Component: Application Wrappers continued

```
my_test_framework
    |- my_test_framework
        |- apps
            |- website
                - base.py
                - home.py
                - cart.py
            |- api
                - base.py
                - letters.py
        |- data
            - shirts.py
        |- framework
            - selenium_utils.py
             - utils.py
             - exceptions.py
        |- tests
             - conftest.py
             - pytest.ini
             - test_simple.py
     - requirements.txt
```

Base Object for API

```
class BaseEndpoint(object):
    """ Common ancestor for all endpoints. """
    base_url = 'http://www.colourlovers.com/api/'

    def get(self, url, ex=200, **kwargs):
       res = requests.get(url, params=kwargs)

        if not res.status_code == 200:
           raise UnexpectedStatusCodeException(response=res)

        return res

    def verify_keys_in_response(self, response_keys):
        # be strict about checking keys
        expected = self.expected_keys
        actual = list(response_keys)
        actual.sort()

        if sorted(expected) == sorted(actual):
           return True
        else:
           raise JsonPayloadException(‘Actual keys != expected keys.')
```

Component: Application Wrappers continued

You can treat an API in pretty much the same way, using an Endpoint Object Model build on top of an HTTP library like requests.

Let’s build an EOM for our pretend custom lettering API.

```
my_test_framework
   |- my_test_framework
      |- apps
         |- website
            - base.py
            - home.py
            - cart.py
         |- api
            - base.py
            - letters.py
      |- data
         - shirts.py
      |- framework
         - selenium_utils.py
          - utils.py
          - exceptions.py
      |- tests
          - conftest.py
          - pytest.ini
          - test_simple.py
   - requirements.txt
```