sqla-inspect Documentation

Release 0.1.1

Gaston Tjebbes - Majerti

Dec 11, 2018

Contents

Contents:

CHAPTER 1

Exportation tools

sqla-inspect provide tools to export sqlalchemy models as :

- A py3o template context
- A csv file
- An excel file

1.1 Py3o Context

py3o is an elegant and scalable solution to design reports using LibreOffice or OpenOffice. py3o.template is the templating component that takes care of merging your data sets with a corresponding templated OpenOffice document.

More here : **'https://pypi.python.org/pypi/py3o.template'_**

sqla-inspect allows to compile a template using a sqlalchemy model instance as context.

```
from sqla_inspect.py3o import compile_template
from .models import DBSession, MyModel

mymodel_instance = DBSession.query(MyModel).first()
output_buffer = compile_template(
    mymodel_instance,
    "/tmp/mytemplate.odt",
    {'custom_key': 'custom_value'},
)
```

The *output_buffer* is a cStringIO.StringIO() object.

sqla-inspect essentially build the templating context through the sqla_inspect.py3o.get_compilation_context.

1.1.1 Relationships handling strategy

The model is inspected in depth, recursively through the relationships.

if you have

```
class Parent(Base):
   id = Column(Integer, primary_key=True)
   name = Column(String(255))


class Child(Base):
   id = Column(Integer, primary_key=True)
   parent_id = Column(ForeignKey('parent.id'))
   parent = relationship('Parent', backref='children')
```

```
>>> print py3o.get_template_context(child)
{
    'id': 1,
    'parent_id': 5,
    'name': 'Youssouf',
    'parent': {'id': 5, 'name': 'Moha'}
}
```

```
>>> print py3o.get_template_context(parent)
{
    'id': 5,
    'name': 'Moha'
    'children': {
        'l': [
            {'id': 1, 'parent_id': 5, 'name': 'Youssouf'}
            {'id': 2, 'parent_id': 5, 'name': 'Myriam'}
        ],
        '0': {'id': 1, 'parent_id': 5, 'name': 'Youssouf'}
        '1': {'id': 2, 'parent_id': 5, 'name': 'Myriam'}
    }
}
```

Here children datas could be accessed :

- In a for loop : through the 'l' key
- By index : through the '0', '1' key (e.g : 'py3o.children.1.name')

WARNING : You need to provide exclude arguments to avoid recursive loops

See *Customization* to know how to set custom formatters or to exclude columns from export.

1.2 Csv Export

```
from sqla_inspect.csv import SqlaCsvExporter

exporter = SqlaCsvExporter(Model)
for row in Model.query():
    exporter.add_row(row)
with open('/tmp/test.csv', 'w') as f_buffer:
    exporter.render(f_buffer)
```

See *Customization* to know how to set custom formatters, labels, exclusions and relationship formatters.

1.3 Excel export

```
from sqla_inspect.csv import SqlaXlsExporter

exporter = SqlaXlsExporter(Model)
for row in Model.query():
    exporter.add_row(row)
with open('/tmp/test.xls', 'w') as f_buffer:
    exporter.render(f_buffer)
```

See *Customization* to know how to set custom formatters, labels, exclusions and relationship formatters.

1.4 Customization

1.4.1 Globally

You can globally set formatters through which value of a specific type will be passed before export.

```
from sqla_inspect.export import FORMATTERS_REGISTRY
FORMATTERS_REGISTRY.add_formatter(
    sqlalchemy.Boolean,
    lambda val: 'Y' and val or 'N'
)
```

All booleans will be converted to 'Y' or 'N'. If you want to do this formatting only for a specific export, add a key (the key as configured in the exporter class, csv/excel/py3o for provided exporters)

```
from sqla_inspect.export import FORMATTERS_REGISTRY
FORMATTERS_REGISTRY.add_formatter(
    sqlalchemy.Boolean,
    lambda val: 'Y' and val or 'N',
    'csv'
)
```

You can globally blacklist some fields to avoid exporting them

```
from sqla_inspect.export import BLACKLISTED_KEYS
BLACKLISTED_KEYS = ('_acl', 'password')
```

1.4.2 Per Column

You can customize columns informations:

- The header label through the 'label' key
- The way a relationship is exported through the 'related_key' (the attribute on the related object that will replace the related object)
- The way the datas is formatted providing a formatter under the 'formatter' key
- Exclude a column setting the 'exclude' key

All this keys can be set at differente levels:

If you want to customize the header labels, you can provide informations in the export/csv key

```
class Model(Base):
    attr1 = Column(
        Integer,
        info={'export': {'csv': {'label': u'My custom header'}}}
    )
```

If not set it will look one level higher in the export key

```
class Model(Base):
    attr1 = Column(
        Integer,
        info={'export': {'label': u'My custom header'}}
    )
```

If not set, only in the case of labels, it will look into the colanderalchemy 'title' attribute (info={'colanderalchemy': {'title': u'My title'}}).

The same things can be done with the excel.SqlaXlsExporter class (that shares the export dict with the SqlaCsvExporter.

1.5 LIMITATIONS

Relationship that point to lists are not handled yet.

CHAPTER 2

sqla_inspect package

2.1 Subpackages

2.1.1 sqla_inspect.tests package

Submodules

sqla_inspect.tests.test_base module

Module contents

2.2 Submodules

2.3 sqla_inspect.ascii module

2.4 sqla_inspect.base module

2.5 sqla_inspect.csv module

2.6 sqla_inspect.excel module

2.7 sqla_inspect.export module

2.8 sqla_inspect.py3o module

2.9 Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search