Generate CSV test data at random from template

Asked 9 years, 7 months ago Modified 9 years, 7 months ago

Viewed 277 times

0

I am going to have to generate many CSV files that will contain random-ish data. There will be rules about the fields, such as some will be integers, some should be names picked from a particular list, some will be text generated from a Markov chain with a given source, etc.

I would like to make this flexible, so the CSV specification/template could be changed without needing to make any coding changes. My first thought is to have the template itself also be a CSV which will map the field name to the rules for generating data, so a template might be something like the following:

```
device id,randint,unique
description,markov,tech.source
vendor,choice,vendors.txt
```

And from that I could generate a CSV with 3 fields, the first being an ID made of random ints with a "unique" constraint, the second being a narrative description from a Markov chain using "tech.source" as the chain definition, and the vendor being a random selection from those defined in another text file.

Since there could be 80 or so fields in a single generated CSV, and there might be many related CSV files that refer to elements in each other (e.g., there may be a list of people, and hardware may have an owner who must be in the list of people), I'm inclined to create an class that encapsulates each line of the CSV as generated.

That class will store a dict mapping field names to values, then those can be passed to a `csv.DictWriter` (this is likely to be implemented in Python) that was created with a list of the fields in the correct order. The classes will be able to implement comparison operators to help map the different types, and to track internal consistency about which software is on which host and who is responsible for it and such things.

The main visceral hangup is the mapping of field names to generation rules. Using a CSV and factory seems to the only option I can come up with, but there's just something nagging me that somehow all this can be done a bit more elegantly. Can anyone help me figure if there is a cleaner, easier to extend in the future, pattern or other structure that I should be considering?

Perhaps one thing that feels nagging about this is creating the CSV templates described above. Once I have a definition of how to implement "device id" it seems like it could become painful to have to copy and paste that or retype it every time I want to make another template. Perhaps, then, I should define the rules for generating everything in a common file, then each template just provides the field names.

design-patterns

Share

Improve this question

Follow

asked May 27, 2015 at 0:49

Eric Renouf

14.5k ● 3 ● 51 ● 72

1 Answer

Sorted by: Highest score (default)

1

> The main visceral hangup is the mapping of field names to generation rules

Why not to map to rule classes directly? You can describe your fields like this (json):

```
[
    {
        "field": "text_description",
        "rule": "MarkovChainGenerator",
        "params": {
            "source": "romeo_and_julliete.txt",
            "degree": 11
        }
    },
    {
        "field": "salary",
        "rule":
"\\OtherNamespace\\SalaryGeneratory",
        "params": {
            "max": "1000.00"
            }
        }
]
```

Share Improve this answer Follow

answered May 28, 2015 at 8:54

Vladislav Rastrusny
29.9k ● 23 ● 99 ● 157

I like that, though I'm trying to decide whether that would be easier or harder for a not-to-technical person to edit. It's more explicit about what's being filled in, so that's a point for easier. A CSV can be edited in Excel though, which seems like a good marketing bullet if nothing else. I do like this though, and it does indeed feel cleaner than the CSV solution – Eric Renouf May 28, 2015 at 11:32

@EricRenouf Well, provide some GUI for editing settings ;) You can use things like this: github.com/jdorn/json-editor You can use YAML also if you like it better. – Vladislav Rastrusny May 28, 2015 at 14:29

Thanks for those pointers too, but brand name Excel has the recognition I get some value from piggy backing on. – Eric Renouf May 28, 2015 at 14:57