plone.app.fhirfield Documentation

Release 1.0.0rc1

Md Nazrul Islam

Oct 25, 2018

Contents

Introduction

FHIR Fast Heathcare Interoperability Resource

CHAPTER 2

FHIR Search Query

This product has a query builder helper, that is actually transforming fhir search params into Plone's search compatible params so that PortalCatalog tool understand what to do. Each of FHIR Indexes provided by this product, by default using this query builder but it is possible to provide prepared query.

2.1 Query Builder

plone.app.fhir.field.helpers.build_elasticsearch_query takes only HL7 fhir standard search parameters and transforms to elasticsearch compatible query that is executing through plone catalog search. However belows are lists of standard parameters those are supported by this *Query Builder* (more to continue adding, until full supports are completed)

2.2 Integrate in your REST API service

This product has been got all battery included to be integrated with plone.restapi for becoming HL7 FHIR standard RESTful API server which would provide search service as defined here.

Example RESTful service could be found here

Parameter Name	Example
_id	
_lastUpdated	
_profile	
birthdate	
gender	
patient	
status	
owner	
subject	

Table 1 – continued from previous page

Parameter Name	Example
effective	
url	
version	
authored	
identifier	
based-on	
part-of	
price-override	
	quantity |
factor-override	
	length |
	code |
	medication |
	address |
	address-city |
address-country	
address-postalcode	
	address-state |
	address-use |
	telecom |
	email |
	phone |
	name |
	family |
given	
code-value-concept	
code-value-date	
	code-value-quantity |
code-value-string	

CHAPTER 3

Migration

3.1 1.0.0rc3 to 1.0.0rc4

1. Keep backup of existing Data.fs (nice to have)
2. Go to plone control panel's Addon settings
3. Uninstall and Install again *plone.app.fhirfield*

3.2 1.0.0b6 to 1.0.0rc3

1. Keep backup of existing Data.fs (important)
2. Just *Clear and Rebuild* existing catalogs. Go to site management -> portal_catalog -> Advanced Tab and click the *Clear and Rebuild* button. Caution: this activity could take longer time.

3.3 1.0.0bx to 1.0.0b6

1. You will need to remove all the indexes who are based on *Fhir*******Index*. Go to ZMI to portal catalog tool. *{site url}/portal_catalog/manage_catalogIndexes*.
2. Update the *plone.app.fhirfield* version and run the buildout.
3. List out any products those defined (profile/catalog.xml) indexes based on *Fhir*******Index* as meta type. Reinstall them all.
4. From ZMI, portal_catalog tool, *{site url}/portal_catalog/manage_catalogAdvanced* `Clear and Rebuild`

Changelog

4.1 1.0.0rc4 (2018-10-25)

Breaking

- Drop support for Plone 4.xx (sorry).

Improvements

- Issue#10 JSON Viewer added in display mode.

- Issue#18 *api* module to make available for all publicly usable methods, functions, classes.

- Issue#17 Add suports for duplicate param names into query string. It is now possible to provide multiple condition for same param type. For example *?_lastUpdated=gt2015-10-15T06:31:18+00:00&_lastUpdated=lt2018-01-15T06:31:18+00:00*

- Issue#10 Add support for *Composite* type FHIR search param.

- Issue#13 Add support for *Address* and *ContactPoint* mapping. It opens up many search params.

- Mappings are more enriched, so many missing mappings are added, like *valueString*, *valueQuantity* and so on.[nazrulworld]

- Issue#12 Full support for *code* search param type has been added, it also opens up for other search parameters (y).

- Issue#15 support for *Humanane* mapping in search.

4.2 1.0.0rc3 (2018-09-22)

Improvements

- Issue: 6 A major improvement has been done, now very slim version (*id*, *resourceType*, *meta*) of FHIR json resource has been indexed in ZCatalog (zope index) version, however previously whole fhir resource was stored as a result now huge storage savings, perhaps improves indexing performance. [nazrulworld]

4.3 1.0.0rc2 (2018-08-29)

Bugfixes

- Issue 5: FHIR search's modifier 'missing' is not working for nested mapping

4.4 1.0.0rc1 (2018-08-27)

Newfeatures

- Identifier search parameter is active now (both array of identifier and single object identifier).

- Array of Reference query support (for example *basedOn* (list of reference)) is active now. Although normal object reference has already been supported.

- All available mappings for searchable resources are generated.

- FHIR search sort feature is available!

- *fhirfield.es.index.mapping.nested_fields.limit*,   *fhirfield.es.index.mapping.depth.limit*   and   *fhir-field.es.index.mapping.total_fields.limit* registry records are available to setup ES mapping

- *URI* and *Number* type parameter based search are fully available.

- **'resourceType' filter is automatically injected into every generated query.** Query Builder knows about which resourceType should be.

Breaking Changes

- *plone.app.fhirfield* have to install, as some registry records (settings) for elasticsearch mapping have been introduced.

- Any deprecated FHIR Field Indexes other than *FhirFieldIndex* (*FhirOrganizationIndex* and so on) are removed

4.5 1.0.0b7 (2018-08-10)

- Media search available.

- *plone.app.fhirfield.SearchQueryError* exception class available, it would be used to catch any fhir query buiding errors. [nazrulworld]

4.6 1.0.0b6 (2018-08-04)

- Fix: minor type mistake on non existing method called.

- Migration guide has been added. [nazrulworld]

4.7 1.0.0b5 (2018-08-03)

Newfeatures

- *FhirFieldIndex* Catalog Index has been refactored. Now this class is capable to handle all the FHIR resources. That's why other PluginIndexes related to FhirField have been deprecated.

- New ZCatalog (plone index) index naming convention has been introduced. Any index name for FhirFieldIndex must have fhir resource type name as prefix. for example: *task_index*

4.8 1.0.0b4 (2018-08-01)

- Must Update (fix): Important updates made on mapping, reference field mapping was not working if value contains with */,* now made it tokenize by indecating index is *not_analyzed*
- *_profile* search parameter is now available. [nazrulworld]

4.9 1.0.0b3 (2018-07-30)

- Mapping improvment for *FhirQuestionnaireResponseIndex*, *FhirObservationIndex*, *FhirProcedureRequestIndex*, *FhirTaskIndex*, *FhirDeviceRequestIndex*

4.10 1.0.0b2 (2018-07-29)

New Features:

- supports for elasticsearch has been added. Now many basic fhir search are possible to be queried.
- upto 22 FHIR fields indexes (*FhirActivityDefinitionIndex*, *FhirAppointmentIndex*, *FhirCarePlanIndex*, *FhirDeviceIndex*, *FhirDeviceRequestIndex*, *FhirHealthcareServiceIndex*, *FhirMedicationAdministrationIndex*, *FhirMedicationDispenseIndex*, *FhirMedicationRequestIndex*, *FhirMedicationStatementIndex*, *FhirObservationIndex*, *FhirOrganizationIndex*, *FhirPatientIndex*, *FhirPlanDefinitionIndex*, *FhirPractitionerIndex*, *FhirProcedureRequestIndex*, *FhirQuestionnaireIndex*, *FhirQuestionnaireResponseIndex*, *FhirRelatedPersonIndex*, *FhirTaskIndex*, *FhirValueSetIndex*)
- Mappings for all available fhir indexes are created.
- *elasticsearch* option is now available for setup.py

4.11 1.0.0b1 (2018-03-17)

- first beta version has been released.

4.12 1.0.0a10 (2018-03-12)

- fix(bug) Issue-3: *resource_type* constraint don't raise exception from validator.

4.13 1.0.0a9 (2018-03-08)

- There is no restriction/limit over fhir resources, all available models are supported.

4.14 1.0.0a8 (2018-01-22)

- fix(bug) Issue-: Empty string value raise json validation error #2:https://github.com/nazrulworld/plone.app.fhirfield/issues/2

4.15 1.0.0a7 (2018-01-21)

- fix(bug) Issue-1: _RuntimeError: maximum recursion depth exceeded while calling a Python object at form view. #1:https://github.com/nazrulworld/plone.app.fhirfield/issues/1

4.16 1.0.0a6 (2018-01-14)

- missing *HealthcareService* fhir model is added as supported model.

4.17 1.0.0a5 (2018-01-14)

- *Person* fhir model added in whitelist.

4.18 1.0.0a4 (2018-01-14)

- IFhirResource.model_interface field type changed to *DottedName* from *InterfaceField*.

4.19 1.0.0a3 (2017-12-22)

- FHIR Patch support added. Now patching fhir resource is more easy to manage.
- plone.supermodel support is confirmed.[nazrulworld]
- plone.rfc822 marshaler field support.[nazrulworld]

4.20 1.0.0a2 (2017-12-10)

- *FhirResourceWidget* is made working state. From now it is possible to adapt FhirResourceWidget' with z3c.form [nazrulworld]

4.21 1.0.0a1 (2017-12-05)

- Initial release. [nazrulworld]

Contribute

5.1 Using the development buildout

Create a virtualenv in the package:

```
$ virtualenv --clear .
```

Install requirements with pip:

```
$ ./bin/pip install -r requirements.txt
```

Run buildout:

```
$ ./bin/buildout
```

Start Plone in foreground:

> $ ./bin/instance fg

Contributors

- Md Nazrul Islam (Author), email2nazrul@gmail.com

License

Credits

8.1 Copyright (third party)

Plone

8.2 Ideas and Concepts

Documentation

Field

Background (plone.app.fhirfield)

FHIR (Fast Healthcare Interoperability Resources) is the industry standard for Healthcare system. Our intend to implement FHIR based system using Plone! plone.app.fhirfield will make life easier to create, manage content for FHIR resources as well search facilities for any FHIR Resources.

9.1 How It Works

This field is working as like other zope.schema field, just add and use it. You will feed the field value as either json string or python dict of FHIR resources through web form or any restapi client. This field has built-in FHIR resource validator and parser.

Example:

```python
from plone.app.fhirfield import FhirResource
from plone.supermodel import model


class IMyContent(model.Schema):

    <resource_type>_resource = FhirResource(
        title=u'your title',
        desciption=u'your desciption',
        resource_type='any fhir resource type[optional]'
    )
```

The field's value is the instance of a specilized class *FhirResourceValue* inside the context, which is kind of proxy class of fhirclient model with additional methods and attributes.

Make field indexable

A specilized Catalog PluginIndexes is named `FhirFieldIndex` is available, you will use it as like other catalog indexes. However importantly you have to maintain a strict convention (index name must be started with any valid fhir resource name (no matter uppercase or lowercase) and should _(underscore) be used as separator, i.e task_index(<resource>_<any name>))

Example:

```
<?xml version="1.0"?>
<object name="portal_catalog" meta_type="Plone Catalog Tool">
    <index name="organization_resource" meta_type="FhirFieldIndex">
        <indexed_attr value="organization_resource"/>
    </index>
</object>
```

9.2 Features

- Plone restapi support

- Widget: z3cform support

- plone.supermodel support

- plone.rfc822 marshaler field support

- 100% FHIR search compliance catalog search.

Available Field's Options

This field has got all standard options (i.e *title*, *required*, *desciption* and so on) with additionally options for the purpose of either validation or constraint those are related to FHIR.

resource_type Required: No

>   Default: None
>
>   Type: String
>
>   The name of FHIR resource can be used as constraint, meaning that we can restricted only accept certain resource. If FHIR json is provided that contains other resource type, would raise validation error. Example: *FhirResource(. . . .,resource_type='Patient')*

model Required: No

>   Default: None
>
>   Type: String + full python path (dotted) of the model class.
>
>   Like *resource_type* option, it can be used as constraint, however additionally this option enable us to use custom model class other than fhirclient's model. Example: *FhirResource(. . . .,model='fhirclient.models.patient.Patient')*

model_interface Required: No

>   Default: None
>
>   Type: String + full python path (dotted) of the model class.
>
>   Unlike *model* option, this option has more versatile goal although you can use it for single resource restriction. The advanced usecase like, the field could accept muiltiple resources types those model class implements the provided interface. For example you made a interface called *IMedicalService* and (*Organization*, *Patient*, *Practitioner*) models those are implementing *IMedicalService*. So when you provides this option value, actually three types of fhir json can now be accepted by this field. Example: *FhirResource(. . . .,model='plone.app.interfaces.IFhirResourceModel')*

Field's Value API

Field's value is a specilized class *plone.app.fhirfield.value.FhirResourceValue* which has reponsibilty to act as proxy of fhirclient model's class. This class provides some powerful methods.

FhirResourceValue::as_json

> Originally this method is derived from fhirclient base model, you will always have to use this method during negotiation (although our serializer doing that for you automatically). This method not takes any argument, it returns FHIR json representation of resource.

FhirResourceValue::patch

> If you are familar with FHIRPath Patch, this method one of the strongest weapon of this class. Patch applying on any FHIR resources is noting but so easy. This method takes one mandatory argument *patch_data* and that value should be list of patch items (jsonpatch).

> Example:

```python
from plone.app.fhirfield import FhirResource
from plone.supermodel import model


class ITask(model.Schema):

    resource = FhirResource(
        title=u'your title',
        desciption=u'your desciption',
        resource_type='Task'
    )

patch_data = [
  {'op': 'replace', 'path': '/source/display', 'value': 'Patched display'},
  {'op': 'replace', 'path': '/status', 'value': 'Reopen'}
]
task_content.resource.patch(patch_data)
```

FhirResourceValue::stringify

This method returns string representation of fhir resource json value. Normally *as_json* returns python's dict type data. This method takes optional *prettify* argument, by setting this argument True, method will return human/print friendly representation.

FhirResourceValue::foreground_origin

There may some situation come, where you will need just pure instance of fhir model, this method serves that purpose. This method returns current fhir resource model's instance.

Example:

```python
from fhirclient.models.task import Task
from plone.app.fhirfield import FhirResource
from plone.supermodel import model


class ITask(model.Schema):

    resource = FhirResource(
        title=u'your title',
        desciption=u'your desciption',
        resource_type='Task'
    )

task = task_content.resource.foreground_origin()
assert isinstance(task, Task)
```

Helper API

This package provides some useful functions those could be usable in your codebase.

resource_type_str_to_fhir_model

> This function return appropriate fhirclient model class based on provided *resource type*. On wrong resource type *zope.interface.Invalid* exception is raisen.
>
> Example:

```
>>> from plone.app.fhirfield.helpers import resource_type_str_to_fhir_model
>>> task_model_class = resource_type_str_to_fhir_model('Task')
```

elasticsearch setup

If your intent to use elasticsearch based indexing and query, this section for you! you can find more details here

13.1 server setup

server version is restricted to *2.4.x*, means we cannot use latest version of elasticsearch. i.e 5.6.x

- Download from here and install according to documentation.
- For development you could use docker container. The Makefile is available, *~$ make run-es*

13.2 collective.elasticsearch setup

Full configuration guide could be found here. Simple steps are described bellow.

1. **create catalog/indexes**: First you will need add indexes for each fhirfield used in your project. each resource type has it's own Meta Index. example is here

2. Install *collective.elasticsearch* addon from plone control panel.

3. Convert your Indexes to elasticsearch. Go To *{portal url}/@@elastic-controlpanel*

4. In the settings form's *Indexes for which all searches are done through ElasticSearch* section add your all indexes those you mentioned into catalog.xml file, also add *portal_type*

5. Now save and again *Convert Catalog*.

Installation

Install plone.app.fhirfield by adding it to your buildout:

```
[buildout]

...

eggs =
    plone.app.fhirfield [elasticsearch]
```

and then running `bin/buildout`. Go to plone control and install `plone.app.fhirfield` or If you are creating an addon that depends on this product, you may add `<dependency>profile-plone.app.fhirfield:default</dependency>` in `metadata.xml` at profiles.

14.1 configuration

This product provides three plone registry based records `fhirfield.es.index.mapping.nested_fields.limit`, `fhirfield.es.index.mapping.depth.limit`, `fhirfield.es.index.mapping.total_fields.limit`. Those are related to ElasticSearch index mapping setup, if you aware about it, then you have option to modify from plone control panel (Registry).

Links

Code repository:

> https://github.com/nazrulworld/plone.app.fhirfield

Continuous Integration:

> https://travis-ci.org/nazrulworld/plone.app.fhirfield

Issue Tracker:

> https://github.com/nazrulworld/plone.app.fhirfield/issues

CHAPTER 16

License

The project is licensed under the GPLv2.

CHAPTER 17

Indices and tables

- genindex
- modindex
- search