Article
Lorenzo Scalese · Oct 5, 2022 14m de lecture

Utilisation de SUSHI pour la création de profils FHIR, partie 2

Bonjour, chers développeurs !

Cet article est le deuxième d'une série sur la façon d'utiliser SUSHI, un outil de création de profils FHIR, en tant que technologie associée à FHIR. Six mois se sont écoulés avant cette deuxième partie.

Dans la précédente partie 1, nous avons abordé les questions suivantes : qu'est-ce que FHIR, qu'est-ce que le profil FHIR, qu'est-ce que FHIR Shorthand ? Et quel genre d'outil est SUSHI ? Que peut-il produire ? Avec des captures d'écran pour des exemples de résultats.

Cet article présente un exemple d'utilisation réelle d'un profil créé avec SUSHI, dans lequel une **Extension** est ajoutée à une ressource Patient à l'aide de SUSHI, et un nouveau **SearchParameter** est défini pour l'élément de cette Extension, jusqu'à ce que le nouveau SearchParameter puisse être utilisé dans l'IRIS for Health's FHIR Repositoy jusqu'à ce que le nouveau SearchParameter puisse être utilisé.

Mise à jour de SUSHI

Je suis désolé de m'écarter du sujet principal, mais si vous êtes comme moi et que vous n'avez pas touché à SUSHI depuis un moment, vous devriez mettre à jour SUSHI.
Au cours des six derniers mois, SUSHI a fait l'objet d'une importante mise à jour, et la version 2.0.0 est sortie en août. La dernière version au moment de la rédaction de cet article était SUSHI 2.1.1.

Comme décrit dans ce lien, la mise à jour est la commande suivante de même que l'installation.

```
$ npm install -g fsh-sushi
```

Vous pouvez vérifier la version en exécutant sushi -version.
De même, l'outil IG Publisher, qui crée un ensemble de fichiers HTML pour le Guide de mise en œuvre sur la base des Profils générés par SUSHI, peut être mis à jour en exécutant la commande **updatePublisher** .

Création de fichiers FISH

Tout d'abord, créez un projet en utilisant la commande sushi --init comme précédemment. Dans cet article, nous allons modifier le fichier **patient.fsh** généré par le modèle.

Cette fois, nous ajouterons une extension de type **lieu de naissance**, qui est une chaîne de caractères String représentant le lieu de naissance du patient, et nous définirons également un SearchParameter pour ce lieu de naissance, afin de pouvoir effectuer une recherche par lieu de naissance du patient !

Ajout d'Extension

Tout d'abord, ajoutez la définition suivante pour ajouter Extension.

Comme dans US Core et JP Core, le type Adresse est habituellement utilisé, mais ici il s'agit simplement du type

String

```
Extension: BirthPlace
Id: birthPlace
Title: "???"
Description: "???????string??????"
* ^url = "http://isc-demo/fhir/StructureDefinition/patient-birthPlace"
* value[x] only string
```

Chaque élément correspond à la StructureDefinition d'Extension comme suit. Certains éléments sont placés à plusieurs endroits. Certaines informations, comme la version du fhir de base et la version de cette Extension elle-même, proviennent du fichier sushi-config.yml.

Entrée SUSHI	Entrée StructureDefinition correspondante
Extensions	nom
Id	id
Titre	title/differencial.element[id=Extension].short
Desctiption	description/differencial.element[id=Extension].definition
^url	url//differencial.element[id=Extension.url].fixedUri
valeur[x]	differencial.element[id=Extension.value[x]].type.code

La StructureDefinition réelle d'Extension générée.
Il est difficile de créer cela à partir de rien et à la main, mais avec SUSHI, c'est relativement facile.


```
{
  "resourceType": "StructureDefinition",
  "id": "birthPlace",
  "url": "http://isc-demo/fhir/StructureDefinition/patient-birthPlace",
  "version": "0.1.0",
  "name": "BirthPlace",
  "title": "???",
  "status": "active",
  "description": "???????string??????",
  "fhirVersion": "4.0.1",
  "mapping": [
    {
      "identity": "rim",
      "uri": "http://hl7.org/v3",
      "name": "RIM Mapping"
    }
  ],
  "kind": "complex-type",
  "abstract": false,
  "context": [
    {
      "type": "element",
      "expression": "Element"
    }
  ],
  "type": "Extension",
  "baseDefinition": "http://hl7.org/fhir/StructureDefinition/Extension",
  "derivation": "constraint",
  "differential": {
    "element": [
      {
        "id": "Extension",
        "path": "Extension",
        "short": "???",
```

```
        "definition": "???????string??????"
      },
      {
        "id": "Extension.extension",
        "path": "Extension.extension",
        "max": "0"
      },
      {
        "id": "Extension.url",
        "path": "Extension.url",
        "fixedUri": "http://isc-demo/fhir/StructureDefinition/patient-birthPlace"
      },
      {
        "id": "Extension.value[x]",
        "path": "Extension.value[x]",
        "type": [
          {
            "code": "string"
          }
        ]
      }
    ]
  }
}
```

Les données d'Extension aux ressources Patient ajoutées avec cette Extension ressembleront à ceci.

```
  "extension": [
    {
      "url": "http://isc-demo/fhir/StructureDefinition/patient-birthPlace",
      "valueString": "???"
    }
  ],
```

Ajout de SearchParamter

Ensuite, ajoutez un **SearchParamter** afin de pouvoir rechercher des ressources en utilisant l'entrée Extension que vous venez d'ajouter comme clé, **mais seules les entrées (elements) définies dans le SearchParamter peuvent être recherchées**. Ceci est un peu différent des tables SQL.

Le nom de SearchParamter est défini séparément du nom de l'élément, et certains éléments correspondent au nom de l'élément = nom de SearchParameter, comme le sexe dans la ressource Patient, alors que d'autres ne correspondent pas, comme le nom de l'élément = adresse. country -> nom de SearchParamter = Certains ne correspondent pas aux éléments structurés, comme address-country.

Naturellement, les éléments ajoutés à l'extension ne sont pas des SearchParameters par défaut (car vous ne savez pas ce qui sera inclus), mais les extensions qui osent définir l'extension et définir une politique pour les stocker sont souvent des éléments importants.

Ajoutez ce qui suit au fichier patient.fsh pour créer la définition de SearchParameter

```
Instance: BirthPlaceSearchParameter
```

```
InstanceOf: SearchParameter
Usage: #definition
* url = "http://isc-demo/fhir/SearchParameter/patient-birthPlace"
* version = "0.0.1"
* name = "birthPlace"
* status = #active
* description = "???????????"
* code = #birthPlace
* base = #Patient
* type = #string
* expression = "Patient.extension.where(url='http://isc-
demo/fhir/StructureDefinition/patient-birthPlace').value"
* comparator = #eq
```

Voici la StructureDefinition générée par SearchParameter.
Comme il s'agit d'une définition relativement simple, le mappage avec l'information SUSHI ci-dessus devrait être facile à comprendre.

```
{
  "resourceType": "SearchParameter",
  "id": "BirthPlaceSearchParameter",
  "url": "http://isc-demo/fhir/SearchParameter/patient-birthPlace",
  "version": "0.0.1",
  "name": "birthPlace",
  "status": "active",
  "description": "???????????",
  "code": "birthPlace",
  "base": [
    "Patient"
  ],
  "type": "string",
  "expression": "Patient.extension.where(url='http://isc-
demo/fhir/StructureDefinition/patient-birthPlace').value",
  "comparator": [
    "eq"
  ]
}
```

Les principales composantes de la définition de SearchParameter sont l' **expression** et le **comparateur**.
L'élément **expression** décrit l'expression **FHIRPath** pour le SearchParameter cible. Si vous êtes intéressé par FHIRPath, veuillez vous référer à [cette page officielle](. html).

Définition utilisée cette fois-ci

Patient.extension.where(url='http://isc-demo/fhir/StructureDefinition/patient-birthPlace').value"

Cette expression spécifie Patient.extension dans un ordre hiérarchique selon la structure Json de la ressource Patient, et réduit l'extension avec url=(omitted) par rapport aux multiples extensions qui peuvent exister, et spécifie la valeur de l'extension.

comparator spécifie le type d'expressions de comparaison qui peuvent être utilisées. Pour plus d'informations, voir ici.

Ajout de la définition d'Extension créée dans Patient

Il y a un autre changement important : l'ajout de l'Extension BirthPlace créée dans la ressource Patient. Modifiez la définition de profil MyProfile dans la ressource Patient générée automatiquement à l'origine comme suit : les modifications apportées au paramètre Cardinality de l'élément Name ont été commentées.

ècomparator spécifie le type d'expressions de comparaison qui peuvent être utilisées. Pour plus d'informations, voir ici.
Profile: MyPatient
Parent: Patient
Description: "An example profile of the Patient resource."
//* name 1..* MS
* extension contains BirthPlace named birthPlace 0..1

```
L'Extension nommée "BirthPlace" qui a été ajoutée précédemment est ajoutée dans la re
ssource Patient avec le nom birthPlace dans le paramètre Cardinality 0..1.

# Création de ressources pour des tests en plus de ce qui précède

SUSHI vous permet également de créer des Instances de ressources qui peuvent être uti
lisées à des fins d'illustration ou autres. Vous pouvez également les utiliser à des
fins de test. Vous pouvez également y inclure l'extension que vous venez de définir.
```

Instance: KamiExample
InstanceOf: MyPatient
Description: "Exemples de ressources Patient "
* nom.family = "Yamada"
* extension[BirthPlace].valueString = "Kagoshima"

```
Vous verrez quel type de données a été produit dans le test final.

## Essayons SUSHI !

Le fichier FSH est prêt ! Maintenant, utilisons la commande SUSHI pour générer chaque
 fichier de définition à partir du fichier fsh !
Exécutez la commande **sushi** et elle est réussie si deux Profils (Patient et Extens
ion étendus) et deux Instances (SearchParameter et ressource de modèle) sont générés
comme suit.

```PowerShell
C:\Users\kaminaka\Documents\Work\FHIR\SUSHI\TestProject\MyProfileProject>sushi .
info  Running SUSHI v2.1.1 (implements FHIR Shorthand specification v1.2.0)
info  Arguments:
info    C:\Users\kaminaka\Documents\Work\FHIR\SUSHI\TestProject\MyProfileProject
info  No output path specified. Output to .
info  Using configuration file: C:\Users\kaminaka\Documents\Work\FHIR\SUSHI\TestProje
ct\MyProfileProject\sushi-config.yaml
info  Importing FSH text...
info  Preprocessed 1 documents with 0 aliases.
info  Imported 2 definitions and 2 instances.
info  Checking local cache for hl7.fhir.r4.core#4.0.1...
info  Found hl7.fhir.r4.core#4.0.1 in local cache.
info  Loaded package hl7.fhir.r4.core#4.0.1
(node:27132) Warning: Accessing non-
existent property 'INVALID_ALT_NUMBER' of module exports inside circular dependency
(Use `node --trace-warnings ...` to show where the warning was created)
(node:27132) Warning: Accessing non-
```
```

```
existent property 'INVALID_ALT_NUMBER' of module exports inside circular dependency
info  Converting FSH to FHIR resources...
info  Converted 2 FHIR StructureDefinitions.
info  Converted 2 FHIR instances.
info  Exporting FHIR resources as JSON...
info  Exported 4 FHIR resources as JSON.
info  Assembling Implementation Guide sources...
info  Generated ImplementationGuide-myprofileproject.json
info  Assembled Implementation Guide sources; ready for IG Publisher.

??????????????????????? SUSHI RESULTS ???????????????????????????
? ?????????????????????????????????????????????????????????????? ?
? ?    Profiles   ?  Extensions  ?   Logicals   ?   Resources   ? ?
? ?????????????????????????????????????????????????????????????? ?
? ?       1       ?      1       ?      0       ?       0       ? ?
? ?????????????????????????????????????????????????????????????? ?
? ?????????????????????????????????????????????????????????????? ?
? ?      ValueSets     ?    CodeSystems    ?     Instances      ? ?
? ?????????????????????????????????????????????????????????????? ?
? ?         0          ?         0         ?         2          ? ?
? ?????????????????????????????????????????????????????????????? ?
?                                                                  ?
????????????????????????????????????????????????????????????????
? FSHing for compliments? Super job!     0 Errors      0 Warnings ?
????????????????????????????????????????????????????????????????

C:\Users\kaminaka\Documents\Work\FHIR\SUSHI\TestProject\MyProfileProject>
```

Les artefacts suivants ont été créés dans le dossier fsh-generated\resource.

nom de fichier	contenu
ImplementationGuide-myprofileproject.json	ImplemamtionGuide, qui résume l'ensemble des contenus présentés ici.
StructureDefinition-MyPatient.json	StructureDefinition avec Extension ajoutée au Patient
StructureDefinition-birthPlace.json	StructureDefinition contenant la définition d'Extension birthPlace
SearchParameter-BirthPlaceSearchParameter.json	Fichier de définition de SearchParameter pour le lieu de naissance (birthPlace)
Patient-KamiExample.json	Exemples d'instances de Patient

Importation et test du profil FHIR dans IRIS for Health

Application d'IRIS for Health au référentiel FHIR

Dans l'article précédent, nous avons exécuté updatePublisher pour générer un groupe de fichiers IG. Dans cet article, nous allons voir comment le fichier StructureDefinitino/SearchParameter peut être importé dans le référentiel FHIR d'IRIS for Health et faire l'objet d'une recherche avec le nouveau SearchParameter.

Pour plus d'informations sur l'importation des profils FHIR, etc., veuillez vous référer à l'article de la communauté des développeurs sur les profils FHIR.
Vous pouvez également vous référer à cet article pour plus d'informations sur la façon de construire un référentiel FHIR.

L'importation est ciblée sur les cinq fichiers qui viennent d'être générés.
- StructureDefinition-MyPatient.json
- StructureDefinition-birthPlace.json
- SearchParameter-BirthPlaceSearchParameter.json

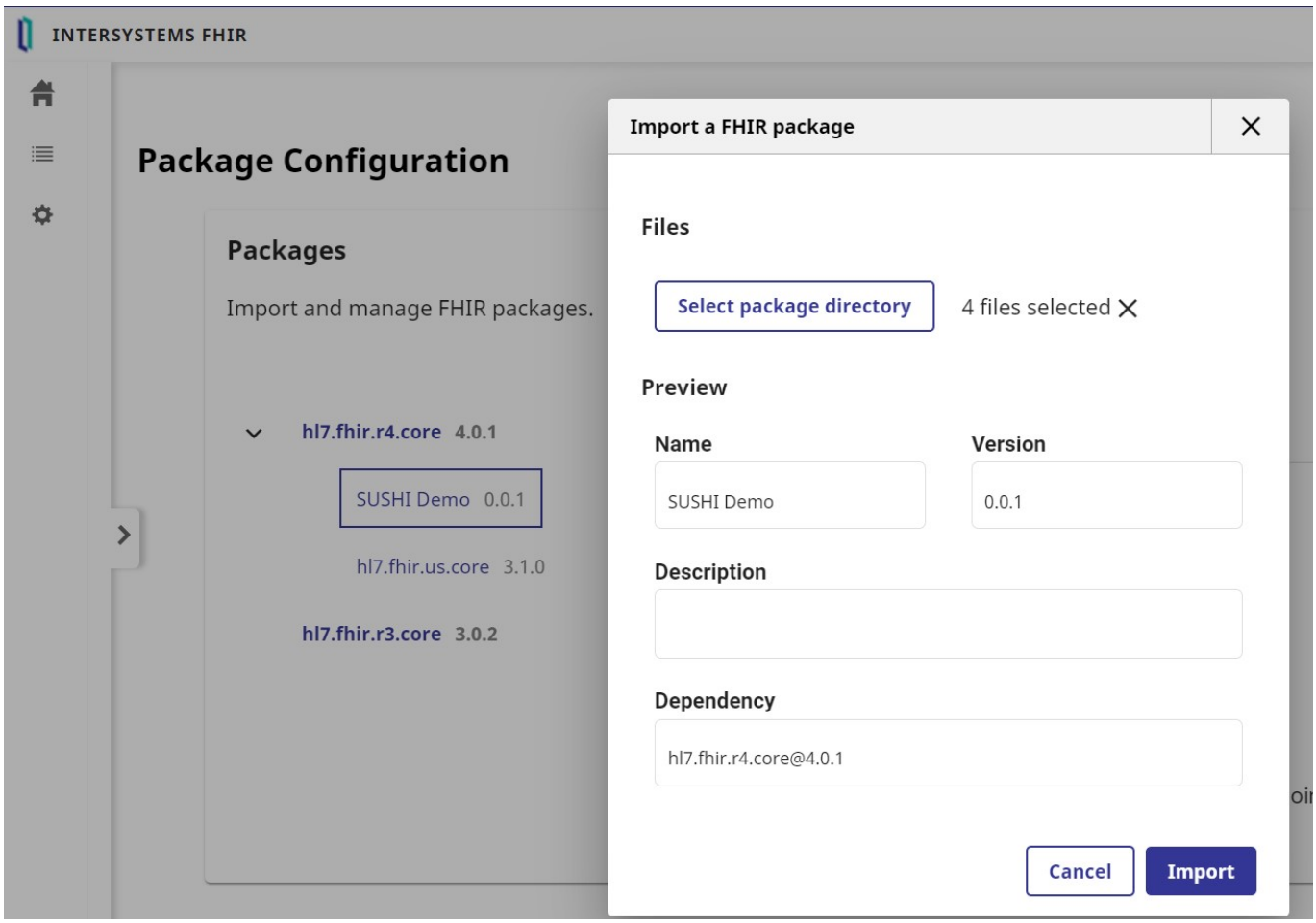
Il y a trois fichiers. Copiez-les dans un autre dossier, et préparez également un fichier package.json pour gérer l'ensemble des informations du paquet.

package.json

```
{
  "name": "SUSHI Demo",
  "title": "SUSHI Demo",
  "version": "0.0.1",
  "author": {
    "name": "ISC"
  },
  "fhirVersions": [
    "4.0.1"
  ],
  "bundleDependencies": false,
  "date": "20201208205547",
  "dependencies": {
    "hl7.fhir.r4.core": "4.0.1"
  },
  "deprecated": false
}
```

Vous pouvez modifier le nom, le titre, l'auteur, la date et d'autres éléments comme vous le souhaitez.
(Remarque : lorsque chaque profil est modifié et réimporté dans IRIS, la version doit être modifiée (augmentée) en conséquence.
(La version actuelle 2021.1 du référentiel FHIR n'a pas de fonction de suppression des profils, il faut donc veiller à ce que le nombre de profils n'augmente pas trop dans l'environnement de production, etc., en vérifiant le bon fonctionnement dans l'environnement de test et en ne les appliquant ensuite qu'un nombre minimum de fois dans l'environnement de production.)

À partir du portail de gestion IRIS, allez Health -> FHIR Configuration -> Package Configuration et sélectionnez le dossier contenant les quatre fichiers ci-dessus dans Import Package, et vous verrez l'écran suivant.



Cliquez sur Import pour terminer l'importation dans IRIS.

Ensuite, créez un nouveau référentiel FHIR sur l'écran de configuration du serveur Server Configuration.
(Vous pouvez également ajouter à un référentiel FHIR existant).

Test de POSTMAN

POSTEZ la ressource de test qui vient d'être générée par SUSHI. À des fins de vérification, il peut être préférable de générer des données qui incluent d'autres valeurs de birthPlace, ou une ressource Patient qui n'inclut pas de birthPlace en premier lieu.

Si birthPlace a été correctement ajouté au SearchParameter dans le référentiel FHIR, la requête GET suivante devrait permettre de récupérer ces informations sur le patient !

```
GET http://localhost:52785/csp/healthshare/sushi/fhir/r4/Patient?birthPlace=Kagoshima
```

Obtenez-vous maintenant les bons résultats ?
Si le nouveau SearchParameter, birthPlace, n'a pas été ajouté correctement, la première réponse à la requête GET contiendra la ressource OperationOutcome suivante qui contient les informations d'erreur suivantes : "Le paramètre birthPlace n'a pas été reconnu. Vérifiez le message de réponse pour ce message.

```
          {
        "resource": {
            "resourceType": "OperationOutcome",
            "issue": [
```

```
                {
                    "severity": "error",
                    "code": "invalid",
                    "diagnostics": "<HSFHIRErr>ParameterNotSupported",
                    "details": {
                        "text": "Unrecognized parameter 'birthPlace'. ???"
                    }
                }
            ]
        },
        "search": {
            "mode": "outcome"
        }
    },
```

Résumé

Vous avez vu le processus de création d'un profil (StructureDefinition/SearchParameter) pour FHIR à l'aide de SUSHI et son importation dans le référentiel FHIR d'IRIS for Health pour étendre ses fonctionnalités.
Dans ce cas, les éléments ajoutés à Extension ont été ajoutés à SearchParameter, mais il est également possible d'ajouter SearchParameter à des éléments qui existent dans la spécification standard FHIR mais qui ne sont pas encore des SearchParameters.

Bien que le développement très flexible de FHIR permette d'étendre les fonctionnalités de cette manière, il est également important de partager des informations sur le type d'extensions réalisées pour assurer l'interopérabilité, c'est-à-dire de créer des guides de mise en œuvre, etc. Comme nous l'avons vu dans les parties 1 et 2 de cette série, SUSHI est un outil open source très unique et puissant qui couvre les deux côtés de la question.

On espère que ces outils seront combinés avec IRIS for Health pour créer une nouvelle solution FHIR.

Le fichier fsh SUSHI utilisé dans cet article et les fichiers modèles StructureDefinition/SearchParameter générés sont disponibles ici.

#FHIR #InterSystems IRIS for Health

URL de la source:https://fr.community.intersystems.com/post/utilisation-de-sushi-pour-la-cr%C3%A9ation-de-profils-fhir-partie-2