guillotina$_{f}hirfieldDocumentation$

Release 0.1.0a1

Md Nazrul Islam

Apr 15, 2019

Contents:

CHAPTER 1

guillotina_fhirfield

FHIR field for guillotina.

- Free software: BSD license
- Documentation: https://guillotina-fhirfield.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install guillotina_fhirfield, run this command in your terminal:

```
$ pip install guillotina_fhirfield
```

This is the preferred method to install guillotina_fhirfield, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for guillotina_fhirfield can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/nazrulworld/guillotina_fhirfield
```

Or download the tarball:

```
$ curl  -OL https://github.com/nazrulworld/guillotina_fhirfield/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use guillotina_fhirfield in a project:

```
import guillotina_fhirfield
```

CHAPTER 4

guillotina_fhirfield

4.1 guillotina_fhirfield package

4.1.1 Subpackages

guillotina_fhirfield.tests package

Submodules

guillotina_fhirfield.tests.conftest module

guillotina_fhirfield.tests.fhir_contents module

guillotina_fhirfield.tests.fixtures module

guillotina_fhirfield.tests.helpers module

class `guillotina_fhirfield.tests.helpers.`**`NoneInterfaceClass`**

Bases: `object`

docstring for ClassName

guillotina_fhirfield.tests.test_field module

guillotina_fhirfield.tests.test_guillotina_fhirfield module

Tests for *guillotina_fhirfield* package.

`guillotina_fhirfield.tests.test_guillotina_fhirfield.`**`test_command_line_interface`**`()`

Test the CLI.

guillotina_fhirfield.tests.test_guillotina_fhirfield.**test_content**(*dummy_request*, *dummy_guillotina*)

Sample pytest test function with the pytest fixture as an argument.

Module contents

4.1.2 Submodules

4.1.3 guillotina_fhirfield.cli module

Console script for guillotina_fhirfield.

4.1.4 guillotina_fhirfield.exc module

exception guillotina_fhirfield.exc.**SearchQueryError**

Bases: `zope.interface.exceptions.Invalid`

exception guillotina_fhirfield.exc.**SearchQueryValidationError**

Bases: *guillotina_fhirfield.exc.SearchQueryError*

4.1.5 guillotina_fhirfield.field module

class guillotina_fhirfield.field.**DefaultFhirFieldSchemaSerializer**(*field*, *schema*, *request*)

Bases: `guillotina.json.serialize_schema_field.DefaultSchemaFieldSerializer`

field_type

class guillotina_fhirfield.field.**FhirField**(*resource_class=None*, *resource_interface=None*, *resource_type=None*, ***kw*)

Bases: `guillotina.schema._field.Object`

FhirResource also known as FHIR field is the schema field derrived from z3c.form's field.

It takes all initilial arguments those are derrived from standard schema field, with additionally `model`, `resource_type` and `resource_interface`

Note: field name must be start with lowercase name of FHIR Resource.

from_dict(*dict_value*)

from_unicode(*str_val*)

class guillotina_fhirfield.field.**FhirFieldValue**(*obj: NewType.<locals>.new_type = None*)

Bases: `object`

FhirResourceValue is a proxy class for holding any object derrived from fhir.resources.resource.Resource

foreground_origin()

Return the original object of FHIR model that is proxied!

patch(*patch_data*)

stringify(*prettify=False*)

guillotina_fhirfield.field.**fhir_field_deserializer**(*fhirfield*, *value*, *context=None*)

guillotina_fhirfield.field.**fhir_field_from_resource_type**(*resource_type: str*, *cache: bool = True*) → Optional[dict]

guillotina_fhirfield.field.**fhir_field_from_schema**(*schema: <InterfaceClass zope.interface.Interface>*, *resource_type: str = None*) → Optional[guillotina_fhirfield.field.FhirField]

guillotina_fhirfield.field.**fhir_field_value_serializer**(*value*)

4.1.6 guillotina_fhirfield.helpers module

guillotina_fhirfield.helpers.**fhir_resource_mapping**(*resource_type: str*, *cache: bool = True*) → dict

guillotina_fhirfield.helpers.**fhir_search_path_meta_info**(*path: str*) → Optional[tuple]

guillotina_fhirfield.helpers.**filter_logic_in_path**(*raw_path: str*) → str
Separates if any logic_in_path is provided

guillotina_fhirfield.helpers.**import_string**(*dotted_path: str*) → type
Shameless hack from django utils, please don't mind!

guillotina_fhirfield.helpers.**parse_json_str**(*str_val: str*, *encoding: str = 'utf-8'*) → Optional[dict]

guillotina_fhirfield.helpers.**parse_query_string**(*request*, *allow_none=False*)
We are not using self.request.form (parsed by Zope Publisher)!! There is special meaning for colon(:) in key field. For example *field_name:list* treats data as List and it doesn't recognize FHIR search modifier like :not, :missing as a result, from colon(:) all chars are ommited.

Another important reason, FHIR search supports duplicate keys (defferent values) in query string.

Build Duplicate Key Query String ::

```
>>> import requests
>>> params = {'patient': 'P001', 'lastUpdated': ['2018-01-01', 'lt2018-09-10
↪']}
>>> requests.get(url, params=params)
>>> REQUEST['QUERY_STRING']
'patient=P001&lastUpdated=2018-01-01&lastUpdated=lt2018-09-10'
```

```
>>> from six.moves.urllib.parse import urlencode
>>> params = [('patient', 'P001'), ('lastUpdated', '2018-01-01'), (
↪'lastUpdated', 'lt2018-09-10')]
>>> urlencode(params)
'patient=P001&lastUpdated=2018-01-01&lastUpdated=lt2018-09-10'
```

param:request param:allow_none

guillotina_fhirfield.helpers.**resource_type_to_resource_cls**(*resource_type: str*, *fhir_release: str = None*) → Union[zope.interface.exceptions.Invalid, type] `guillotina_fhirfield.helpers.`**`search_fhir_resource_cls`**(*resource_type: str*, *cache: bool = True*, *fhir_release: str = None*) → Optional[str]

This function finds FHIR resource model class (from fhir.resources) and return dotted path string.

Parameters

- **`resource_type`** – the resource type name (required). i.e Organization
- **`cache`** – (default True) the flag which indicates should query fresh or serve from cache if available.
- **`fhir_release`** – FHIR Release (version) name. i.e STU3, R4

:return dotted full string path. i.e fhir.resources.organization.Organization

Example:

```
>>> from guillotina_fhirfield.helpers import search_fhir_resource_cls
>>> from zope.interface import Invalid
>>> dotted_path = search_fhir_resource_cls('Patient')
>>> 'fhir.resources.patient.Patient' == dotted_path
True
>>> dotted_path = search_fhir_resource_cls('FakeResource')
>>> dotted_path is None
True
```

`guillotina_fhirfield.helpers.`**`translate_param_name_to_real_path`**(*param_name*, *resource_type=None*)

`guillotina_fhirfield.helpers.`**`validate_resource_type`**(*resource_type: str*) → None

FHIR resource type validation

4.1.7 guillotina_fhirfield.interfaces module

Module where all interfaces, events and exceptions live.

4.1.8 guillotina_fhirfield.patch module

`guillotina_fhirfield.patch.`**`patch_fhir_base_model`**()

“

4.1.9 guillotina_fhirfield.variables module

4.1.10 Module contents

`guillotina_fhirfield.`**`includeme`**(*root*)

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/nazrulworld/guillotina_fhirfield/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

guillotina_fhirfield could always use more documentation, whether as part of the official guillotina_fhirfield docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/nazrulworld/guillotina_fhirfield/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *guillotina_fhirfield* for local development.

1. Fork the *guillotina_fhirfield* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/guillotina_fhirfield.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv guillotina_fhirfield
$ cd guillotina_fhirfield/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 guillotina_fhirfield tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/nazrulworld/guillotina_fhirfield/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_guillotina_fhirfield
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Md Nazrul Islam <email2nazrul@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

CHANGES

7.1 0.1.0a2 (unreleased)

- Nothing changed yet.

7.2 0.1.0a1 (2018-12-28)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index