数据驱动就是数据的改变从而驱动自动化测试的执行，最终引起测试结果的改变。简单来说，就是参数化的应用。数据量小的测试用例可以使用代码的参数化来实现数据驱动，数据量大的情况下建议使用一种结构化的文件（例如 YAML，JSON 等）来对数据进行存储，然后在测试用例中读取这些数据。

参数化实现数据驱动

参数化数据驱动原理与之前分享的 UI 自动化测试框架测试数据的数据驱动 大同小异。

本文依然使用 `@pytest.mark.parametrize` 装饰器来进行参数化，使用参数化来实现数据驱动。

通过参数化的方式，分别判断 id 为 2，3 的部门的 parentid 为 1：

```
import pytest

class TestDepartment:
    department = Department()

    @pytest.mark.parametrize("id", [2, 3])
    def test_department_list(self, id):
        r = self.department.list(id)
        assert self.department.jsonpath(expr="$..parentid")[0] == 1

```

上面的代码首先使用 `@pytest.mark.parametrize` 装饰器，传递了两组数据，测试结果显示有两条测试用例被执行，而不是一条测试用例。也就是 pytest 会将两组测试数据自动生成两个对应的测试用例并执行，生成两条测试结果。

YAML 文件实现数据驱动实战

当测试数据量大的情况下，可以考虑把数据存储在结构化的文件中。从文件中读取出代码中所需要格式的数据，传递到测试用例中执行。

本次实战以 YAML 进行演示。YAML 以使用动态字段进行结构化，它以数据为中心，比 Excel、csv、JSON、XML 等更适合做数据驱动。

将上面参数化的两组数据存储到 YAML 文件中，创建一个 `data/department_list.yml` 文件，代码如下：

```
- 2
- 3

```

上面的代码定义了一个 YAML 格式的数据文件 `department_list.yml` ，文件中定义了一个列表，列表中有两个数据，最后生成的是这样的数据格式：`[1,2]` 。将测试用例中参数化的数据改造成

从 `department_list.yml` 文件中读取，代码如下：

```
class TestDepartment:
    department = Department()

    @pytest.mark.parametrize("id", \
    yaml.safe_load(open("../data/department_list.yml")))
    def test_department_list(self, id):
        r = self.department.list(id)
        assert self.department.jsonpath(expr="$..parentid")[0] == 1

```

上面的代码，只需要使用 `yaml.safe_load()` 方法，读取 `department_list.yml` 文件中的数据，分别传入到用例 `test_department_list()` 方法中完成输入与结果的验证。