IHEチュートリアル
「情報をつなぐ～FHIRの利活用～」

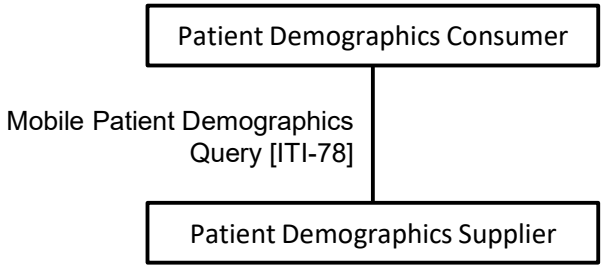
FHIRの実装事例

日本IHE協会 ITI技術委員会
宮川　力

今回の発表内容

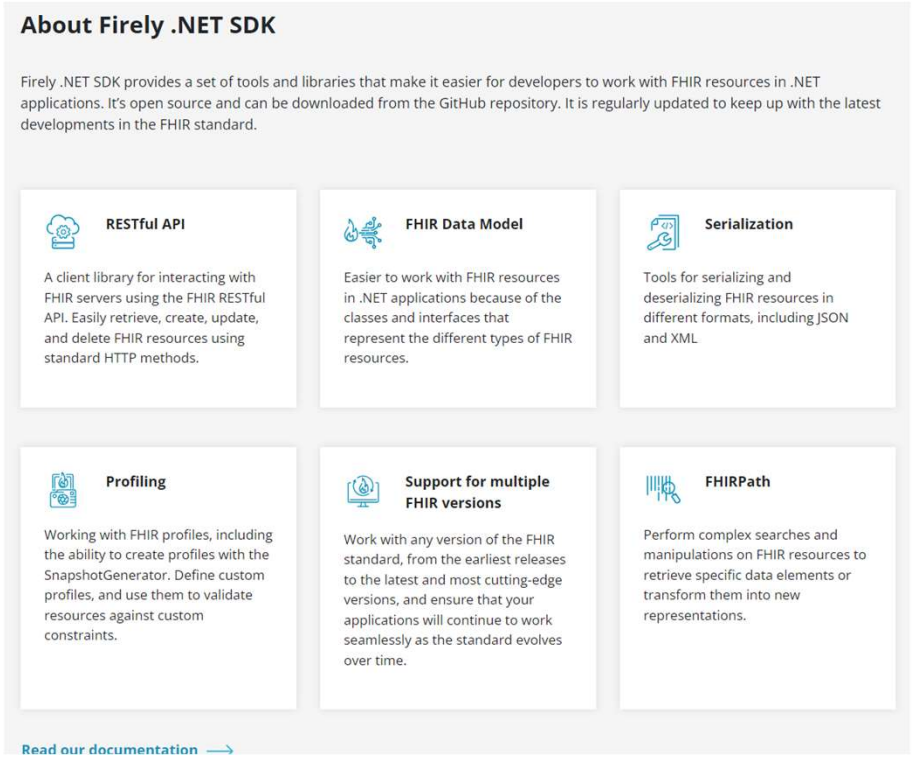
- .NET(C#)におけるFHIRの実装事例(PDQm)を紹介します。
- オープンソースのライブラリになります。利用にあたっては、ライセンス等を確かめた上でご利用ください。
- 実装にあたりポイントとなる部分(全体公開ではない)をご紹介します。

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



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

- 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 = "00000001";
    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<IActionResult> 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

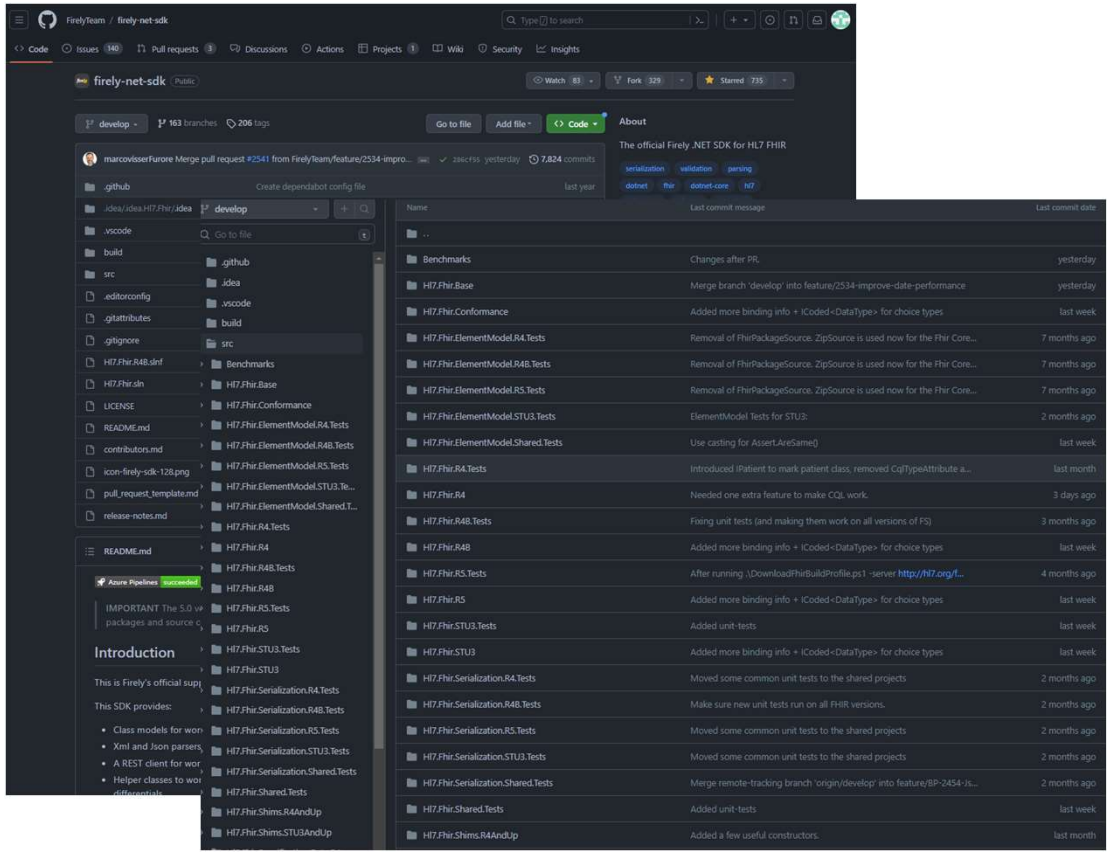
コントローラーへのオプション追加

```
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
- FhirJsonConvertFactory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かを確認できる。

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。