ckanext.importer Documentation

Stadt Karlsruhe

Sep 24, 2018

Contents

ckanext.importer provides utilities for easily importing metadata from an external data source into CKAN and keeping the CKAN metadata up-to-date when the contents of the data source is modified.

To achieve this, each entity (package, resource, view) in CKAN is linked to its counterpart in the original data source via an **external ID** (**EID**), for example the entity's ID in the data source.

As an example, let's create a package with a resource:

```
from ckanext.importer import Importer

imp = Importer('my-importer-id')

with imp.sync_package('my-package-eid') as pkg:
    # If no package with the given EID exists then it is
    # automatically created. Otherwise the existing package
    # is retrieved.

    # The package can be modified like a dict
    pkg['title'] = 'My Package Title'

    # For package extras there's the special `.extras` attribute
    # which provides a dict-interface:
    pkg.extras['my-extra-key'] = 'my-extra-value'

    with pkg.sync_resource('my-resource-eid') as res:
        # Just like packages, resources are automatically created
        # and retrieved based on their EID.
        res['name'] = 'My Resource Name'

    # Once the `sync_resource` context manager exists, the
    # created/updated resource is automatically uploaded to CKAN.

# Once the `sync_package` context manager exists, the created/updated
# package is automatically uploaded to CKAN.
```

For more details on how to use *ckanext.importer* please refer to *Usage*.

CHAPTER 1

Installation

ckanext.importer uses the usual installation routine for CKAN extensions:

1. Activate your CKAN virtualenv:

```
cd /usr/lib/ckan/default
source bin/activate
```

2. Install *ckanext.importer* and its dependencies:

```
pip install -e git+https://github.com/stadt-karlsruhe/ckanext-importer
↪#egg=ckanext-importer
pip install -r src/ckanext-importer/requirements.txt
```

On a production system you'll probably want to pin a certain release version of *ckanext.importer* instead:

```
pip install -e git+https://github.com/stadt-karlsruhe/ckanext-importer@v0.1.0
↪#egg=ckanext-importer
```

3. Restart CKAN. For example, if you're using Apache,

```
sudo service apache2 restart
```

CHAPTER 2

Usage

ckanext.importer provides utilities to write Python code for importing and synchronizing CKAN metadata from an external data source.

Note: At this point in time, *ckanext.importer* does *not* provide a web UI or any command line tools.

The starting point for using *ckanext.importer* is an `Importer`. Each `Importer` instance corresponds to a separate data source and is identified using an ID that can be freely chosen (but must be unique among all importers used on the target CKAN instance):

```
from ckanext.importer import Importer

imp = Importer('my-importer-id')
```

Once you have created an importer, you use its `sync_package()` method to create/update the CKAN metadata for a dataset. The CKAN package is linked to your external dataset using an external ID (EID). *ckanext.importer* automatically stores the EID along with the other package metadata inside CKAN. Like the importer ID, the package's EID can be chosen freely, but must be unique among all packages for this importer.

```
with imp.sync_package(eid='my-package-eid') as pkg:
    # ckanext.importer has automatically checked whether a
    # package for this importer ID and package EID already
    # exists and -- if that is the case -- retrieved it.
    # Otherwise, a suitable package has been automatically
    # created for you.

    # Use the package's dict-interface to insert/update the
    # metadata from your data source. For example:
    pkg['title'] = 'My Package Title'

# Once the context manager exits, the modified package is
# automatically uploaded to CKAN.
```

Typically, you don't have only one dataset, but an external data source (for example a database) containing many datasets to be imported:

```
for external_dataset in external_datasource:
    with imp.sync_package(eid=external_dataset.id) as pkg:
        pkg['title'] = external_dataset.name
```

Synchronizing a package's resources works pretty much the same: the object returned by `sync_package()` is an instance of `Package` and provides a `sync_resource()` method:

```
with imp.sync_package(eid='my-package-eid') as pkg:
    pkg['title'] = 'My Package Title'

    with pkg.sync_resource(eid='my-resource-eid') as res:
        res['name'] = 'My Resource Name'
        res['url'] = 'https://some-resource-url'
```

Resource EIDs need to be unique among all resources of the same package.

Finally, the same mechanism can be used to synchronize resource views via the `Resource.sync_view()` method (which returns a `View` instance):

```
with pkg.sync_resource(eid='my-resource-eid') as res:
    res['name'] = 'My Resource Name'
    res['url'] = 'https://some-resource-url'

    with res.sync_view(eid='my-view-eid') as view:
        view['view_type'] = 'text_view'
        view['title'] = 'My View Title'
```

See the *API Reference* for more information.

CHAPTER 3

License

CHAPTER 4

Changelog

See the file CHANGELOG.md.

CHAPTER 5

API Reference