CSV Importer Documentation

Release 0.1

Anthony TRESONTANI

April 05, 2016

Contents

Django adaptor is a tool which allow you to transform easily a CSV/XML file into a python object or a django model instance. It is based on the django-style declarative model.

CHAPTER 1

Installation

Use either `easy_install`:

```
easy_install django-adaptors
```

or `pip`:

```
pip install django-adaptors
```

CHAPTER 2

CSV data

2.1 Basic example

Consider the following:

```
>>> from adaptor.model import CsvModel
>>> class MyCSvModel(CsvModel):
...     name = CharField()
...     age = IntegerField()
...     length = FloatField()
...
...     class Meta:
...         delimiter = ";"
```

You declare a `MyCsvModel` which will match to a CSV file like this:

    Anthony;27;1.75

To import the file or any iterable object, just do:

```
>>> my_csv_list = MyCsvModel.import_data(data = open("my_csv_file_name.csv"))
>>> first_line = my_csv_list[0]
>>> first_line.age
27
```

Without an explicit declaration, data and columns are matched in the same order:

```
Anthony --> Column 0 --> Field 0 --> name
27      --> Column 1 --> Field 1 --> age
1.75    --> Column 2 --> Field 2 --> length
```

2.2 Django Model

If you now want to interact with a django model, you just have to add a **dbModel** option to the class meta.

```
>>> from adaptor.model import CsvModel
>>> class MyCSvModel(CsvModel):
...     name = CharField()
...     age = IntegerField()
...     length = FloatField()
...
...     class Meta:
```

```
...         delimiter = ";"
...         dbModel = Person
```

That will automatically match to the following django model.

```
>>> class Person(models.Model):
...     name = CharField(max_length = 100)
...     age = IntegerField()
...     length = FloatField()
```

If field names of your Csv model does not match the field names of your django model, you can manage this with the match keyword:

```
>>> from adaptor.model import CsvModel
>>> class MyCSvModel(CsvModel):
...     fullname = CharField(match = "name")
...
```

If you don't want to have to re-declare a CSV model whereas the Django model already exist, use a CsvDbModel.

```
>>> from my_projects.models import Person
>>> from adaptor.model import CsvDbModel
>>>
>>> class MyCsvModel(CsvDbModel):
...
...     class Meta:
...         dbModel = Person
...         delimiter = ";"
```

The django model should be imported in the model

2.3 Fields

Fields available are:

- **IntegerField** : return an int
- **DecimalField**: return a decimal.Decimal
- **FloatField** : return a float
- **CharField** : return a string
- **DateField** : return a datetime
- **ForeignKey** : return a django model object
- **IgnoredField** : skip the value
- **ComposedKeyForeign** : return a django model object retrieve with multiple values as keys.
- **BooleanField** : return a boolean

Options :

You can give, as argument, the following options:

row_num

define the position in the file for this field.

match

define the django model name matching this field. If a list is defined, all the field matching will received the value.

transform

Apply the function before returning the result. You can also define a function called transform_<attribute_name>.

def transform_username(self, username): return username.capitalize()

prepare

Apply the function on the raw value (still a string).

validator

A class which should implement a validate function: def validate(self, value): and return a Boolean. This allow to apply some business validation on the object before uploading.

multiple

Allow a field to read as many values as the number of remaining data on the line.

keys

A list of fields which composed the key. Only for **ComposedKeyForeign**.

is_true

a function which determine when a boolean is True. Only for **BooleanField**.

Here is an example of a way to use the transform attribute. >>> from adaptor.model import CsvModel >>> >>> class MyCsvModel(CsvModel): >>> >>> user = ForeignKey(transform = lambda user: user.username)

ForeignKey has an additional argument:

pk

allow you to define on which value the object will be retrieved.

You can also skip a row during `prepare`, `transform` or in a `validator` by raising a SkipRow exception.

2.4 Meta options

delimiter

define the delimiter of the csv file. If you do not set one, the sniffer will try to find one itself.

has_header

Skip the first line if True.

dbModel

If defined, the importer will create an instance of this model.

silent_failure

If set to True, an error in a imported line will not stop the loading.

exclude

CsvDbModel only. To do take into account the django field of the django model defined in this list.

layout

Set it to LinearLayout (by default) or Tabular Layout. Modify the way your data are organised in.22 the file. Tabular read:

B1 B2 B3

A1 C1 C2 C3 A2 C4 C5 C6 –> (A1,B1,C1), (A1,B2,C2), (A1,B3,C3), (A2,B1,C4) ... A3 C7 C8 C9

update

Set as a dictionnary with the 'keys' value defining the list of 'natural keys'. If the value is found, update instead of creating a new object. If the value is not found, create a new object.

2.5 Importer option

When importing data, you can add an optional argument *extra_fields* which is a string or a list. This allow to add a value to any line of the csv file before the loading.

2.6 Grouped CSV

If you want to create more than object by line, you should use a group CSV model. This object will create the object in the same order than the csv_models attribute provided.

csv_models

list of csv model, processed in the same order than the list

CHAPTER 3

USING XML

The xml adaptor is using XPATH to retrieve information from the XML file.

3.1 Fields

Fields are the same but are called XML<FieldName>. For example, IntegerField –> XMLIntegerField.

There is 2 specifics XML field: XMLRoot and XMLEmbed.

XMLRoot

allow you to define the root XML element of your tree. If you want to retrieve multiple items, it should be set root element distinguish these items.

XMLEmbed

can be seen as an inner XMLEmbed element. Used to defined list of elements or just to organise your code better.

Supported parameters now are: prepare, transform and is_true for XMLBooleanField. Some additionnal supported parameters are:

path

required. The XPath expression to find the XML element.

null

Is set to True, if the value is not found, do not raise an exception. Default is False.

default

If null is set to True and no value is found, return this default value instead of None.

attribute

If set, will use this attribute instead of the text value of the XML element

3.2 Meta

There is no meta option supported for the moment.

CHAPTER 4

More samples

Just look at the tests.py file in the adaptor folder.

CHAPTER 5

Contributing

Clone the repo to your local workspace then run:

```
mkvirtualenv adaptors
python setup.py develop
pip install -r requirements.txt
```

and you should be able to see the tests pass by running:

```
make test
```

CHAPTER 6

Any Questions

For any questions, you can contact me at csv.tresontani@gmail.com