在实际工作中，为了便于维护，对于环境的切换和配置，通常不会使用硬编码的形式完成。在之前文章**《多环境下的接口测试》**中，已经介绍了如何将环境的切换作为一个可配置的选项。本文会把这部分内容进行重构，使用数据驱动的方式完成多环境的配置。

环境准备

参考**《多环境下的接口测试》**，将环境配置部分改为数据驱动的模式：
代码如下：

```
#把host修改为ip，并附加host header

env={
    "docker.testing-studio.com": {
        "dev": "127.0.0.1",
        "test": "1.1.1.2"
    },
    "default": "dev"
}
data["url"]=str(data["url"]).replace(
    "docker.testing-studio.com",
    env["docker.testing-studio.com"][env["default"]]
)
data["headers"]["Host"]="docker.testing-studio.com"

```

实战演示

依然以 YAML 为示例，将所有的环境配置信息放到 `env.yml` 文件中。如果怕出错，可以先使用 `yaml.safe_dump(env)` 将 dict 格式的代码转换为 YAML。
如下所示，打印出来的，就是成功转换 YAML 格式的配置信息：

```
def test_send(self):
    env={
        "docker.testing-studio.com": {
            "dev": "127.0.0.1",
            "test": "1.1.1.2"
        },
        "default": "dev"
    }
    yaml2 = yaml.safe_dump(env)
    print("")
```

```
    print(yaml2)

```

将打印出来的内容粘贴到 `env.yml` 文件中：

```
docker.testing-studio.com:
  dev: "127.0.0.1"
  test: "1.1.1.2"
  level: 4
default:
  "dev"

```

将环境准备中的代码稍作修改，把 `env` 变量从一个典型 dict 改为使用 `yaml.safe_load` 读取 `env.yml`：

```
# 把host修改为ip，并附加host header
env = yaml.safe_load(open("./env.yml"))
data["url"] = str(data["url"]).\
    replace("docker.testing-studio.com",
    env["docker.testing-studio.com"][env["default"]])
data["headers"]["Host"] = "docker.testing-studio.com"

```

如此一来，就可以实现使用数据驱动的方式，通过修改 `env.yml` 文件来直接修改配置信息。