Article
Guillaume Rongier · Jan 20, 2023 12m de lecture

Open Exchange

Création de questionnaires FHIR

Intersystems IRIS for Health offre un excellent support pour la norme sectorielle FHIR. Les principales caractéristiques sont :
1. Serveur FHIR
2. Base de données FHIR
3. API REST et ObjectScript pour les opérations CRUD sur les ressources FHIR (patient, questionnaire, vaccins, etc.)

Cet article explique comment utiliser chacune de ces fonctionnalités, et présente un front-end angulaire permettant de créer et d'afficher des ressources FHIR de type Quiz.

Étape 1 - Déploiement de votre serveur FHIR à l'aide d'InterSystems IRIS for Health

Pour créer votre serveur FHIR, il faut ajouter les instructions suivantes dans le fichier iris.script (à partir de : https://openexchange.intersystems.com/package/iris-fhir-template)

```
    zn "HSLIB"
    set namespace="FHIRSERVER"
    Set appKey = "/fhir/r4"
    Set strategyClass = "HS.FHIRServer.Storage.Json.InteractionsStrategy"
    set metadataPackages = $lb("hl7.fhir.r4.core@4.0.1")
    set importdir="/opt/irisapp/src"

    //Install a Foundation namespace and change to it
    Do ##class(HS.HC.Util.Installer).InstallFoundation(namespace)
    zn namespace

    // Install elements that are required for a FHIR-enabled namespace
    Do ##class(HS.FHIRServer.Installer).InstallNamespace()

    // Install an instance of a FHIR Service into the current namespace
    Do ##class(HS.FHIRServer.Installer).InstallInstance(appKey, strategyClass, metada
taPackages)

    // Configure FHIR Service instance to accept unauthenticated requests
    set strategy = ##class(HS.FHIRServer.API.InteractionsStrategy).GetStrategyForEndp
oint(appKey)
    set config = strategy.GetServiceConfigData()
    set config.DebugMode = 4
    do strategy.SaveServiceConfigData(config)

    zw ##class(HS.FHIRServer.Tools.DataLoader).SubmitResourceFiles("/opt/irisapp/fhir
data/", "FHIRServer", appKey)

    do $System.OBJ.LoadDir("/opt/irisapp/src","ck",,1)
```

```
    zn "%SYS"
    Do ##class(Security.Users).UnExpireUserPasswords("*")

    zn "FHIRSERVER"
    zpm "load /opt/irisapp/ -v":1:1

    //zpm "install fhir-portal"
    halt
```

En utilisant la classe utilitaire HS.FHIRServer.Installer, vous pouvez créer votre serveur FHIR.

Étape 2 - Utilisez l'API FHIR REST ou ObjectScript pour lire, mettre à jour, supprimer et trouver des données FHIR

Je préfère utiliser la classe ObjectScript HS.FHIRServer.Service pour faire toutes les opérations CRUD.

Pour obtenir toutes les données FHIR provenant d'un type de ressource (comme le questionnaire) :

```
/// Retreive all the records of questionnaire
ClassMethod GetAllQuestionnaire() As %Status
{

    set tSC = $$$OK
    Set %response.ContentType = ..#CONTENTTYPEJSON
    Set %response.Headers("Access-Control-Allow-Origin")="*"

    Try {
        set fhirService = ##class(HS.FHIRServer.Service).EnsureInstance(..#URL)
        set request = ##class(HS.FHIRServer.API.Data.Request).%New()
        set request.RequestPath = "/Questionnaire/"
        set request.RequestMethod = "GET"
        do fhirService.DispatchRequest(request, .pResponse)
        set json = pResponse.Json
        set resp = []
        set iter = json.entry.%GetIterator()
        while iter.%GetNext(.key, .value) {
          do resp.%Push(value.resource)
        }

        write resp.%ToJSON()
    } Catch Err {
        set tSC = 1
        set message = {}
        set message.type= "ERROR"
        set message.details = "Error on get all questionnairies"
    }

    Quit tSC
}
```

Pour obtenir un élément de données spécifique du référentiel de données FHIR (comme une occurrence de questionnaire) :

```
/// Retreive a questionnaire by id
```

```
ClassMethod GetQuestionnaire(id As %String) As %Status
{

    set tSC = $$$OK
    Set %response.ContentType = ..#CONTENTTYPEJSON
    Set %response.Headers("Access-Control-Allow-Origin")="*"

    Try {
        set fhirService = ##class(HS.FHIRServer.Service).EnsureInstance(..#URL)
        set request = ##class(HS.FHIRServer.API.Data.Request).%New()
        set request.RequestPath = "/Questionnaire/"_id
        set request.RequestMethod = "GET"
        do fhirService.DispatchRequest(request, .pResponse)
        write pResponse.Json.%ToJSON()
    } Catch Err {
        set tSC = 1
        set message = {}
        set message.type= "ERROR"
        set message.details = "Error on get the questionnaire"
    }

    Quit tSC
}
```

Pour créer une nouvelle occurrence de ressource FHIR (comme un nouveau questionnaire) :

```
/// Create questionnaire
ClassMethod CreateQuestionnaire() As %Status
{
  set tSC = $$$OK
  Set %response.ContentType = ..#CONTENTTYPEJSON
  Set %response.Headers("Access-Control-Allow-Origin")="*"

  Try {
    set fhirService = ##class(HS.FHIRServer.Service).EnsureInstance(..#URL)
    set request = ##class(HS.FHIRServer.API.Data.Request).%New()
    set request.RequestPath = "/Questionnaire/"
    set request.RequestMethod = "POST"
    set data = {}.%FromJSON(%request.Content)
    set data.resourceType = "Questionnaire"
    set request.Json = data
    do fhirService.DispatchRequest(request, .response)
    write response.Json.%ToJSON()
  } Catch Err {
    set tSC = 1
    set message = {}
    set message.type= "ERROR"
    set message.details = "Error on create questionnaire"
  }

  Return tSC
}
```

Pour mettre à jour une ressource FHIR (comme un questionnaire) :

```
/// Update a questionnaire
ClassMethod UpdateQuestionnaire(id As %String) As %Status
{
  set tSC = $$$OK
```

```
  Set %response.ContentType = ..#CONTENTTYPEJSON
  Set %response.Headers("Access-Control-Allow-Origin")="*"

  Try {
    set fhirService = ##class(HS.FHIRServer.Service).EnsureInstance(..#URL)
    set request = ##class(HS.FHIRServer.API.Data.Request).%New()
    set request.RequestPath = "/Questionnaire/"_id
    set request.RequestMethod = "PUT"
    set data = {}.%FromJSON(%request.Content)
    set data.resourceType = "Questionnaire"
    set request.Json = data
    do fhirService.DispatchRequest(request, .response)
    write response.Json.%ToJSON()
  }Catch Err {
    set tSC = 1
    set message = {}
    set message.type= "ERROR"
    set message.details = "Error on update questionnaire"
  }
  Return tSC
}
```

Pour supprimer un occurrence de ressource FHIR (comme un questionnaire) :

```
/// Delete a questionnaire by id
ClassMethod DeleteQuestionnaire(id As %String) As %Status
{

    set tSC = $$$OK
    Set %response.ContentType = ..#CONTENTTYPEJSON
    Set %response.Headers("Access-Control-Allow-Origin")="*"

    Try {
        set fhirService = ##class(HS.FHIRServer.Service).EnsureInstance(..#URL)
        set request = ##class(HS.FHIRServer.API.Data.Request).%New()
        set request.RequestPath = "/Questionnaire/"_id
        set request.RequestMethod = "DELETE"
        do fhirService.DispatchRequest(request, .pResponse)
    } Catch Err {
        set tSC = 1
        set message = {}
        set message.type= "ERROR"
        set message.details = "Error on delete the questionnaire"
    }

    Quit tSC
}
```

Comme vous pouvez le voir, pour créer, il faut utiliser POST, pour mettre à jour, il faut utiliser PUT, pour supprimer, il faut utiliser DELETE et pour lancer une requête, il faut utiliser le verbe GET.

Étape 3 - Créez un client en Angular pour utiliser votre application de serveur FHIR.

J'ai créé une application angulaire en utilisant PrimeNG et en installant le paquet npm install --save @types/fhir. Ce

paquet a tous les types FHIR mappé à TypeScript.

Classe de contrôleur en Angular :

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Period, Questionnaire } from 'fhir/r4';
import { ConfirmationService, MessageService, SelectItem } from 'primeng/api';
import { QuestionnaireService } from './questionnaireservice';

const QUESTIONNAIREID = 'questionnaireId';

@Component({
    selector: 'app-questionnaire',
    templateUrl: './questionnaire.component.html',
    providers: [MessageService, ConfirmationService],
    styleUrls: ['./questionnaire.component.css'],
    encapsulation: ViewEncapsulation.None
})
export class QuestionnaireComponent implements OnInit {

    public questionnaire: Questionnaire;
    public questionnairies: Questionnaire[];
    public selectedQuestionnaire: Questionnaire;
    public questionnaireId: string;
    public sub: any;
    public publicationStatusList: SelectItem[];

    constructor(
        private questionnaireService: QuestionnaireService,
        private router: Router,
        private route: ActivatedRoute,
        private confirmationService: ConfirmationService,
        private messageService: MessageService){
            this.publicationStatusList = [
                {label: 'Draft', value: 'draft'},
                {label: 'Active', value: 'active'},
                {label: 'Retired', value: 'retired'},
                {label: 'Unknown', value: 'unknown'}
            ]
        }

    ngOnInit() {
        this.reset();
        this.listQuestionnaires();
        this.sub = this.route.params.subscribe(params => {

            this.questionnaireId = String(+params[QUESTIONNAIREID]);

            if (!Number.isNaN(this.questionnaireId)) {
                this.loadQuestionnaire(this.questionnaireId);
            }
        });
    }

    private loadQuestionnaire(questionnaireId) {
        this.questionnaireService.load(questionnaireId).subscribe(response => {
            this.questionnaire = response;
```

```
            this.selectedQuestionnaire = this.questionnaire;
            if(!response.effectivePeriod) {
                this.questionnaire.effectivePeriod = {};
            }
        }, error => {
            console.log(error);
            this.messageService.add({ severity: 'error', summary: 'Error', detail: 'E
rror on load questionnaire.' });
        });
    }


    public loadQuestions() {
        if(this.questionnaire && this.questionnaire.id) {
            this.router.navigate(['/question', this.questionnaire.id]);
        } else {
            this.messageService.add({ severity: 'warn', summary: 'Warning', detail: '
Choose a questionnaire.' });
        }
    }

    private listQuestionnaires() {
        this.questionnaireService.list().subscribe(response => {
            this.questionnairies = response;
            this.reset();
        }, error => {
            console.log(error);
            this.messageService.add({ severity: 'error', summary: 'Error', detail: 'E
rror on load the questionnaries.' });
        });
    }

    public onChangeQuestionnaire() {
        if (this.selectedQuestionnaire && !this.selectedQuestionnaire.id) {
            this.messageService.add({ severity: 'warn', summary: 'Warning', detail: '
Select a questionnaire.' });
        } else {
            if(this.selectedQuestionnaire && this.selectedQuestionnaire.id) {
                this.loadQuestionnaire(this.selectedQuestionnaire.id);
            }
        }
    }

    public reset() {
        this.questionnaire = {};
        this.questionnaire.effectivePeriod = {};
    }


    public save() {

        if(this.questionnaire.id && this.questionnaire.id != "") {
            this.questionnaireService.update(this.questionnaire).subscribe(
                (resp) => {
                    this.messageService.add({
                        severity: 'success',
                        summary: 'Success', detail: 'Questionnaire saved.'
                    });
                    this.listQuestionnaires()
```

```
                    this.loadQuestionnaire(this.questionnaire.id);
                },
                error => {
                    console.log(error);
                    this.messageService.add({
                        severity: 'error',
                        summary: 'Error', detail: 'Error on save the questionnaire.'
                    });
                }
            );
        } else {
            this.questionnaireService.save(this.questionnaire).subscribe(
                (resp) => {
                    this.messageService.add({
                        severity: 'success',
                        summary: 'Success', detail: 'Questionnaire saved.'
                    });
                    this.listQuestionnaires()
                    this.loadQuestionnaire(resp.id);
                },
                error => {
                    console.log(error);
                    this.messageService.add({
                        severity: 'error',
                        summary: 'Error', detail: 'Error on save the questionnaire.'
                    });
                }
            );
        }

    }

    public delete(id: string) {

        if (!this.questionnaire || !this.questionnaire.id) {
            this.messageService.add({ severity: 'warn', summary: 'Warning', detail: '
Select a questionnaire.' });
        } else {
            this.confirmationService.confirm({
                message: 'Do you confirm?',
                accept: () => {
                    this.questionnaireService.delete(id).subscribe(
                        () => {
                            this.messageService.add({ severity: 'success', summary: '
Success', detail: 'Questionnaire deleted.' });
                            this.listQuestionnaires();
                            this.reset();
                        },
                        error => {
                            console.log(error);
                            this.messageService.add({ severity: 'error', summary: 'Er
ror', detail: 'Error on delete questionnaire.' });
                        }
                    );
                }
            });
        }
    }
```

```
}
```

Fichier HTML Angular

```
                Name

Title

Date

Status

Publisher

Start Period

End Period

Description
```

Classe de Service Angular

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
```

```
import { Observable } from 'rxjs';
import { environment } from 'src/environments/environment';
import { take } from 'rxjs/operators';
import { Questionnaire } from 'fhir/r4';

@Injectable({
  providedIn: 'root'
})
export class QuestionnaireService {

  private url = environment.host2 + 'questionnaire';

  constructor(private http: HttpClient) { }

  public save(Questionnaire: Questionnaire): Observable {
    return this.http.post(this.url, Questionnaire).pipe(take(1));
  }

  public update(Questionnaire: Questionnaire): Observable {
    return this.http.put(`${this.url}/${Questionnaire.id}`, Questionnaire).pipe(take(
1));
  }

  public load(id: string): Observable {
    return this.http.get(`${this.url}/${id}`).pipe(take(1));
  }

  public delete(id: string): Observable {
    return this.http.delete(`${this.url}/${id}`).pipe(take(1));
  }

  public list(): Observable {
    return this.http.get(this.url).pipe(take(1));
  }

}
```

Step 4 - Application in action

1. Aller à l'application https://openexchange.intersystems.com/package/FHIR-Questionnaires.

2. Clone/git dépose le dépôt dans n'importe quel répertoire local.

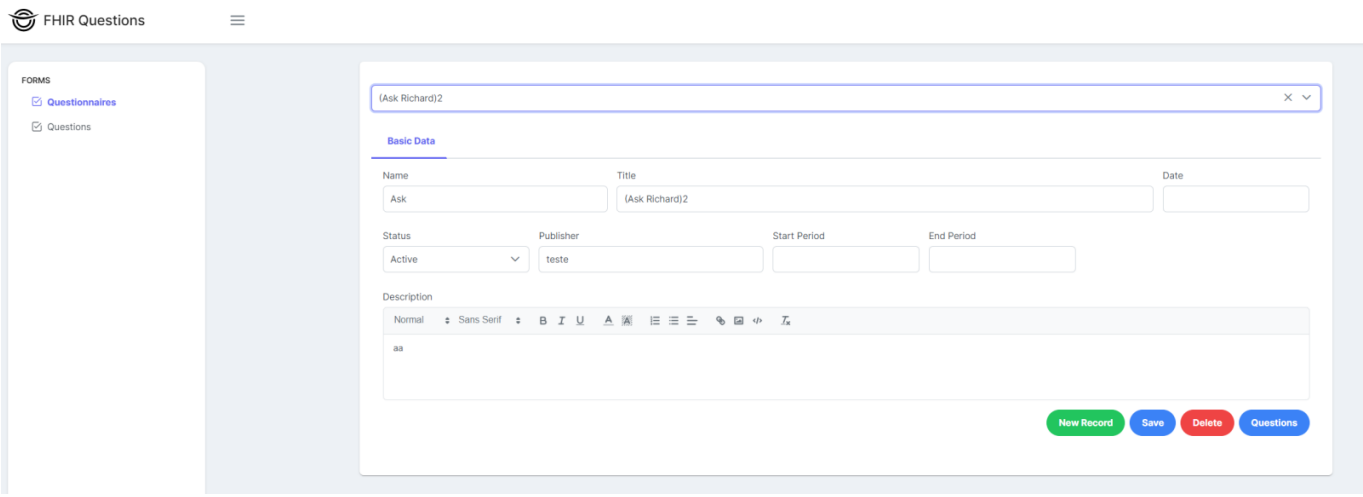
```
$ git clone https://github.com/yurimarx/fhir-questions.git
```

3. Ouvrir le terminal dans ce répertoire et exécutez :

```
$ docker-compose up -d
```

4. Ouvrir l'application web : http://localhost:52773/fhirquestions/index.html

Voici quelques illustrations :



#Angular #FHIR #REST API #InterSystems IRIS for Health
Voir l'application sur InterSystems Open Exchange

URL de la source:https://fr.community.intersystems.com/post/cr%C3%A9ation-de-questionnaires-fhir