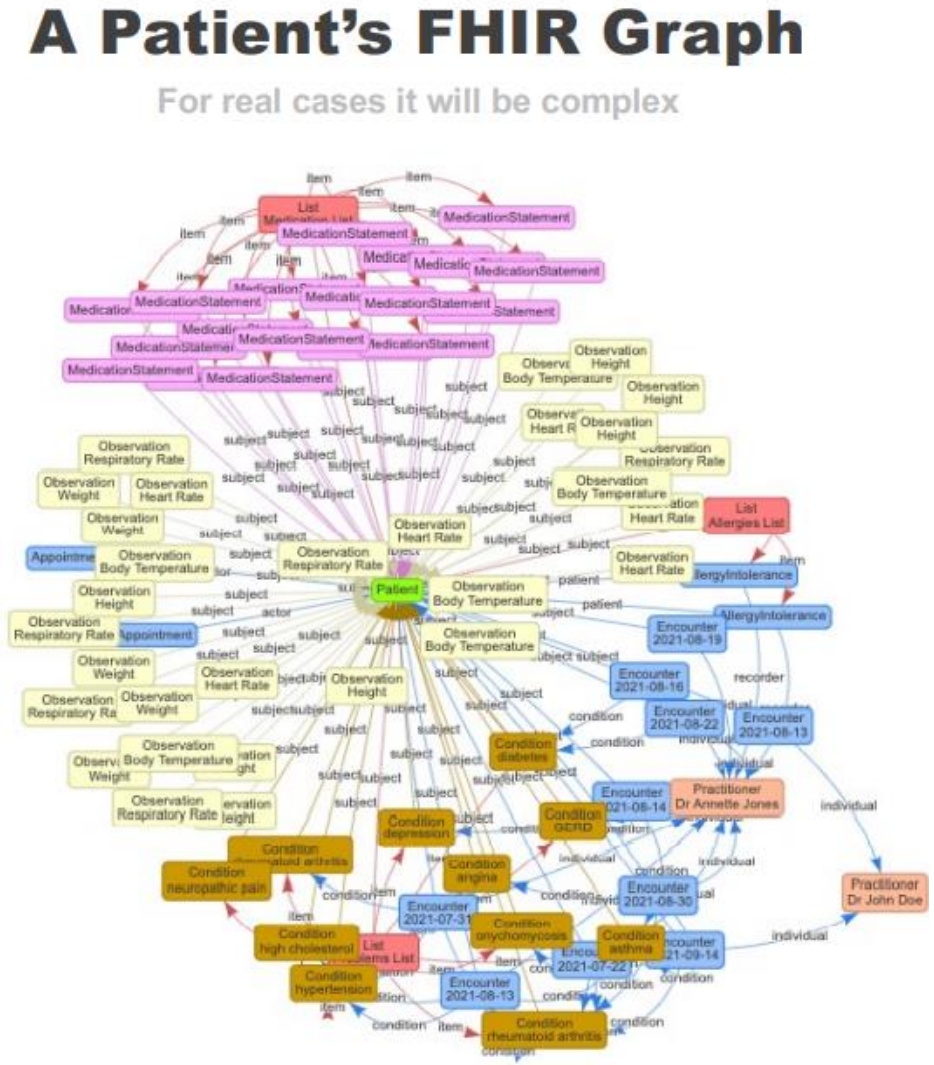
FHIRデータを取り扱う際の課題

- FHIRは、700を超えるリソースとサポート要素(データタイプなど)を含む広範なデータ構造のコレクションを定義しており、それぞれが独自の構造とデータ制約を持つ。

これら多数のリソースと要素の名前、関係、特定の制約を覚えることは、大変困難

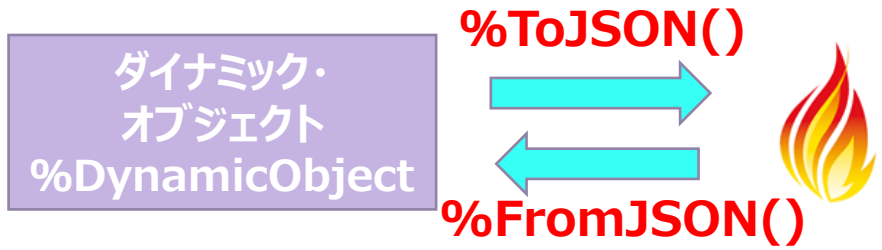
- FHIRリソースは、キーと値のペアのフラットなコレクションに限定されない。入れ子になった要素、コレクション、さまざまなコンポーネント間の複雑な関係などが含まれる。

このような複雑なデータ構造をナビゲートし、操作することは、特に見慣れない要素や深く入れ子になっている要素を扱う場合には、困難な場合がある。



従来の手法(%DynamicAbstractObject)

- FHIR データ(JSON ドキュメント)を%DynamicAbstractObject にロードし、オブジェクトとして取り扱いが可能




```
set mypatient = ##class(%DynamicObject).%FromJSON(fhirstream)
do myPatient.%Set("active", 0, "boolean")
do myPatient.%ToJSON(stream)
do stream.%Save()
```

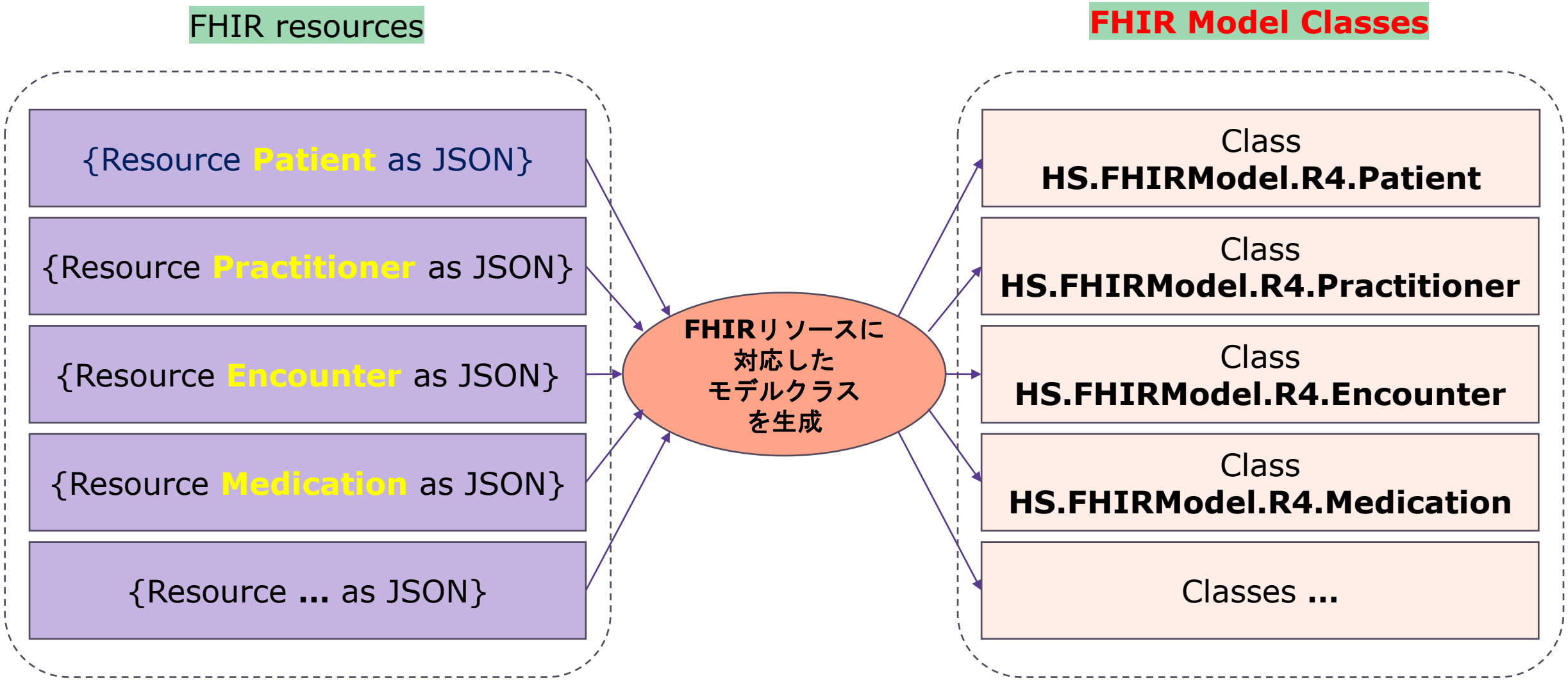
```
set eObj={}
do eObj.%Set("conclusionCode",[].%Set(0,{}.%Set("coding",[].%Set(0,{}))))
```

しかし・・・

- FHIRの構造を自分でナビゲートして操作・編集
- IDEからのガイダンスがない
- 型安全性が保たれない
- 特定の状況に応じたドキュメントが無い

- 定型的なコードを繰り返し書く必要がある
  - 入れ子構造の割り当て
  - 異なる型のコレクションに対する反復処理
  - データモデル間の変換
- 開発者の生産性への影響
  - ツールサポートの欠如
  - 一貫したコーディングパターンの欠如
  - デバッグ・サポートの欠如

Solution：FHIRリソースのオブジェクトモデル化



InterSystems FHIR Object Model

目的：FHIRペイロードを扱う際に開発者に課される、認知的負担を軽減する

開発者にとっての具体的な利点：

- ✓直感的な命名規則
- ✓IDE上のオートコンプリートサポート
- ✓デバッグの簡素化
- ✓わかりやすいドキュメント

生産性の向上 UP!

```
ClassMethod CreatePatient(ident As %String) As %Status
{
    #dim patient As HS.FHIRModel.R4.Patient
    Set patient = ##class(HS.FHIRModel.R4.Patient).%New()
    Do patient.IncludeName()
    Set name = patient.name.MakeEntry()
    Set firstName = "John"
    Do name.IncludeGiven()
    Do name.In
    Do name.gi
    Set name.f
    Do patient
     If patien
     Set fName
     }
     Do patien
    Set patien
    Set patient.birthDate = "1985-05-15"
```

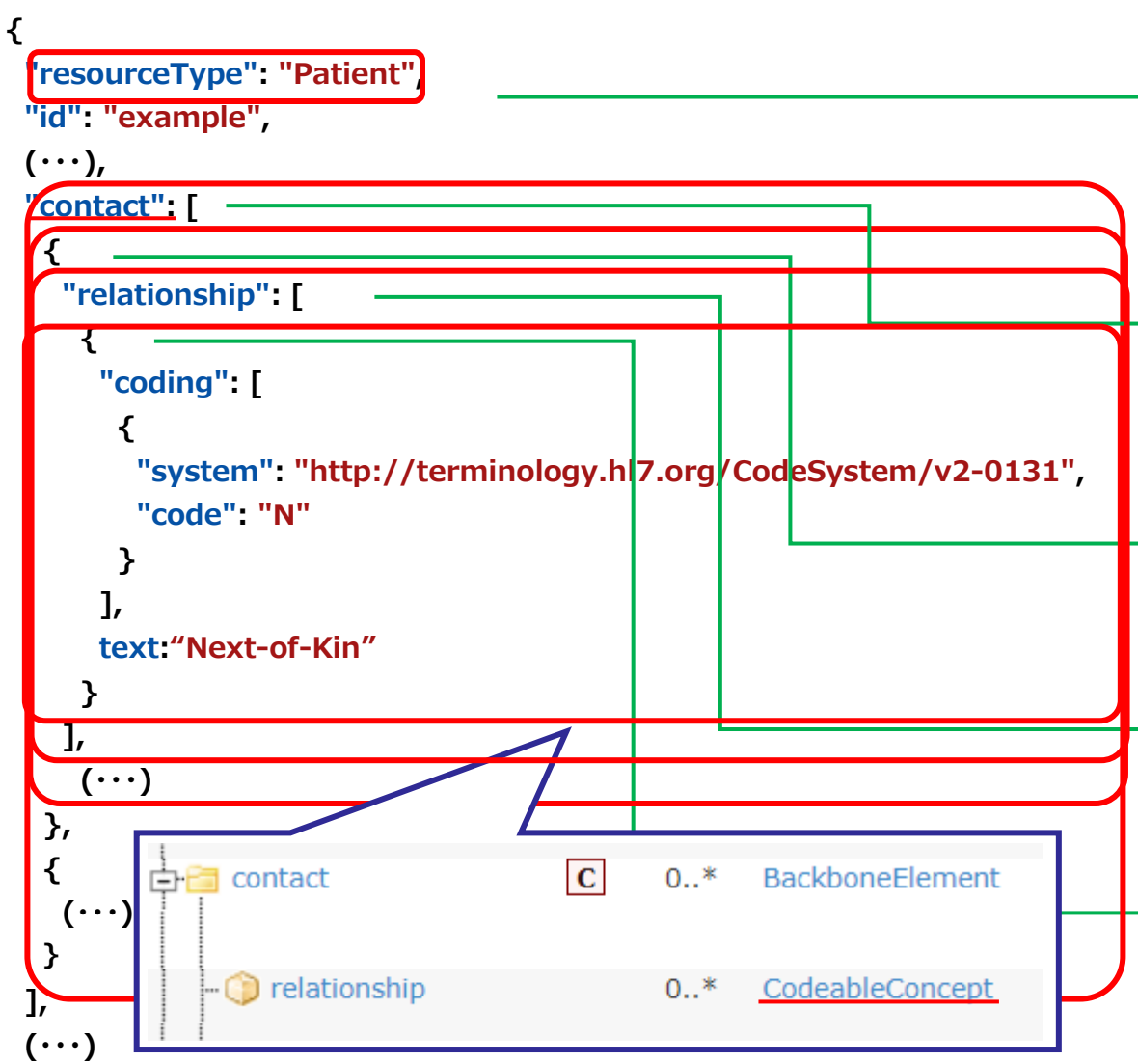
abc IncludeAddress
abc IncludeContact
abc IncludeGiven
abc IncludeIdentifier
abc IncludeLine
abc IncludeName
abc IncludeTelecom
abc interactions
abc EnsureInstance
abc ident
abc identifier

FHIR Object Model
ー機能概要ー

FHIR Model クラスの主な機能と特徴

- 静的クラス表現 ：
  FHIR リソースとデータ要素の構造および命名規則と密接に対応する %RegisteredObject クラスのパッケージ （総数1079個のクラス）
- 使い易いメソッド：
  FHIR仕様の命名規則に準拠してリソース内で入れ子になった構造やコレクションを割り当てるためのメソッドや、繰り返し、クエリ、データ変換などの操作のためのメソッド。IDEのサポートが利用でき、開発がより簡単に。
- 共通の操作メソッド：
  すべてのFHIR Modelクラスは、FHIRデータを操作するための共通のメソッドのセットを実装し、統一的なコーディング・パターンで操作可能。これは動的オブジェクト(%DynamicAbstractObject)に対しても共通。これらの操作メソッドは、%Library.AbstractSet クラスから継承。
- 静的モデル⇔動的オブジェクトの変換：
  静的モデルと動的オブジェクト間で相互に変換可能
- クエリ実行：
  JSON パス言語 (JPL) を使用して、FHIR データ構造に対してクエリを実行可能(apply()メソッド)

FHIRリソース(JSON)とFHIR Modelクラスの対応



リソースクラス：*HS.FHIRModel.R4.[リソース名]*
HS.FHIRModel.R4.Patient.cls

リソース固有のエレメントのコレクション：
HS.FHIRModel.R4.[リソース名]X.SeqOf[リソース名]X[エレメントクラス名]
HS.FHIRModel.R4.PatientX.SeqOfPatientXContact

リソース固有のエレメント：
HS.FHIRModel.R4.[リソース名]X. [エレメントクラス名]
HS.FHIRModel.R4.PatientX.Contact

データタイプのコレクション：*HS.FHIRModel.R4.SeqOf [データタイプクラス名]*
HS.FHIRModel.R4.SeqOfCodeableConcept

データタイプ：*HS.FHIRModel.R4.[データタイプクラス名]*
HS.FHIRModel.R4.CodeableConcept

FHIRデータ操作の流れ

元となるFHIRリソース(JSON)がある場合：

1. FHIR リソースを JSON ペイロードとして受領
2. JSON ペイロードを %DynamicObject (%DynamicAbstractObjectのサブクラス) に変換
3. fromDao() を使用して、%DynamicObjectを該当する FHIRModelクラスのオブジェクトに変換
4. 必要に応じて、FHIR オブジェクトを操作・編集
5. toString() を使用して、FHIR オブジェクトを JSON 形式に戻す

新規にオブジェクトからFHIRリソースを作成する場合：

1. FHIR オブジェクト新規作成・編集
2. toString() を使用して、FHIR オブジェクトを JSON 形式に変換

FHIR Model オブジェクト⇔DAOの変換

fromDao(dao As %DynamicAbstractObject) As HS.FHIRModel.R4.<リソース名>
：DAO から指定された FHIR Model オブジェクトに変換

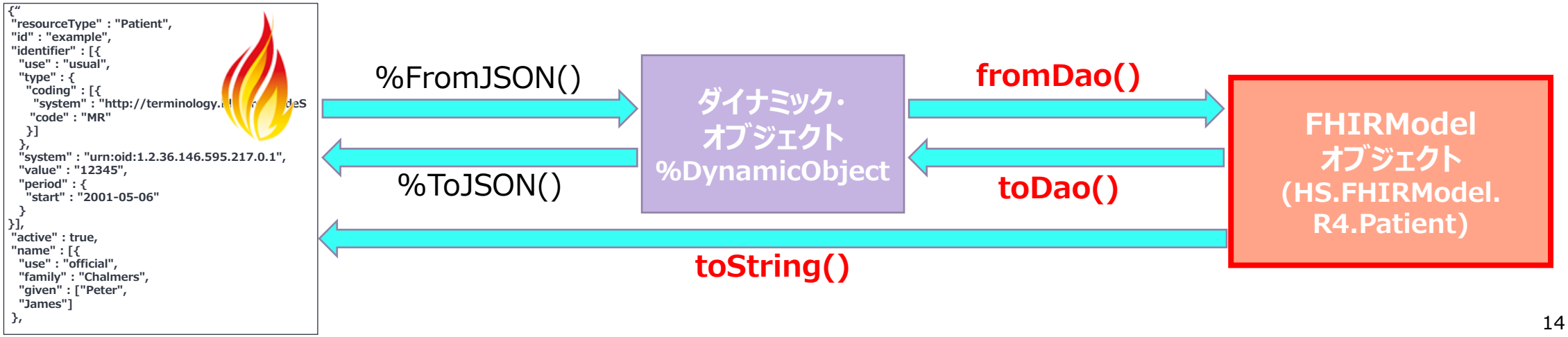
toDao() As %DynamicAbstractObject
：FHIRModelオブジェクトから DAO に変換

toString()
：FHIRModelオブジェクトから直接、
文字列形式のJSON ペイロードに変換

```
//DAOからリソースタイプを取得し、該当のFHIR Model オブジェクトに変換
Set rType = dao.resourceType
Set cls = $CLASSMETHOD("HS.FHIRModel.R4."_rType,"fromDao",dao)

//FHIR Model オブジェクトからDAOに変換
Set newDao = cls.toDao()

//FHIR Model オブジェクトからFHIRペイロードに変換
Set payload = cls.toString()
```



FHIRオブジェクト操作＜値の取得＞

get(key As %DataType) as %Any

：指定されたkeyで識別される要素を取得

keyは、キー・バリュー・コレクションのラベルか、

ゼロベースのシーケンスの数値位置(entry番号)のいずれか

```
//配列項目のentry番号を指定して値を取得
Set patientName = patient.name.get(0)

//ラベルを指定して値を取得
Set familyName = patient.get("gender")

//入れ子になった値の取得
Set patientIdValue=patient.identifier.get(0).get("value")
```


```
{
 "resourceType": "Patient",
 "identifier": [
  {
   "use": "usual",
   "type": {
    "coding": [
     {
      "system": "http://terminology.hl7.org/...,
      "code": "MR"
     }
    ]
   },
   "system": "urn:oid:1.2.36.146.595.217.0.1",
   "value": "12345",
   "period": {
    "start": "2001-05-06"
   },
   "assigner": {
    "display": "Acme Healthcare"
   }
  }
 ],
 "active": true,
 "name": [
  {
   "use": "official",
   "family": "Chalmers",
   "given": [
    "Peter",
    "James"
   ]
  },
  (...)
 ],
 (...),
 "gender" : "female",
```

FHIRオブジェクト操作＜データの追加・更新＞

put(key As %DataType, value As %Any) as %AbstractSet

：key-value のペアを新しく追加、もしくは、既にある場合は更新。

```
//nameの０番目の要素に「”use”:”offcial”」を追加
Do patient.name.get(0).put("use","official")
```

```
"name": [
  {
    "use": "official",
  }
]
```

FHIRオブジェクト操作＜入れ子構造の操作＞

Include[ElementName]()
：要素またはコレクションの入れ子になったデータ構造を
　リソース・オブジェクトに追加する

MakeEntry()
：コレクション・オブジェクトの要素を生成

add(value As %Any) as %AbstractSet
：繰り返し項目のエントリを追加する

```
//Addressコレクション構造を追加
Do patient.IncludeAddress()
//Addressオブジェクトを生成
Set addr = patient.address.MakeEntry()
Do addr.IncludeLine()
Set line1 = "123 Main Street"
Do addr.line.add(line1)
<中略>
//中の要素の値をセットしたAddressオブジェクトをリソースに追加
Do patient.address.add(addr)
```

```
{
 "resourceType" : "Patient",
 (…),
 "address": [
  {
    "line": [
     {“123 Main Street“}
    ],
    "city": “Boston",
    "state": “MA",
    "postalCode": “02111"
  }
 ]
 (…)
}
```

ナビケーション

iterator() as %Iterator

：%Iterator オブジェクトのインスタンスを返し、
順序付きリストと名前付き順序なしコレクションの概念を使用して、
すべての要素にナビゲートする

iterator.hasNext()

：コレクション内の次の要素があればtrue (1) を返す

iterator.next()

：次の要素を取得

```
//telecomエレメント(繰り返し項目)のイテレーターを定義
 Set telecomItr = patient.contact.get(0).telecom.iterator()
While telecomItr.hasNext() {
   Set telecom = telecomItr.next().value
   if telecom.system = "phone" {
     Write !, telecom.value
 }
}
```

```
"contact": [
 {
  "telecom": [
  {
   "system": "phone",
   "value": "(03) 5555 6473",
   "use": "work",
   "rank": 1
  },
  {
   "system": "phone",
    "value": "(03) 3410 5613",
    "use": "mobile",
    "rank": 2
  },
  {
   (...)
  }
  ],
```

JSONパス言語によるクエリ実行

apply(expression As %Any) as %AbstractSet

：expressionに、SQL/JSON Path Language (JPL/JSON パス言語) 式を指定し、FHIRデータ構造に対してクエリを実行する

ドキュメント：SQL/JSON パス言語式の使用
https://docs.intersystems.com/irisforhealth20241/csp/docbookj/DocBook.UI.Page.cls?KEY=RSQL_jsontable#RSQL_jsonetable_jpl

```
//Practitioner(doctor)のtelecomコレクションからemail情報を抽出する
Set doctor = ##class(HS.FHIRModel.R4.Practitioner).fromDao(dao)
Set key = "system"
Set value = "email"
Set query = "$[*]?(@."_key_"=="'_value_"')"
Set email = doctor.telecom.apply(query)
```

```
{
  "resourceType": "Practitioner",
  "id": "f001",
  (...)
  "telecom": [
    {
      "system": "phone",
      "value": "0205568263",
      "use": "work"
    },
    {
      "system": "email",
      "value": "E.M.vandenbroek@bmc.nl",
      "use": "work"
    },
    {
      "system": "fax",
      "value": "0205664440",
      "use": "work"
    }
  ],
  (...)
 ]
}
```

InterSystems FHIR Object Model(再掲)

目的：FHIRペイロードを扱う際に開発者に課される、認知的負担を軽減する

開発者にとってのメリット：

- ✓直感的な命名規則
- ✓IDE上のオートコンプリートサポート
- ✓デバッグの簡素化
- ✓わかりやすいドキュメント

生産性の向上 UP!

```
ClassMethod CreatePatient(ident As %String) As %Status
{
    #dim patient As HS.FHIRModel.R4.Patient
    Set patient = ##class(HS.FHIRModel.R4.Patient).%New()
    Do patient.IncludeName()
    Set name = patient.name.MakeEntry()
    Set firstName = "John"
    Do name.IncludeGiven()
    Do name.In
    Do name.gi
    Set name.f
    Do patient
     If patien
     Set fName
     }
     Do patien
    Set patien
    Set patient.birthDate = "1985-05-15"
```

```
abc IncludeAddress
abc IncludeContact
abc IncludeGiven
abc IncludeIdentifier
abc IncludeLine
abc IncludeName
abc IncludeTelecom
abc interactions
abc EnsureInstance
abc ident
abc identifier
```

今後の計画

- FHIR R5 に対応した モデルクラス
- カスタムプロファイルに対応した モデルクラス
- DTLエディタにおけるCDA/FHIR/等のNativeサポート

その他の
FHIR新機能/強化機能

JsonAdvSQL
(FHIR検索パフォーマンスの向上)
[2024.1～]

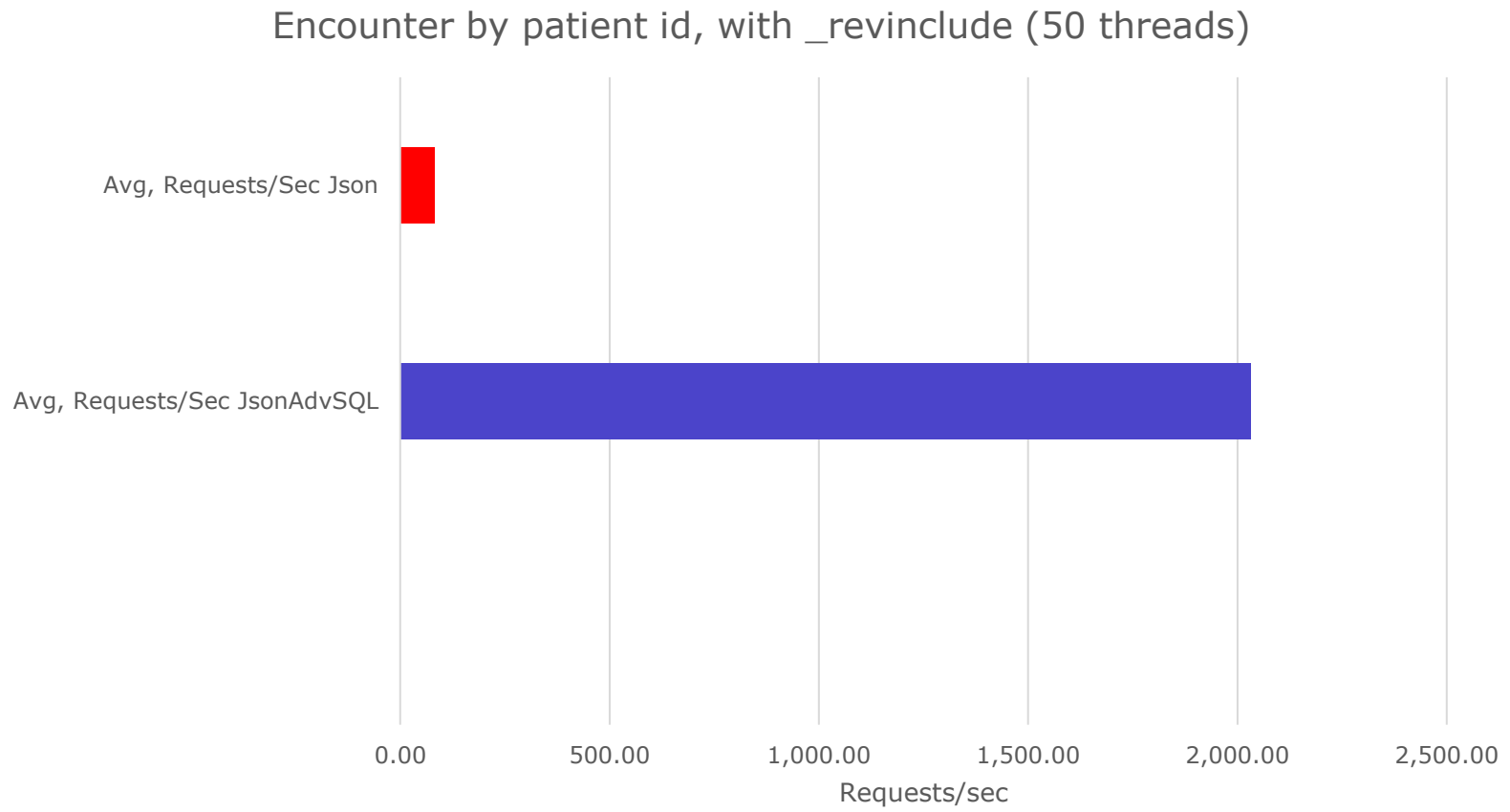
JsonAdvSQL

新しいFHIR検索ストラテジである、JsonAdvSQLを導入。

これにより、フレームワークの成熟に伴って、より複雑化するFHIRクエリに対応し、FHIRクエリの大幅な**パフォーマンス向上**を実現。

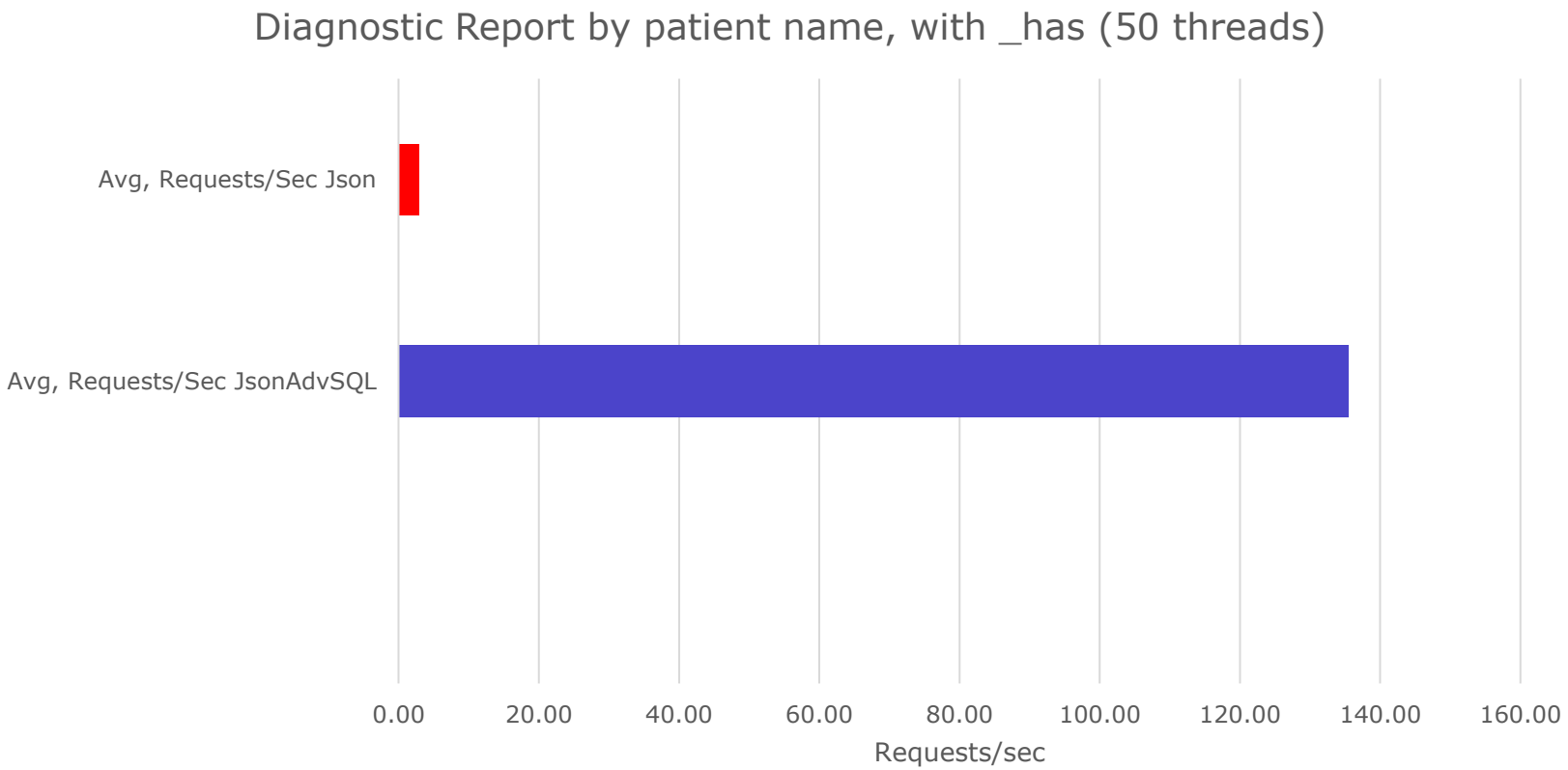
また、FHIR検索の種類やサポートするオプションなどの機能強化も行い、より幅広いFHIR検索が可能に。

ベンチマーク例①：
Patient IDを指定して、_revincludeでEncounterを検索



※_revinclude：逆方向の包含

ベンチマーク例②：
Patient IDを指定して、条件に合うDiagnosticReportリソースから参照されているEncounterリソースを_hasで検索



※_has：リバースチェーン検索

JsonAdvSQL

リリースノート：

https://docs.intersystems.com/irisforhealth20241/csp/docbookj/DocBook.UI.Page.cls?KEY=HXIHRN_new20241#HXIHRN_new20241_healthinterop_jsonadvsql

従来のストラテジで作成したリポジトリを、JsonAdvSQLストラテジベースにダウンタイム無しで移行可能。
https://docs.intersystems.com/irisforhealth20241/csp/docbookj/DocBook.UI.Page.cls?KEY=HXFHIR_SQL_Legacy#HXFHIR_SQL_upgrade

プロファイルベースの Validation

[2023.3～]

プロファイルベースのバリデーション

2023.3からカスタムプロファイルベースのバリデーションにも対応。

POST <*FHIR Endpoint*>/<*Resource Type*>/$validate?profile=<*Profile URL*>|<*Profile Version Number*>

ドキュメント：

https://docs.intersystems.com/irisforhealth20241/csp/docbook/DocBook.UI.Page.cls?KEY=HXFHIR_server_arch_supported#HXFHIR_server_arch_supported_operation_validate

Developer Community記事

https://jp.community.intersystems.com/node/562776

SMART Scopes

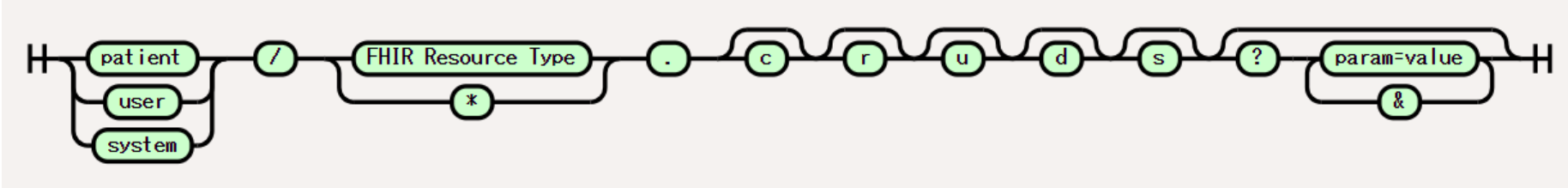
SMART Scopes

InterSystems FHIR サーバは、SMART App Launch 2.xで定義されている、“CRUDS” (作成/読み取り/更新/削除/検索) スタイルの許可を使用する SMART スコープをサポート。(2024.1～)

例：

patient/*.rs	現在の患者に関するあらゆるリソースの読み取りと検索を許可
user/*.cruds	現在のユーザーがアクセスできるすべてのリソースの読み取りと書き込みの許可

シンタックス：



詳細は、下記の、HL7.orgの ドキュメントをご参照

https://hl7.org/fhir/smart-app-launch/scopes-and-launch-context.html#scopes-for-requesting-fhir-resources

※下位互換性を確保するため、引き続き 1.0 FHIR スコープもサポート。

まとめ

- FHIR Object Model
- JsonAdvSQL
- プロファイルベースのValidation
- 新しいSMART Scopesへの対応

IRIS for HealthのFHIR機能はこれからも進化します！