Tincture Documentation

Release 0.0.0

Joe Friedl

May 06, 2013
CONTENTS

1 Compatibility 3

2 Contents 5
   2.1 User Guide ................................................................. 5
   2.2 API Documentation .................................................... 6
Tincture is a collection of Django components adapted for use with SQLAlchemy.
Currently, support for Django 1.4 and SQLAlchemy 0.7 is planned. Support will be expanded to newer versions once Tincture becomes stable.
2.1 User Guide

2.1.1 Class-based generic views

Many of Django’s class-based generic views can be used with SQLAlchemy without modification. For those that don’t, Tincture attempts to provide substitutions.

The APIs of Tincture’s generic views differ slightly from Django’s, where appropriate. For example, many of the views have a `session` attribute that holds a SQLAlchemy `Session` object.

Simple Usage

Say you have the following SQLAlchemy model:

```python
# some_app/models.py
from sqlalchemy.orm import Column, Integer, String
class Person(Base):
    __tablename__ = 'people'
    id = Column(Integer, primary_key=True)
    name = Column(String)
```

You could create a list view for `Person` objects much the same way as with Django’s `ListView`, with the addition of the `session` attribute:

```python
# some_app/views.py
from tincture.views.generic import ListView
from some_app.models import Person

class PersonListView(ListView):
    session = db.session
    model = Person
```

**Note:** The session object can’t be obtained automatically, so it must be provided or `ImproperlyConfigured` will be raised when the view tries to retrieve any model objects.
PersonListView could then be added to your urlconf like any other class-based view:

```python
# some_project/urls.py
from django.conf.urls import patterns, url, include
from some_app.views import PersonListView

urlpatterns = patterns('',
   (r'^people/$', PersonListView.as_view()),
)```

### 2.2 API Documentation

#### 2.2.1 Class-based generic views

**Note:** This documentation only documents the *differences* between Django and Tincture. For all other functionality, see Django’s documentation.

**Mixins**

**Session**

**SessionMixin**

```python
class tincture.views.generic.session.SessionMixin
```

- **session**
  - A SQLAlchemy session instance.

- **get_session()**
  - Returns a SQLAlchemy session instance. By default, this returns `SessionMixin.session`.

**Single object mixins**

**SingletonMixin**

```python
class tincture.views.generic.detail.SingleObjectMixin
```

- **model**
  - The SQLAlchemy model that this view will display data for. Specifying `model = Foo` is effectively the same as specifying `query_object = session.query(Foo)`.

- **query_object**
  - A SQLAlchemy Query object.

- **session**
  - A SQLAlchemy Session instance. This session will be used to generate a Query object for the view.

- **pk_url_kwarg**
  - A tuple of URLconf keyword argument names that comprise the primary key. The default is (`'pk'`). Note that this is named differently from Django’s `SingleObjectMixin.pk_url_kwarg`. 
get_object (query_object=None)
Returns the single object this view will display. If query_object is provided, it will be used to obtain
the object. Otherwise, get_query_object() will be used.

get_query_object ()
Returns the SQLAlchemy Query object that represents the data this view will display.
See Django’s SingleObjectMixin.get_queryset.

SingleObjectTemplateResponseMixin
class tincture.views.generic.detail.SingleObjectTemplateResponseMixin

get_template_names ()
Returns a list of template names. In addition to the template names from Django that aren’t based on the
model, this returns:
• <lowercase model class name><template_name_suffix>.html

Note: With SQLAlchemy there’s no built-in way to detect an app name, so you may want to provide your
own implementation of this method to make template discovery more robust.
See Django’s SingleObjectTemplateResponseMixin.get_template_names.

Multiple object mixins

MultipleObjectMixin
class tincture.views.generic.list.MultipleObjectMixin

model
The SQLAlchemy model that this view will display data for. Specifying model = Foo is effectively the
same as specifying query_object = session.query(Foo).

query_object
A SQLAlchemy Query object.
See Django’s MultipleObjectMixin.queryset.

session
A SQLAlchemy Session instance. This session will be used to generate a Query object for the view.

get_query_object ()
Returns the SQLAlchemy Query object that represents the data this view will display.
See Django’s MultipleObjectMixin.get_queryset.

paginate_query_object (query_object, page_size)
See Django’s MultipleObjectMixin.paginate_queryset.

get_paginate_by (query_object)
See Django’s MultipleObjectMixin.get_paginate_by.

getaginator (query_object, per_page,orphans=0, allow_empty_first_page=True)
See Django’s MultipleObjectMixin.getaginator.

get_context_object_name (object_list)
Returns the context variable name that will be used to contain the list of data that this view is manipulating.
If object_list is a SQLAlchemy Query object, it’ll use the lowercased name of the first entity in the query. For example, a query for the Person and Dog models will return ‘person_list’.

MultipleObjectTemplateResponseMixin
class tincture.views.generic.list.MultipleObjectTemplateResponseMixin

get_template_names()
Returns a list of template names. In addition to the template names from Django that aren’t based on the model, this returns:

•<lowercase model class name><template_name_suffix>.html

The Query object’s first entity is used to retrieve the class name used here.

Note: With SQLAlchemy there’s no built-in way to detect an app name, so you may want to provide your own implementation of this method to make template discovery more robust.

See Django’s MultipleObjectTemplateResponseMixin.get_template_names.

Generic views

Detail views

DetailView
class tincture.views.generic.detail.BaseDetailView
class tincture.views.generic.detail.DetailView

See Django’s DetailView.

Mixins

•tincture.views.generic.session.SessionMixin
•tincture.views.generic.detail.SingleObjectMixin
•tincture.views.generic.detail.SingleObjectTemplateResponseMixin

List views

ListView
class tincture.views.generic.list.BaseListView
class tincture.views.generic.list.ListView

See Django’s ListView.

Mixins

•tincture.views.generic.session.SessionMixin
•tincture.views.generic.list.MultipleObjectMixin
•tincture.views.generic.list.MultipleObjectTemplateResponseMixin