CSV Importer Documentation

Release 0.1

Anthony TRESONTANI

Jul 18, 2017

Contents

1 Installation 3

2 Basic sample 5

3 Django Model 7

4 Fields 9

5 Meta options 11

6 Importer option 13

7 Grouped CSV 15

8 Any Questions 17

Contents: **CSV importer** is a tool which allow you to transform easily a csv file into a python object or a django model instance. It is based on a django-style declarative model.

CHAPTER 1

Installation

Simple, like Pypi package:

easy_install csvImporter

or with Pip

pip install csvImporter

CHAPTER 2

Basic sample

Here is a basic sample:

```
>>> 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

CHAPTER 3

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 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:

```
>>> 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 csvImporter.model import CsvDbModel
>>>
>>> class MyCsvModel(CsvDbModel):
>>>
>>>     class Meta:
```

```
>>>         dbModel = Person
>>>         delimiter = ";"
```

The django model should be imported in the model

CHAPTER 4

Fields

Fields available are:

- **IntegerField** : return an int
- **FloatField** : return a float
- **CharField** : return a string
- **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.

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.

```
>>> 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.

CHAPTER 5

Meta options

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

CHAPTER 6

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.

CHAPTER 7

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 8

Any Questions

For any question, you can contact my at csv.tresontani@gmail.com