collective.fhirpath

Release 0.6.0

Md Nazrul Islam (nazrulworld)

Sep 09, 2020

CONTENTS:

CHAPTER

ONE

BACKGROUND (COLLECTIVE.FHIRPATH)

fhirpath implementation in Plone, essential battery included, ready to use.

1.1 Installation

Install collective.fhirpath by adding it to your buildout:

```
[buildout]
...
eggs +=
    collective.fhirpath
```

and then running `bin/buildout`

From Plone controlpanel in the addon settings, install `collective.fhirpath`.

1.2 How It Works

``FhirResource`` the fhirfield

Make sure this specialized field is used properly according to plone.app.fhirfield documentation.

Make field indexable

A specilized Catalog PluginIndexes is named `FhirFieldIndex` is available, you will use it as like other catalog indexes.

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>
```

Elasticsearch settings

Make sure elasticsearch has been configured accourding to collective.elasticsearch docs.

1.2.1 Usages

FHIR Search::

```
>>> from fhirpath.interfaces import IElasticsearchEngineFactory
>>> from fhirpath.interfaces import IFhirSearch
>>> from fhirpath.interfaces import ISearchContextFactory
>>> from plone import api
>>> from collective.elasticsearch.es import ElasticSearchCatalog
>>> from zope.component import queryMultiAdapter
```

```
>>> es_catalog = ElasticSearchCatalog(api.portal.get_tool("portal_catalog"))
>>> factory = queryMultiAdapter(
....        (es_catalog,), IElasticsearchEngineFactory
.... )
>>> engine = factory(fhir_release="STU3")
>>> search_context = queryMultiAdapter((engine,), ISearchContextFactory)(
.... resource_type, unrestricted=False)
>>> search_factory = queryMultiAdapter((search_context,), IFhirSearch)
```

```
>>> params = (
....        ("_profile", "http://hl7.org/fhir/Organization"),
....        ("identifier", "urn:oid:2.16.528.1|91654"),
....        ("type", "http://hl7.org/fhir/organization-type|prov"),
....        ("address-postalcode", "9100 AA"),
....        ("address", "Den Burg"),
....    )
>>> bundle = search_factory(params)
>>> len(bundle.entry)
2
>>> # with query string.
>>> # query_string = self.request["QUERY_STRING]
>>> query_string = "_profile=http://hl7.org/fhir/Organization&
↪identifier=urn:oid:2.16.528.1|91654&type=http://hl7.org/fhir/organization-
↪type|prov&address-postalcode=9100+AA"
>>> bundle = search_factory(query_string=query_string)
>>> len(bundle.entry)
2
```

ZCatlog FHIR Search::

```
>>> from collective.fhirpath.interfaces import IZCatalogFhirSearch
>>> zcatalog_factory = queryMultiAdapter((search_context,), IZCatalogFhirSearch)
```

```
>>> # with query string.
>>> # query_string = self.request["QUERY_STRING]
>>> query_string = "_profile=http://hl7.org/fhir/Organization&
↪identifier=urn:oid:2.16.528.1|91654&type=http://hl7.org/fhir/organization-
↪type|prov&address-postalcode=9100+AA"
>>> brains = zcatalog_factory(query_string=query_string)
>>> len(brains)
2
```

FHIR Query::

```
>>> from fhirpath.interfaces import IElasticsearchEngineFactory
>>> from fhirpath.interfaces import IFhirSearch
```

(continues on next page)

(continued from previous page)

```
>>> from fhirpath.interfaces import ISearchContextFactory
>>> from plone import api
>>> from collective.elasticsearch.es import ElasticSearchCatalog
>>> from zope.component import queryMultiAdapter
>>> from fhirpath.query import Q_
>>> from fhirpath.fql import T_
>>> from fhirpath.fql import sort_
>>> from fhirpath.enums import SortOrderType
```

```
>>> es_catalog = ElasticSearchCatalog(api.portal.get_tool("portal_catalog"))
>>> factory = queryMultiAdapter(
....         (es_catalog,), IElasticsearchEngineFactory
.... )
>>> engine = factory(fhir_release="STU3")
>>> query_builder = Q_(resource="Organization", engine=engine)
....     query_builder = query_builder.where(
....         T_("Organization.meta.profile", "http://hl7.org/fhir/Organization")
....     ).sort(sort_("Organization.meta.lastUpdated", SortOrderType.DESC))
```

```
>>> result = query_builder(async_result=False, unrestricted=True).fetchall()
>>> result.header.total
2
>>> query_result = query_builder(async_result=False, unrestricted=True)
>>> for resource in query_result:
....         count += 1
....         assert resource.__class__.__name__ == "Organization"
```

```
>>> query_builder = Q_(resource="Organization", engine=engine)
>>> query_builder = query_builder.where(T_("Organization.id", "f001"))
>>> result_query = query_builder(async_result=False, unrestricted=True)
>>> resource = result_query.single()
>>> resource is not None
True
```

```
>>> query_builder = Q_(resource="Organization", engine=engine)
>>> query_builder = query_builder.where(
....         T_("Organization.meta.profile", "http://hl7.org/fhir/Organization")
....     )
>>> result_query = builder(async_result=False, unrestricted=True)
>>> result = result_query.first()
>>> isinstance(result, result_query._query.get_from()[0][1])
True
```

1.3 Use `FHIRModelServiceMixin`

For better performance optimization, you should use `FHIRModelServiceMixin` to response `FHIRModel`, `FhirFieldValue` object efficiently.

Example 1:

```
>>> from plone.restapi.services import Service
>>> from collective.fhirpath.utils import FHIRModelServiceMixin
>>> class MyFHIRGetService(FHIRModelServiceMixin, Service):
```

(continues on next page)

(continued from previous page)

```
....     """ """
....     def reply(self):
....         # do return bellow's types of data
....         # could be ``dict`` type data
....         # could be instance of ``FHIRAbstractModel`` derrived class.
....         # could be instance of ``plone.app.fhirfield.FhirResourceValue`` derrived␣
↪class.
....         # or self.reply_no_content()
```

1.4 configuration

This product provides three plone registry based records `fhirpath.es.index.mapping.nested_fields.limit`, `fhirpath.es.index.mapping.depth.limit`, `fhirpath.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).

1.5 Documentation

Full documentation for end users can be found in the "docs" folder, and is also available online at https://collective-fhirpath.readthedocs.io/

1.6 Contribute

- Issue Tracker: https://github.com/nazrulworld/collective.fhirpath/issues
- Source Code: https://github.com/nazrulworld/collective.fhirpath
- Documentation: https://collective-fhirpath.readthedocs.io/

1.7 Support

If you are having issues, please let us know at: Md Nazrul Islam<email2nazrul@gmail.com>

1.8 License

The project is licensed under the GPLv2.

CHAPTER

TWO

RESTFULL API SERVICE

2.1 Server Side codex examples

Listing 1: Search aka GET Service

```
# -*- coding: utf-8 -*-
from collective.elasticsearch.es import ElasticSearchCatalog
from collective.fhirpath.utils import FHIRModelServiceMixin
from fhirpath.enums import FHIR_VERSION
from fhirpath.interfaces import IElasticsearchEngineFactory
from fhirpath.interfaces import IFhirSearch
from fhirpath.interfaces import ISearchContextFactory
from plone import api
from plone.restapi.services import Service
from zope.component import queryMultiAdapter
from zope.interface import implementer
from zope.publisher.interfaces import IPublishTraverse


@implementer(IPublishTraverse)
class FHIRSearchService(FHIRModelServiceMixin, Service):
    """ """

    def __init__(self, context, request):
        """ """
        super(FHIRSearchService, self).__init__(context, request)
        self.params = []

    def get_es_catalog(self):
        """ """
        return ElasticSearchCatalog(api.portal.get_tool("portal_catalog"))

    def get_factory(self, resource_type, unrestricted=False):
        """ """
        factory = queryMultiAdapter(
            (self.get_es_catalog(),), IElasticsearchEngineFactory
        )
        engine = factory(fhir_release=FHIR_VERSION.STU3)
        context = queryMultiAdapter((engine,), ISearchContextFactory)(
            resource_type, unrestricted=unrestricted
        )

        factory = queryMultiAdapter((context,), IFhirSearch)
```

(continues on next page)

(continued from previous page)

```
        return factory

    def reply(self):
        """ """
        bundle = self.build_result()

        if self.resource_id:
            if bundle.total == 0:
                return self.reply_no_content(404)
            return bundle.entry[0].resource

        return bundle

    def publishTraverse(self, request, name):  # noqa: N802
        # Consume any path segments after /@fhir as parameters
        self.params.append(name)
        return self

    @property
    def resource_id(self):
        """ """
        if 1 < len(self.params):
            return self.params[1]
        return None

    @property
    def resource_type(self):
        """ """

        if 0 < len(self.params):
            _rt = self.params[0]
            return _rt
        return None

    def _get_fhir_fieldname(self, resource_type=None):
        """We assume FHIR Field name is ``{resource type}_resource``"""
        resource_type = resource_type or self.resource_type

        return "{0}_resource".format(resource_type.lower())

    def get_query_string(self):
        """ """
        if self.resource_id:
            return "_id={0}".format(self.resource_id)

        return self.request["QUERY_STRING"]

    def build_result(self):
        """ """
        factory = self.get_factory(self.resource_type)

        return factory(query_string=self.get_query_string())
```

Listing 2: FHIR Resource Add aka POST Service

```
# -*- coding: utf-8 -*-
from Acquisition import aq_base
from Acquisition.interfaces import IAcquirer
from collective.fhirpath.utils import FHIRModelServiceMixin
from plone.restapi.deserializer import json_body
from plone.restapi.exceptions import DeserializationError
from plone.restapi.interfaces import IDeserializeFromJson
from plone.restapi.services import Service
from plone.restapi.services.content.utils import add as add_obj
from plone.restapi.services.content.utils import create as create_obj
from Products.CMFPlone.utils import safe_hasattr
from zope.component import queryMultiAdapter
from zope.event import notify
from zope.interface import alsoProvides
from zope.interface import implementer
from zope.lifecycleevent import ObjectCreatedEvent
from zope.publisher.interfaces import IPublishTraverse

import json
import plone.protect.interfaces


__author__ = "Md Nazrul Islam <nazrul@zitelab.dk>"


@implementer(IPublishTraverse)
class FHIRResourceAdd(FHIRModelServiceMixin, Service):
    """Creates a new FHIR Resource object.
    """

    def __init__(self, context, request):
        """ """
        super(FHIRResourceAdd, self).__init__(context, request)
        self.params = []

    def publishTraverse(self, request, name):  # noqa: N802
        # Consume any path segments after /@fhir as parameters
        self.params.append(name)
        return self

    @property
    def resource_type(self):
        """ """

        if 0 < len(self.params):
            _rt = self.params[0]
            return _rt
        return None

    def reply(self):
        """ """
        data = json_body(self.request)
        # Disable CSRF protection
        if "IDisableCSRFProtection" in dir(plone.protect.interfaces):
            alsoProvides(self.request, plone.protect.interfaces.
↪IDisableCSRFProtection)
```

(continues on next page)

(continued from previous page)

```
        response = self._create_object(data)

        if isinstance(response, dict) and "error" in response:
            self.request.response.setStatus(400)

        return response

    def _create_object(self, fhir):
        """ """
        form_data = {
            "@type": fhir["resourceType"],
            "id": fhir["id"],
            "title": "{0}-{1}".format(self.resource_type, fhir["id"]),
        }
        fhir_field_name = "{0}_resource".format(fhir["resourceType"].lower())
        form_data[fhir_field_name] = fhir

        self.request["BODY"] = json.dumps(form_data)

        context = self.context
        obj = create_obj(
            context, form_data["@type"], id_=form_data["id"], title=form_data["title"]
        )

        if isinstance(obj, dict) and "error" in obj:
            self.request.response.setStatus(400)
            return obj

        # Acquisition wrap temporarily to satisfy things like vocabularies
        # depending on tools
        temporarily_wrapped = False
        if IAcquirer.providedBy(obj) and not safe_hasattr(obj, "aq_base"):
            obj = obj.__of__(context)
            temporarily_wrapped = True

        # Update fields
        deserializer = queryMultiAdapter((obj, self.request), IDeserializeFromJson)
        if deserializer is None:
            self.request.response.setStatus(501)
            return dict(
                error=dict(
                    message="Cannot deserialize type {0}".format(obj.portal_type)
                )
            )

        try:
            deserializer(validate_all=True, create=True)
        except DeserializationError as e:
            self.request.response.setStatus(400)
            return dict(error=dict(type="DeserializationError", message=str(e)))

        if temporarily_wrapped:
            obj = aq_base(obj)

        # Notify Dexterity Created
        if not getattr(deserializer, "notifies_create", False):
```

(continues on next page)

(continued from previous page)

```
            notify(ObjectCreatedEvent(obj))

        # Adding to Container
        add_obj(context, obj, rename=False)

        self.request.response.setStatus(201)
        response = getattr(obj, fhir_field_name)

        self.request.response.setHeader(
            "Location",
            "/".join(
                [
                    self.context.portal_url(),
                    "@fhir",
                    response.resource_type,
                    response.id,
                ]
            ),
        )

        return response
```

Listing 3: FHIR Resource Update aka PATCH Service

```
# -*- coding: utf-8 -*-
from collective.elasticsearch.es import ElasticSearchCatalog
from collective.fhirpath.interfaces import IZCatalogFhirSearch
from collective.fhirpath.utils import FHIRModelServiceMixin
from fhirpath.enums import FHIR_VERSION
from fhirpath.interfaces import IElasticsearchEngineFactory
from fhirpath.interfaces import ISearchContextFactory
from plone import api
from plone.restapi.deserializer import json_body
from plone.restapi.services import Service
from plone.restapi.services.locking.locking import is_locked
from zope.component import queryMultiAdapter
from zope.interface import alsoProvides
from zope.interface import implementer
from zope.publisher.interfaces import IPublishTraverse

import plone.protect.interfaces


@implementer(IPublishTraverse)
class FHIRResourcePatch(FHIRModelServiceMixin, Service):
    """Patch a FHIR Resource object.
    """

    def __init__(self, context, request):
        """ """
        super(FHIRResourcePatch, self).__init__(context, request)
        self.params = []

    def publishTraverse(self, request, name):  # noqa: N802
        # Consume any path segments after /@fhir as parameters
        self.params.append(name)
```

(continues on next page)

(continued from previous page)

```
        return self

    def get_es_catalog(self):
        """ """
        return ElasticSearchCatalog(api.portal.get_tool("portal_catalog"))

    def get_factory(self, resource_type, unrestricted=False):
        """ """
        factory = queryMultiAdapter(
            (self.get_es_catalog(),), IElasticsearchEngineFactory
        )
        engine = factory(fhir_release=FHIR_VERSION.STU3)
        context = queryMultiAdapter((engine,), ISearchContextFactory)(
            resource_type, unrestricted=unrestricted
        )

        factory = queryMultiAdapter((context,), IZCatalogFhirSearch)
        return factory

    @property
    def resource_id(self):
        """ """
        if 1 < len(self.params):
            return self.params[1]
        return None

    @property
    def resource_type(self):
        """ """

        if 0 < len(self.params):
            _rt = self.params[0]
            return _rt
        return None

    def reply(self):
        """ """
        query_string = "_id={0}".format(self.resource_id)

        factory = self.get_factory(self.resource_type)
        brains = factory(query_string=query_string)

        if len(brains) == 0:
            self.reply_no_content(404)

        obj = brains[0].getObject()

        if is_locked(obj, self.request):
            self.request.response.setStatus(403)
            return dict(error=dict(type="Forbidden", message="Resource is locked."))

        data = json_body(self.request)

        # Disable CSRF protection
        if "IDisableCSRFProtection" in dir(plone.protect.interfaces):
            alsoProvides(self.request, plone.protect.interfaces.
↪IDisableCSRFProtection)
```

(continues on next page)

(continued from previous page)

```
        fhir_value = getattr(obj, "{0}_resource".format(self.resource_type.lower()))
        fhir_value.patch(data["patch"])

        self.request.response.setStatus(204)
        # Return None
        self.reply_no_content(204)
```

Listing 4: REST Service registration (configuration.zcml)

```
<configure xmlns="http://namespaces.zope.org/zope"
  xmlns:plone="http://namespaces.plone.org/plone"
  xmlns:zcml="http://namespaces.zope.org/zcml">


  <include package="plone.rest" file="configure.zcml" />
  <plone:service method="GET"
                 name="@fhir"
                 for="Products.CMFCore.interfaces.ISiteRoot"
                 factory=".get.FHIRSearchService"
                 permission="zope2.View"
  />

  <plone:service method="POST" name="@fhir" for="Products.CMFCore.interfaces.ISiteRoot
↪" factory=".post.FHIRResourceAdd" permission="cmf.ManagePortal" />

  <plone:service method="PATCH" name="@fhir" for="Products.CMFCore.interfaces.
↪ISiteRoot" factory=".patch.FHIRResourcePatch" permission="cmf.ManagePortal" />

</configure>
```

2.2 REST Client Examples

Getting single resource, here we are getting Patient resource by ID.

Example(1):

```
>>> response = admin_session.get('/@fhir/Patient/19c5245f-89a8-49f8-b244-666b32adb92e
↪')
>>> response.status_code
200

>>> response.json()['resourceType'] == 'Patient'
True

>>> response = admin_session.get('/@fhir/Patient/19c5245f-fake-id')
>>> response.status_code
404
```

Search Observation by Patient reference with status condition. Any observation until December 2017 and earlier than January 2017.

Example(2):

```
>>> response = admin_session.get('/@fhir/Observation?patient=Patient/19c5245f-89a8-
↪49f8-b244-666b32adb92e&status=final&_lastUpdated=lt2017-12-31T00%3A00%3A00%2B00
↪%3A00&_lastUpdated=gt2017-01-01T00%3A00%3A00%2B00%3A00')
>>> response.status_code
200
>>> response.json()["total"]
1
```

Add FHIR Resource through REST API

Example(3):

```
>>> import os
>>> import json
>>> import uuid
>>> import DateTime
>>> import time

>>> with open(os.path.join(FIXTURE_PATH, 'Patient.json'), 'r') as f:
...     fhir_json = json.load(f)

>>> fhir_json['id'] = str(uuid.uuid4())
>>> fhir_json['name'][0]['text'] = 'Another Patient'
>>> response = admin_session.post('/@fhir/Patient', json=fhir_json)
>>> response.status_code
201
>>> time.sleep(1)
>>> response = admin_session.get('/@fhir/Patient?active=true')
>>> response.json()["total"]
2
```

Update (PATCH) FHIR Resource the Patient is currently activated, we will deactive.

Example(4):

```
>>> patch = [{'op': 'replace', 'path': '/active', 'value': False}]
>>> response = admin_session.patch('/@fhir/Patient/19c5245f-89a8-49f8-b244-
↪666b32adb92e', json={'patch': patch})
>>> response.status_code
204
```

CHAPTER

THREE

CHANGELOG

3.1 0.6.1 (2020-09-09)

- `plone.app.fhirfield:default``has been added in dependency, so no need separete install of ``plone.app.fhirfield`.

3.2 0.6.0 (2020-09-09)

Improvements

- `FHIRModelServiceMixin` class has been available under `utils` module, which can be used with your `plone.restapi` services to response type as `FhirModel` aka pydantic's `BaseModel` or `plone.app.fhirfield.FhirFieldValue` object with the best possible effecient way.

3.3 0.5.0 (2020-08-18)

Improvements

- Supports the revolutionary version of fhir.resources via fhirpath we may expect some refactor on your existing codebase because of some breaking changes, please see changes at `fhir.resources`, `plone.app.fhirfield` and `fhirpath`.
- Brings back support for Python version 3.6
- Three configurations (`fhirpath.es.index.mapping.nested_fields.limit`, `fhirpath.es.index.mapping.depth.limit`, `fhirpath.es.index.mapping.total_fields.limit`) based on plone registry has now been available.

3.4 0.4.0 (2020-05-15)

Breakings

- As a part of supporting latest `fhirpath` version (from `0.6.1`), drop python version later than `3.7.0`.
- `ElasticsearchEngineFactory.__call__`'s argument name `fhir_version` changed to `fhir_release`.

3.5 0.3.0 (2019-11-10)

Improvements

- ZCatalog featured fhir search added, from which you will get ZCatalog´s brain feature.
- `FhirFieldIndex` named PluginIndex is now available.
- FHIR `STU3``and ``R4` search mapping is now available.
- Others improvements that make able to use in production project (of course without guarantee.)

3.6 0.2.0 (2019-09-16)

- first working versions, with lots of improvements.

3.7 0.1.0 (2019-09-06)

- Initial release. [nazrulworld]