build Documentation

Release 1.0

Benjamin Hodgson

August 07, 2015

Contents

A tiny base class for fluent builders.

class `build.`**`Builder`**

A base class for builders, supporting the "with" style of chaining methods. "With" methods are dynamically generated based on the names defined in the 'defaults' attribute (to be overridden by subclasses) according to the template `with_{name}`.

To subclass Builder:

1. Define a class attribute (a list of tuples) called `defaults`.
2. Override the `dict` method if you need to use a custom dict class
3. Define a `build` method which performs the required steps to build the object and returns an instance of the object.

A 'with' method for each entry in `defaults` will be generated for you.

```
>>> class MyBuilder(Builder):
...     # declare the defaults for the builder
...     defaults = [
...         ("abc", 123),
...         ("def", 456),
...         ("xyz", 789)
...     ]
...
...     def build(self):
...         # convert self.data into the object you're building
...         return self.data
...
>>> result = MyBuilder().with_abc(-1).with_def(-2).build()
>>> result == {'xyz': 789, 'def': -2, 'abc': -1}
True
```

`dict`(*pairs*)

Override me if you want to use a custom *`dict`* subclass for `self.data`.

`build.`**`evaluate_callables`**(*data*)

Call any callable values in the input dictionary; return a new dictionary containing the evaluated results. Useful for lazily evaluating default values in `build` methods.

```
>>> data = {"spam": "ham", "eggs": (lambda: 123)}
>>> result = evaluate_callables(data)
>>> result == {'eggs': 123, 'spam': 'ham'}
True
```

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

b

B

D

E