Specific Object for an Endpoint

```
class ColorEndpoint(base_endpoint.BaseEndpoint):

    def __init__(self):
        self.name = 'color'
         self.endpoint = 'color/'
         self.endpoint_url = self.base_url + self.endpoint
         self.expected_keys = [‘apiUrl’, ‘badgeUrl', ‘dateCreated']
        self.expected_keys.sort()


    def get_color(self, hex, format='json', verbose=True, **kwargs):
         kwargs['format'] = format
         url = self.endpoint_url + hex

         # pass to the base endpoints *requests* wrapper
         res = self.get(url, **kwargs)

         if verbose:
             logger.info('Response json: \n%s' % plog(res.json()))
         return res
```

Endpoint Objects allow greater abstraction in test cases

```
class ExampleApiTests(object):

    # create instances of endpoint
    color_endpoint = color.ColorEndpoint()

    def test_get_color(self):
       # setup: make API request
        data = '000000'
        res = self.color_endpoint.get_color(data)

        # test point: verify the correct response for a correct api call
        assert res.status_code == 200

        # test point: verify that we get back the same hex that we requested
        assert res.json()[0]['hex'] == data

        # test point: verify that the json keys are correct
        assert self.color_endpoint.verify_keys_in_response(res.json()[0].keys())
```

Specific Object for an Endpoint

```
class ColorEndpoint(base_endpoint.BaseEndpoint):

    def __init__(self):
        self.name = 'color'
         self.endpoint = 'color/'
         self.endpoint_url = self.base_url + self.endpoint
         self.expected_keys = [‘apiUrl’, ‘badgeUrl', ‘dateCreated']
        self.expected_keys.sort()


def get_color(self, hex, format='json', verbose=True, **kwargs):
    kwargs['format'] = format
    url = self.endpoint_url + hex

    # pass to the base endpoints *requests* wrapper
    res = self.get(url, **kwargs)

    if verbose:
        logger.info('Response json: \n%s' % plog(res.json()))
    return res
```