

IHEチュートリアル 「情報をつなぐ～FHIRの利活用～」

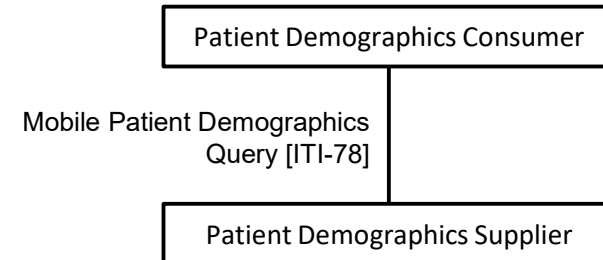
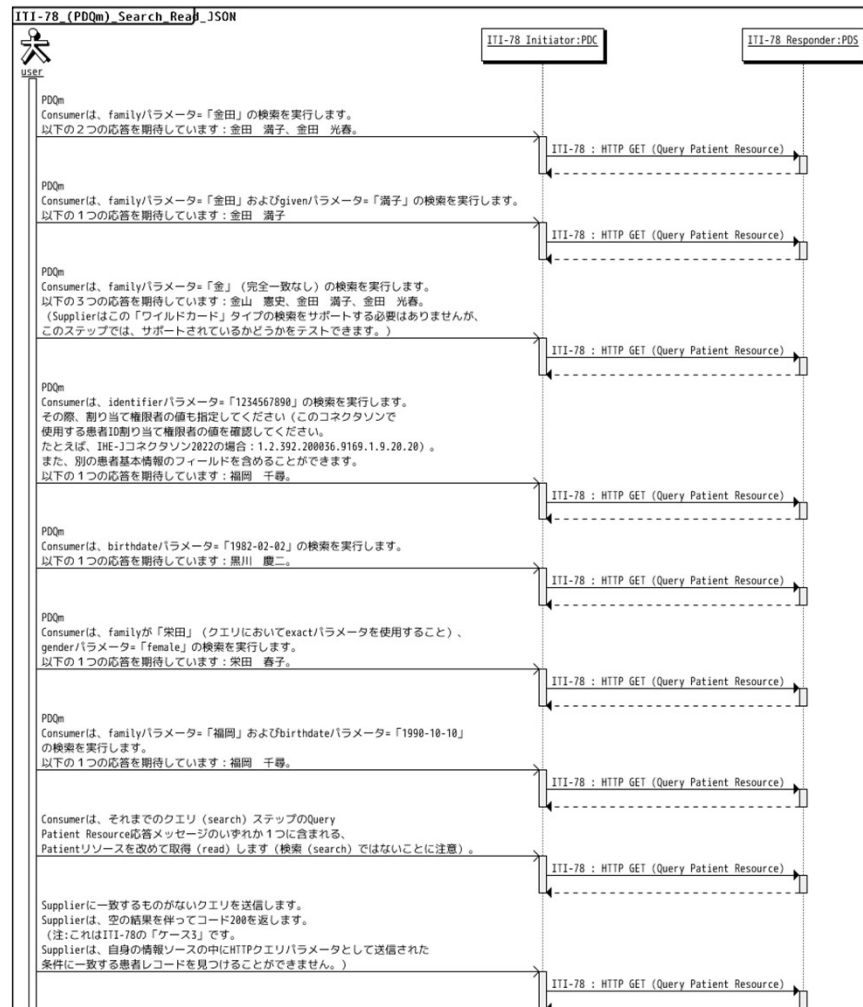
FHIRの実装事例

日本IHE協会 ITI技術委員会
宮川 力

今回の発表内容

- .NET(C#)におけるFHIRの実装事例(PDQm)を紹介します。
- オープンソースのライブラリになります。利用にあたっては、ライセンス等確かめた上でご利用ください。
- 実装にあたりポイントとなる部分(全体公開ではない)をご紹介します。

IHE PDQmとは



- Patientリソースに対するクエリを実現する
- 患者のIDや氏名、カナ、生年月日、性別から、対象を特定し、Patientリソースを取得する
- ↓
- IHEコネクタソンでの検証可能なFHIR関連のプロファイルのひとつ
- 昨年は、韓国のベンダとも接続確認を実施した

ライブラリ紹介

firely-net-sdk

About Firely .NET SDK

Firely .NET SDK provides a set of tools and libraries that make it easier for developers to work with FHIR resources in .NET applications. It's open source and can be downloaded from the GitHub repository. It is regularly updated to keep up with the latest developments in the FHIR standard.



RESTful API

A client library for interacting with FHIR servers using the FHIR RESTful API. Easily retrieve, create, update, and delete FHIR resources using standard HTTP methods.



FHIR Data Model

Easier to work with FHIR resources in .NET applications because of the classes and interfaces that represent the different types of FHIR resources.



Serialization

Tools for serializing and deserializing FHIR resources in different formats, including JSON and XML.



Profiling

Working with FHIR profiles, including the ability to create profiles with the SnapshotGenerator. Define custom profiles, and use them to validate resources against custom constraints.



Support for multiple FHIR versions

Work with any version of the FHIR standard, from the earliest releases to the latest and most cutting-edge versions, and ensure that your applications will continue to work seamlessly as the standard evolves over time.



FHIRPath

Perform complex searches and manipulations on FHIR resources to retrieve specific data elements or transform them into new representations.

[Read our documentation](#) →

<https://fire.ly/products/firely-net-sdk/#about>

- .NET用ライブラリ
- オープンソース
BSD 3-Clause Licenseライセンス
- Data Modelや
Serialization、Validation、
FHIR Path等の実装の面倒な処理が搭載されている
特にVersionごとのData Modelは実装する上でJsonのスペルミスや構造間違いの防止に有用である
- クライアントは網羅されているが、サーバ関連処理は実装が必要

サーバ側の実装

- サーバ側の実装例の一部です。
- 説明の流れ
 - ライブラリ参照方法
 - 検索にヒットしたData Model作成
 - Data ModelのSerialize方法
 - RestAPIの作成の順で説明します。
- クライアントは、実装例はインターネット検索すると多くあるので探してみてください。

ライブラリの利用 nugetでの検索

The screenshot shows the NuGet package manager interface. The search bar at the top contains 'HL7.Fhir.R4'. The search results list several packages, with 'HL7.Fhir.R4' selected. The details for 'HL7.Fhir.R4' are shown on the right, including the version '5.8.1' and the 'インストール' (Install) button. Below the details, the '依存関係' (Dependencies) section is highlighted with a red box, showing that the package depends on 'net8.0' and 'HL7.Fhir.Conformance (>= 5.8.1)'. On the left, a section titled '.NET Standard のバージョンの選択' (Select .NET Standard version) shows a table of supported versions for different .NET implementations.

.NET Standard のバージョンの選択

1.0 1.1 1.2 1.3 1.4 1.5 1.6 2.0 2.1

.NET Standard 2.0 では、37,118 個の API のうち、32,638 個が使用可能です。

🔄 テーブルを展開する

.NET 実装	バージョンのサポート
.NET と .NET Core	2.0、2.1、2.2、3.0、3.1、5.0、6.0、7.0、8.0
.NET Framework ¹	4.6.1 ² 、4.6.2、4.7、4.7.1、4.7.2、4.8、4.8.1

HL7.Fhir.R4 作成者: Firely (info@fire.ly) and contributors, 5.52M 件のダウンロード 5.8.1
Firely's SDK for working with HL7 FHIR R4. This is the root package for the SDK includes core functionality to working with RESTful FHIR servers, POJO classes for FHIR, parsin...

HL7.Fhir.R4B 作成者: Firely (info@fire.ly) and contributors, 299K 件のダウンロード 5.8.1
Firely's SDK for working with HL7 FHIR R4B. This is the root package for the SDK includes core functionality to working with RESTful FHIR servers, POJO classes for FH...

HL7.Fhir.Conformance 作成者: Firely (info@fire.ly) and contributors, 913K 件のダウンロード 5.8.1
Conformance POJOs and related functionality for Firely's HL7 FHIR SDK. Is a dependency for the FHIR release-specific NuGet packages, but could be used separat...

HL7.Fhir.Specification.R4 作成者: Firely (info@fire.ly) and contributors, 689K 件のダウンロード 5.8.1
Metapackage including Firely's SDK Base Class Library and specification conformance data for HL7 FHIR R4. Its sole purpose is to provide a backwards-compatible package...

HL7.Fhir.Specification.R4B 作成者: Firely (info@fire.ly) and contributors, 186K 件のダウンロード 5.8.1
Metapackage including Firely's SDK Base Class Library and specification conformance data for HL7 FHIR R4B. Its sole purpose is to provide a backwards-compatible packag...

HL7.Fhir.R4 作成者: Firely (info@fire.ly) and contributors 5.8.1
Firely's SDK for working with HL7 FHIR R4. This is the root package for the SDK includes core functionality to working with RESTful FHIR servers, POJO classes for FHIR, parsing/serialization of FHIR data and working with conformance data and terminologies.

バージョン: 5.8.1
作成者: Firely (info@fire.ly) and contributors
ライセンス: BSD-3-Clause
お読みください: 「お読みください」を表示
ダウンロード: 5,521,685
公開日: 2024年5月2日 (2024/05/02)
プロジェクト URL: <https://github.com/FirelyTeam/firely-net-sdk>
不正使用を報告: <https://www.nuget.org/packages/HL7.Fhir.R4/5.8.1/ReportAbuse>

タグ: HL7, FHIR, Firely, SDK, POJO, Fhir, client, Terminology, StructureDefinition, parsing, serialization, server

依存関係

- net8.0
- HL7.Fhir.Conformance (>= 5.8.1)
- .NETStandard, Version=v2.0
- HL7.Fhir.Conformance (>= 5.8.1)

- nugetにてHL7.Fhir.R4で検索する
ダウンロードは5百万超
- 最新は.net8および.net standard v2.0をサポート
standard v2.0が提供されているため、多くの.net バージョンで利用可能

<https://learn.microsoft.com/ja-jp/dotnet/standard/net-standard>

返却オブジェクト Patientリソースのインスタンス作成

```
using Hl7.Fhir.Model;
```

```
private Patient CreateMale()
{
    var medicalInstitutionCode = "9999999999";
    var patientId = "123456789";

    var patient = new Patient();
    patient.Id = "000000001";
    patient.Identifier.Add(
        new Identifier(
            $"urn:oid:1.2.392.100495.20.3.51.[medicalInstitutionCode]",
            patientId)
    );

    patient.Name.Add(KanjiName());
    patient.Name.Add(KanaName());

    patient.Gender = AdministrativeGender.Male;
    patient.BirthDate = "1970-01-01";

    return patient;
}

private HumanName KanjiName()
{
    var name = new HumanName().WithGiven("太郎").AndFamily("山田");

    name.Use = HumanName.NameUse.Usual;
    name.Text = "山田 太郎";
    name.Extension.Add(
        new Extension()
        {
            Url = "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
            Value = new Code("IDE")
        }
    );

    return name;
}

private HumanName KanaName()
{
    var name = new HumanName().WithGiven("タロウ").AndFamily("ヤマダ");

    name.Use = HumanName.NameUse.Usual;
    name.Text = "ヤマダ タロウ";
    name.Extension.Add(new Extension()
    {
        Url = "http://hl7.org/fhir/StructureDefinition/iso21090-EN-representation",
        Value = new Code("SYL")
    });

    return name;
}
```

- HAPIと似たような実装可能
- DataModelがあらかじめ定義されているため、少ないソース量で実装可能である
- またスペルミスや構造の間違い等も防止できる
- データ取得部分をDBからのデータを流し込む形になる
- BirthDateが文字型になっているなどFHIRに特化した方になっている点は注意が必要になる
- SDKの参考ページ
<https://docs.fire.ly/projects/Firely-NET-SDK/model/patient-example.html>

注意事項 Data ModelのJSONへのSerialize

```
/// <summary>
/// 出力処理
/// </summary>
/// <param name="bundle"></param>
private void Serialize(Bundle bundle, string fileName)
{
    if (!System.IO.Directory.Exists(Settings.Output))
        System.IO.Directory.CreateDirectory(Settings.Output);

    var path = System.IO.Path.Combine(Settings.Output, fileName);
    using (var tx = new System.IO.StreamWriter(path))
    using (var writer = new JsonTextWriter(tx))
    {
        writer.Formatting = Settings.JsonFormartPretty ? Formatting.Indented : Formatting.None;
        var serializer = new FhirJsonSerializer();
        serializer.Serialize(bundle, writer);
    }
}
```

- ファイル等に出力する場合、FhirJsonSerializerクラスを利用する
※Data Modelインスタンスを普通にSerializeしても期待する形にならない。
- Data ModelとJsonの形が一致しないことが原因
子オブジェクトがValueのみの場合に、親プロパティに直接セットするなど、FHIR仕様があるため専用のクラスを利用する必要あり。

Bundleリソースへ作成

```
// example searchset Bundle setup, fictional data only
var search_response = new Bundle();
search_response.Type = Bundle.BundleType.Searchset;

// adding some metadata
search_response.Id = "[insert temporary uuid here, or real id if you store this Bundle]";
search_response.Meta = new Meta()
{
    VersionId = "1",
    LastUpdatedElement = Instant.Now()
};

// we assume the search has already taken place on your database, and the resulting
// resources are available to us in a list called 'dataset'
search_response.Total = dataset.Count;

// for searches, we need to fill in the 'self' link, that represents the search request
// as understood by the server, e.g. "http://myserver.org/fhir/Patient?name=steve"
// if you are paging the response, also fill in the other relevant links, like 'next',
// 'last', etc.
search_response.SelfLink = new Uri("[search request]");

foreach (var r in dataset)
{
    var full_url = r.ResourceBase.ToString() + r.ResourceType.ToString() + r.Id;

    // instead of using AddResourceEntry, we use the search variant to also include
    // the search mode
    search_response.AddSearchEntry(r, full_url, Bundle.SearchEntryMode.Match);
}

// the Bundle is now ready to be sent to the requester

// walking through the entries in the Bundle:
foreach (var e in search_response.Entry)
{
    // Let's write the fully qualified url for the resource to the console:
    Console.WriteLine("Full url for this resource: " + e.FullUrl);

    // Do something with this resource, for example write human readable text of
    // the resource to the console:
    var resource = (DomainResource)e.Resource;
    Console.WriteLine("Human readable text of this resource: " + resource.Text);
}
```

- 検索結果リソース(dataset)を取得後、バンドルに詰め込んで返却する。
- Firely社のページにサンプルソースが提示されている

<https://docs.fire.ly/projects/Firely-NET-SDK/en/5.3.0/model/bundle-example.html>

RestAPIを作成する

Bundleリソースを作成し返却するのみ

```
[HttpGet]
public async Task<ActionResult> GetAsync([FromQuery] GetArgs args)
{
    var logId = LogId.Generate();
    var resources = await _repository.FindAllAsync(logId, args.ToFindArgs());
    var bundle = this.CreateBundle(resources);
    return this.Ok(bundle);
}
```

引数用のクラス

```
/// <summary>
/// <see cref="GetAsync"/> メソッドの引数
/// </summary>
public class GetArgs : QueryArgs
{
    /// <summary>患者番号</summary>
    [Name("identifier")]
    public string? Identifier { get; set; }

    #region 氏名
    /// <summary>氏名 (前方一致) ※「部分一致」または「完全一致」を指定している場合は無効</summary>
    [Name("name")]
    public string? Name { get; set; }

    /// <summary>氏名 (部分一致) ※「完全一致」を指定している場合は無効</summary>
    [Name("name:contains")]
    public string? NameContains { get; set; }

    /// <summary>氏名 (完全一致) </summary>
    [Name("name:exact")]
    public string? NameExact { get; set; }
    #endregion
}
```

- 今回はAspNetCoreのプロジェクト例
- 引数より、対象となるPatientリソースの入ったBundleリソースを生成し返却するだけ
- クエリ引数はAspNetの機構を利用して取得する
それぞれの解析の実装は別途必要

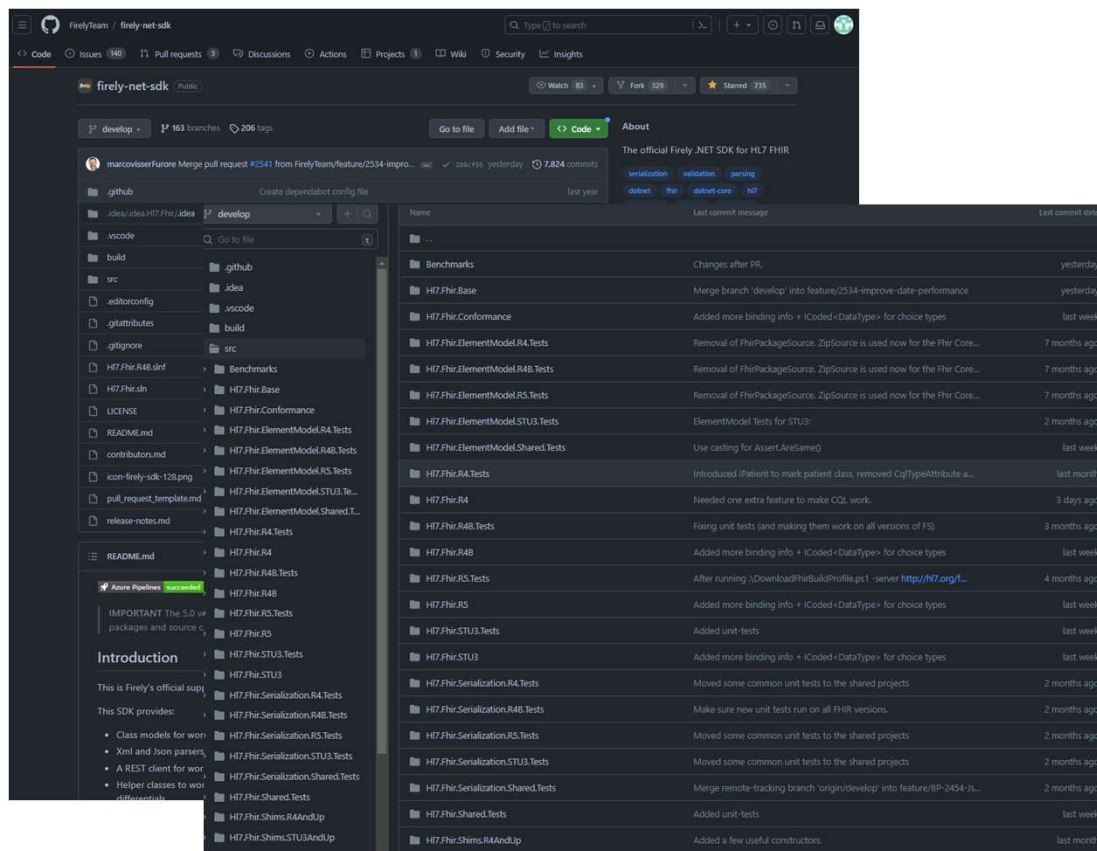
コントローラーへのオプション追加

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers()
    .AddJsonOptions(options =>
    {
        options.JsonSerializerOptions.ForFhirFormat(typeof(Bundle).Assembly);
        options.JsonSerializerOptions.Converters.Add(new JsonStringEnumConverter());
    }
);
```

```
static class JsonExtensions
{
    /// <summary>Serializer設定</summary>
    public static JsonSerializerOptions ForFhirFormat(
        this JsonSerializerOptions options,
        Assembly modelAssembly)
    {
        options.Encoder = System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscaping;
        var converter = new FhirJsonConverterFactory(modelAssembly, new(), new());
        options.Converters.Add(converter);
        return options;
    }
}
```

- 初期化時にJsonSerializeOptionsに対するConverterの追加が必要
- FhirJsonConverterFactoryクラスを利用することで、前述のJsonのSerializeの特別処理と同様結果が得られる。
- EnumStringConverter(Enumを文字列に置き換える)も追加が必要
- これにより、FHIRのJson型が生成される。

よりSDKを理解するには GitHubでのソース参照



- ソースはGitHub上で参照可能
- テストプロジェクトを参考に、どのような処理が可能かの確認できる。

<https://github.com/FirelyTeam/firely-net-sdk>

ご参加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 または、
アンケート用紙にてお願いします。