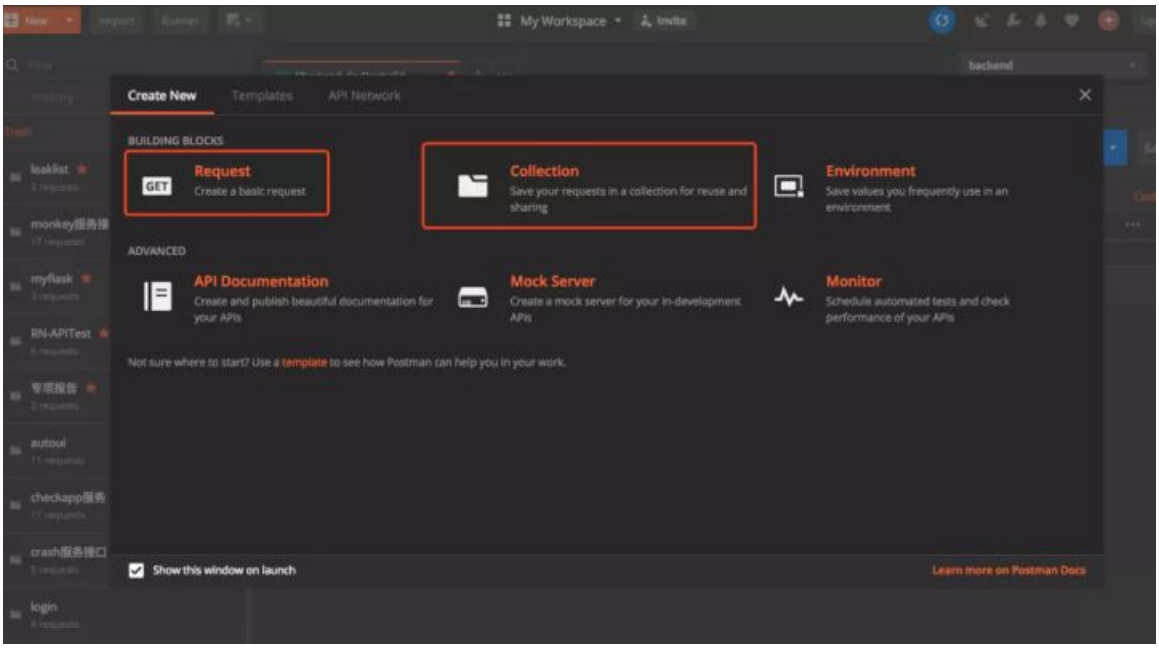
Postman 是谷歌开发的一款网页调试和接口测试工具，能够发送任何类型的 HTTP 请求，支持 GET/PUT/POST/DELETE 等方法，可以直接填写 URL，header，body 等就可以发送一个请求，非常简单易用，是接口测试必备利器。本文将详细介绍 Postman 的使用，接口测试关键步骤以及 Jenkins 持续集成。

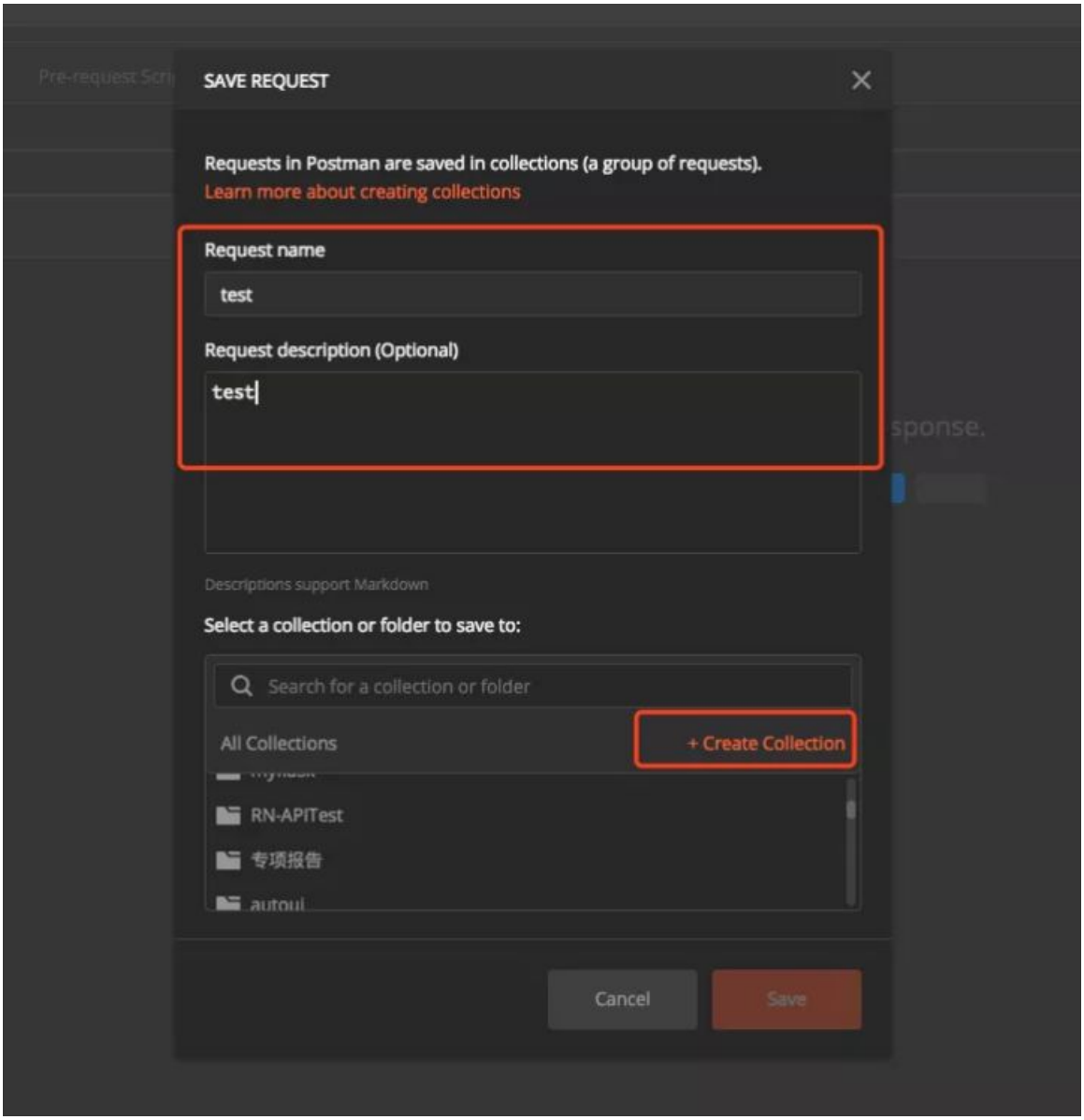
Postman 的使用

创建用例集

启动 Postman 以后，会看到这个控制面板。

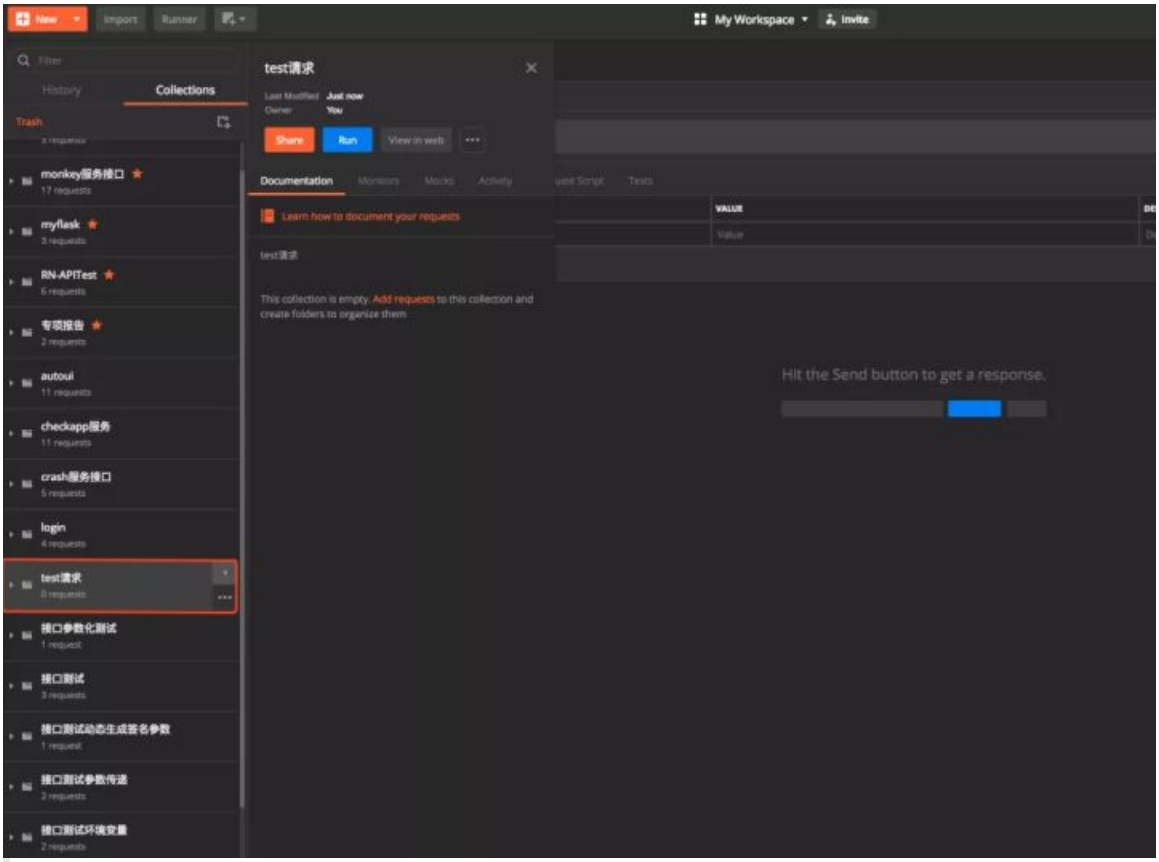


点击 Request 是创建一个 Request 测试请求，但是需要创建用例集保存这个请求。



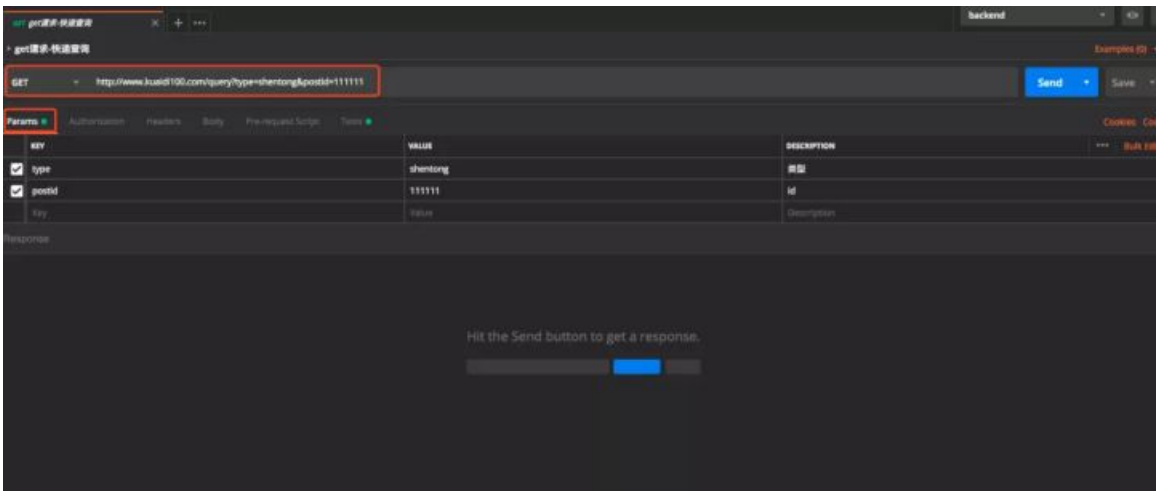
点击 Collection 是创建一个用例集来保存测试请求。

创建 Collection 完成后，会在左侧生成用例集文件架，每次创建的测试接口都要保存到用例集中。

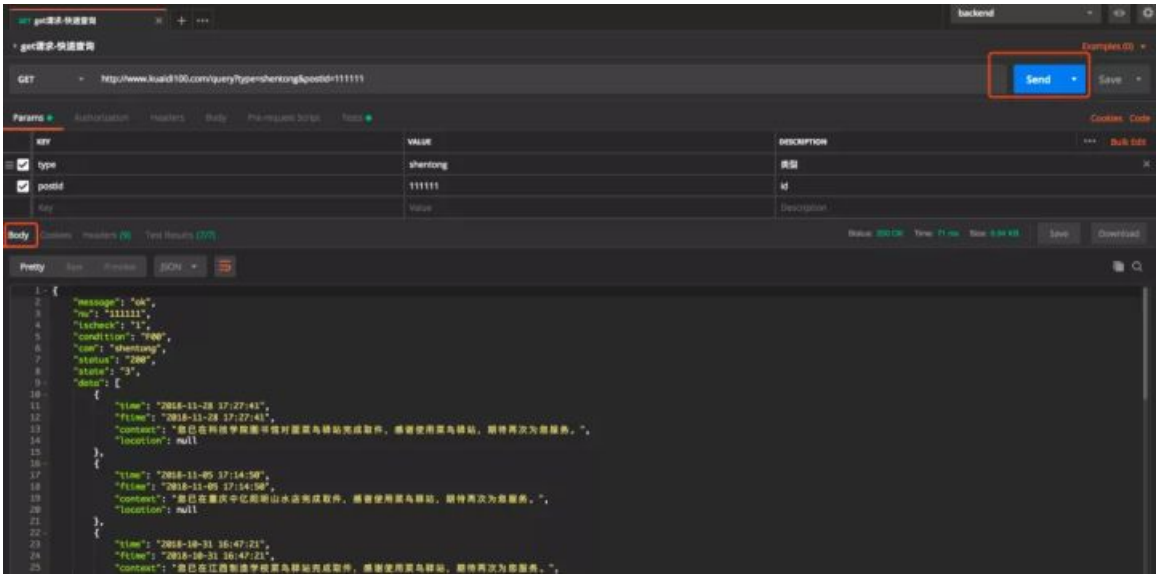


第一个接口测试

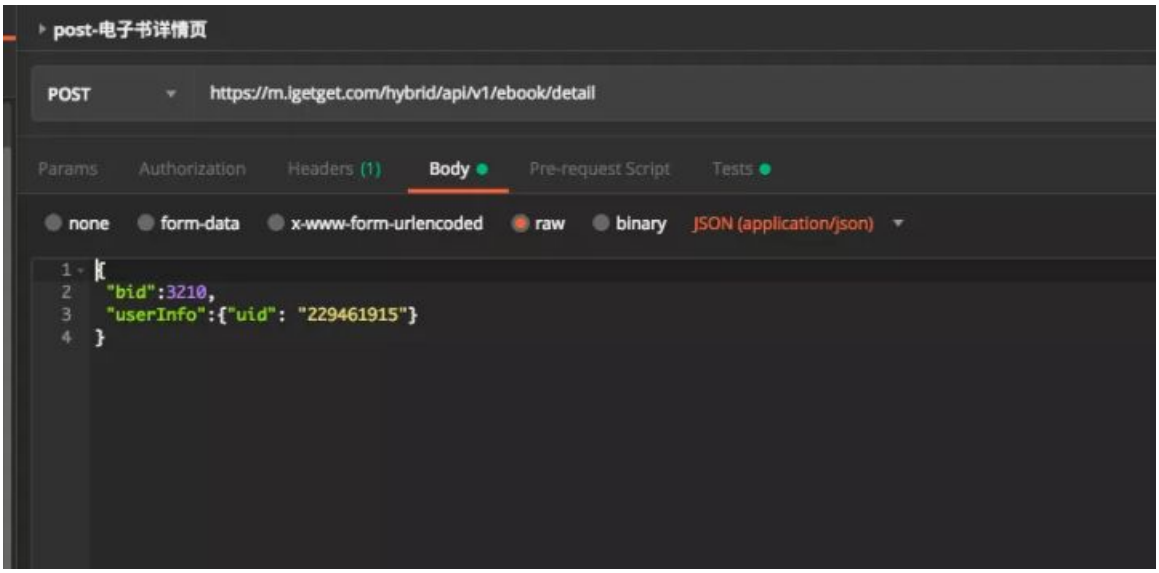
创建 get 请求为例，通常需要写 url、params、headers，会把 params 拼接到 url 末尾。



点击 send 按钮并且请求成功，会展示响应结果。

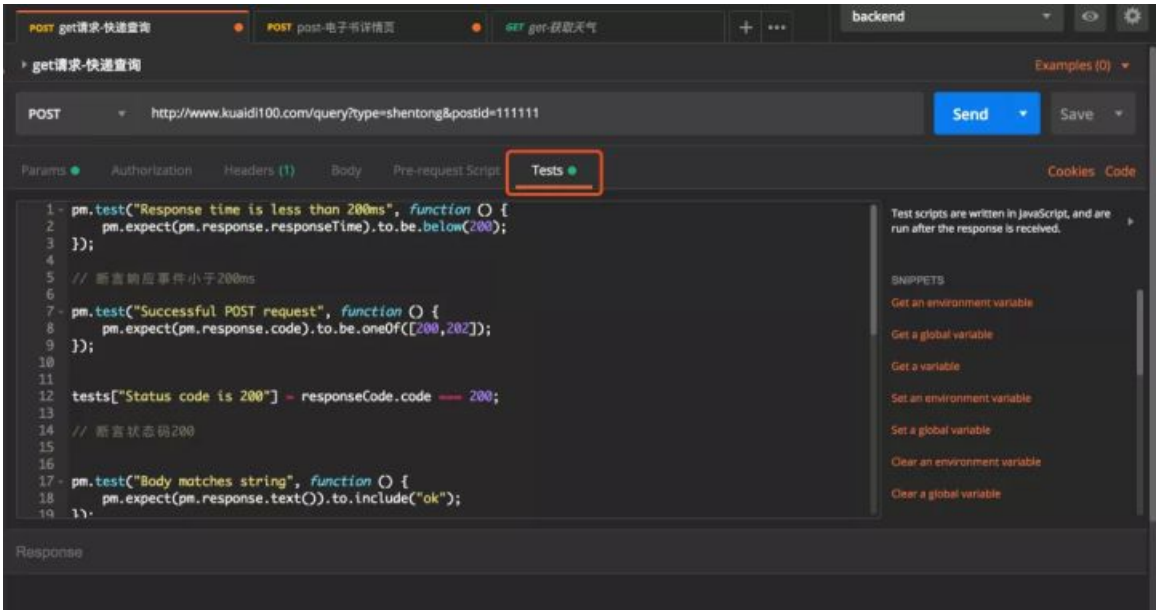


创建 post 请求为例，通常需要写 url、body、headers 等参数，body 参数格式一般是 form 或者 json 格式。具体 body 使用那个格式，需要按照接口文件中的参数。



接口断言

点击 Tests 编写测试断言



断言响应时间

```
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});
// 断言响应事件小于 200ms

```

断言状态码

```
pm.test("Successful POST request", function () {
    pm.expect(pm.response.code).to.be.oneOf([200,202]);
});
// 断言状态码 200-202 区间

```

断言响应中包含某个字符串

```
pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include("ok");
});

// 断言响应中包含 "ok"

```

断言响应中的字段等于某个值

```
pm.test("message test", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData["message"]).to.eql("ok");
});

// 断言响应中 "message" = ok"

```

断言响应中的字段不等于某个值

```
var jsonData = JSON.parse(responseBody);
tests["message 不为 bad"] = jsonData["message"] != "bad";

// 断言响应中 "message" != bad"

```

断言响应中的列表长度

```
pm.test("data list test", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData["data"].length).to.eql(41);
});

// 断言响应中 "list" 的字段长度

```

断言响应中的列表中第几个元素的字段值

```
pm.test("data list 0 test", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData["data"][0]["time"]).to.eql("2018-11-28 17:27:41");
});

// 断言响应中 "list 0 的 " 的 time 字段的值

```

Json schema 验证

tv4 是 postman 内置的 JSON Schema 验证库 , 参考:https://geraintluff.github.io/tv4/

responseBody 如下 ==:==

```
{
    "errCode": 0,
    "errMsg": "",
    "data": {
        "id": 3210,
        "title": "",

const customerSchema = {
```

```
  "type": "object",
  "properties": {
    "errCode": {
        "type": "integer",
        "minimum": 0,
        "maximum": 3,
        "minLength": 2,
        "maxLength": 3
    },
    "errMsg": {"type": "string"},
  }
};

var customer = JSON.parse(responseBody);
// console.log(customer);
tests["Valid Data1"] = tv4.validate(customer, customerSchema);
// 验证 json 中的 errCode 类型是 integer, 并且验证最小值和最大值区间、验证长度区间
```