全量字段校验

接口自动化测试课程

学习目标

Learning Objectives

1. 掌握如何对接口响应数据进行全量字段校验

目录

Contents

全量字段校验

01

- 接口校验思考
- 校验目标
- 实现思路

学习目标

1. 理解全量字段校验的目标

全量字段校验

接口校验思考

问题一：接口响应校验哪些字段?

答：客户端或其他服务使用的字段

问题二：需要校验字段的哪些内容?

答：字段名/字段值或字段类型

全量字段校验

全量字段校验

实现思路

1.定义接口返回数据的期望格式

2.与接口实际响应数据对比校验

目录

Contents

JSON Schema

02

- 快速入门
- JSON Schema关键字
- 综合案例

学习目标

1. 掌握JSON Schema关键字
2. 根据JSON数据的要求定义JSON Schema

JSON Schema介绍

Json Schema: 用来定义json数据约束的一个标准

入门案例

待校验的数据

JSON数据

```
{
    "success": true,
    "code": 10000,
    "message": "操作成功"
}
```

校验规则描述

- 整个JSON数据是一个对象
- 包含success、code、message字段，并且是必须存在的字段
- success字段为布尔类型
- code为整数
- message为字符串

JSON Schema介绍

快速入门

JSON Schema

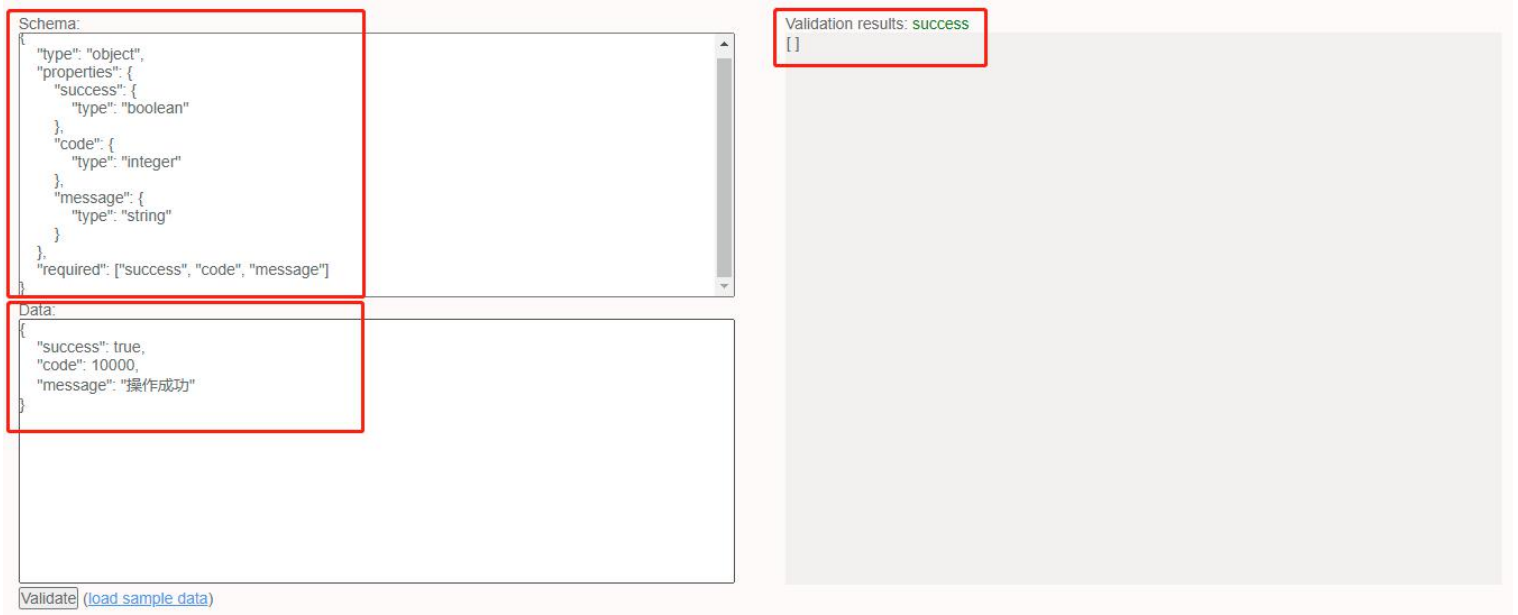
```
{
    "type": "object",
    "properties": {
        "success": {
            "type": "boolean"
        },
        "code": {
            "type": "integer"
        },
        "message": {
            "type": "string"
        }
    },
    "required": ["success", "code", "message"]
}
```

注意：待校验JSON对象的所有字段分别都通过对应的JSON Schema的校验检测，这个对象才算是通过校验！

JSON Schema介绍

json schema在线校验工具：

- http://json-schema-validator.herokuapp.com
- https://www.jsonschemavalidator.net



JSON Schema重点关键字

关键字	描述
type	表示待校验元素的类型
properties	定义待校验的JSON对象中，各个key-value对中value的限制条件
required	定义待校验的JSON对象中，必须存在的key
const	JSON元素必须等于指定的内容
pattern	使用正则表达式约束字符串类型数据

JSON Schema关键字详解

type：用于限定待校验JSON元素所属的数据类型

type取值	对应的python数据类型	描述
object	object	对象
array	list	数组
integer	int	整数
number	float或int	数字
null	None	空
boolean	bool	布尔
string	str	字符串

JSON Schema关键字详解

properties：定义待校验的JSON对象中包含的字段

说明：

- 该关键字的值是一个对象
- 当type取值为object时使用
- 用于指定JSON对象中的各种不同key应该满足的校验逻辑

案例

JSON Schema关键字–type

需求：

1. 已知JSON数据
2. 要求定义每个一级字段的数据类型

```
{
    "success": true,
    "code": 10000,
    "message": "操作成功",
    "money": 6.66,
    "address": null,
    "data": {
        "name": "tom"
    },
    "luckyNumber": [6, 8, 9]
}
```

案例

JSON Schema关键字详解–type

定义JSON Schema：

```
{
    "type": "object",
    "properties": {
        "success": {
            "type": "boolean"
        },
        "code": {
            "type": "integer"
        },
        "message": {
            "type": "string"
        },
        "money": {
            "type": "number"
        },
        "address": {
            "type": "null"
        },
        "data": {
            "type": "object"
        },
        "luckyNumber": {
            "type": "array"
        }
    }
}
```

案例 JSON Schema关键字–properties

需求：

1. 已知JSON数据
2. 要求定义JSON对象中包含的所有字段及数据类型

```
{
    "success": true,
    "code": 10000,
    "message": "操作成功",
    "data": {
        "name": "tom",
        "age": 18
    }
}
```

案例

JSON Schema关键字–properties


```
{
    "type": "object",
    "properties": {
        "success": {"type": "boolean"},
        "code": {"type": "integer"},
        "message": {"type": "string"},
        "data": {
            "type": "object",
            "properies": {
                "name": {"type": "string"},
                "age": {"type": "integer"}
            }
        }
    }
}
```

JSON Schema关键字详解

required：定义待校验的JSON对象中，必须存在的key

说明：

- 用于限制JSON对象中必须包含哪些key
- 该关键字的值是一个数组，而数组中的元素必须是字符串，而且必须是唯一的

案例

JSON Schema关键字–required

需求：

1. 已知JSON数据
2. 要求JSON对象中必须包含success、code、message字段

```
{
    "success": true,
    "code": 10000,
    "message": "操作成功",
    "data": null
}
```

案例 JSON Schema关键字–required


```
{
    "type": "object",
    "required": ["success", "code", "message"]
}
```

JSON Schema关键字详解

const：用于校验JSON元素必须等于指定的内容

说明：

- 如果待校验的JSON元素的值和该关键字指定的值相同，则通过校验。否则，无法通过校验
- 该关键字的值可以是任何值，包括null

案例

JSON Schema关键字–const

需求：

1. 已知JSON数据
2. 校验code字段的值必须等于10000

```
{
    "code": 10000
}
```

案例

JSON Schema关键字–const


```
{
    "type": "object",
    "properties": {
        "code": {
            "const": 10000
        }
    }
}
```

JSON Schema关键字详解

pattern：正则表达式

说明：

- 正则表达式：字符串的匹配模式
- 包含指定字符串：pattern =“指定字符串”

案例 JSON Schema关键字–pattern

需求：

1. 已知JSON数据
2. 校验message字段的值必须包含‘操作成功’
3. 校验手机号必须是11为数字

```
{
    "message": "操作成功!",
    "mobile": "13800000002"
}
```

案例

JSON Schema关键字–pattern


```
{
    "type": "object",
    "properties": {
        "message": {
            "pattern": "操作成功"
        },
        "mobile": {
            "pattern": "^[0–9]{11}$"
        }
    }
}
```

JSON Schema综合案例


```
{
    "success": false,
    "code": 10000,
    "message": "xxx登录成功",
    "data": {
        "age": 20,
        "name": "lily"
    }
}
```

校验规则描述

需求：

- 整个JSON数据是一个对象
- 包含success、code、message、data字段，并且是必须存在的字段
- success字段为布尔类型
- code为整数
- message为以“登录成功”结尾的字符串
- data为对象，必须包含name、age字段
- data中的name字段必须等于lily
- data中的age必须是20

JSON Schema综合案例

JSON Schema

```
{
    "type": "object",
    "properties": {
        "success": {"type": "boolean"},
        "code": {"type": "integer"},
        "message": {
            "pattern": "登录成功$"
        },
        "data": {
            "type": "object",
            "properties": {
                "name": {"const": "lily"},
                "age": {"const": 20}
            }
        }
    }
}
```

1. JSON Schema重点关键字有哪些?
2. 如何根据JSON数据的规范要求定义JSON Schema?

目录

Contents

python实现JSON数据校验

- jsonschema介绍和安装
- 使用方式

学习目标

1. 掌握如何使用jsonschema校验JSON数据

jsonschema实现JSON数据校验

jsonschema介绍和安装

jsonschema：是Python语言中实现对JSON Schema校验的模块

```
pip install jsonschema
```

jsonschema实现JSON数据校验

jsonschema使用方式

```
jsonschema.validate(instance, schema)
    """
        instance：要验证的JSON数据
        schema：用于校验JSON数据的验证规则
    """
```

说明：

- 该函数首先验证所提供的schema本身是否有效，然后再验证json数据
- 如果schema模式本身是无效的，则抛出 jsonschema.exceptions.SchemaError 异常
- 如果JSON数据校验不通过，则抛出 jsonschema.exceptions.ValidationError 异常

案例

jsonschema使用案例

需求：测试ihrm登录接口–登录成功

要求：全量校验接口返回结果（data字段为字符串，其他字段进行值匹配）

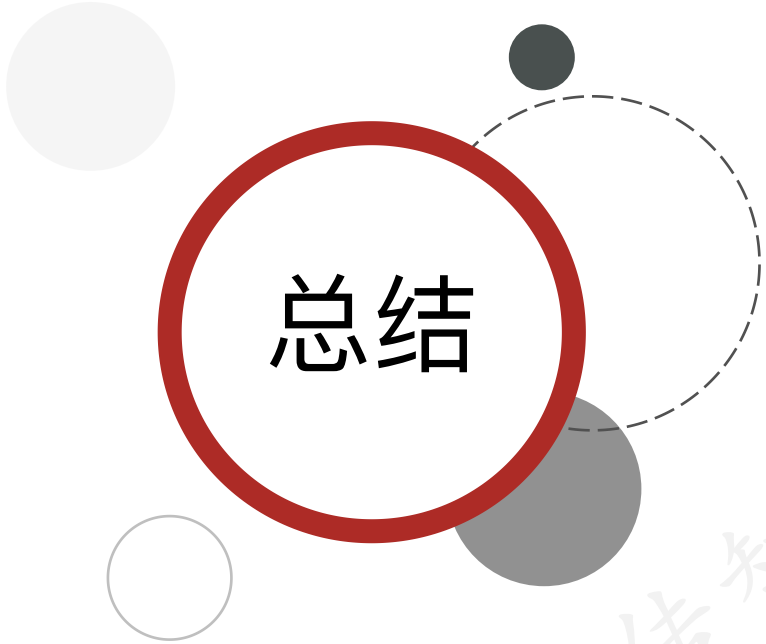
jsonschema实现JSON数据校验

案例

```
import jsonschema, requests, unittest
class TestJsonSchema(unittest.TestCase):
    def test01(self):
        url = "http://ihrm-test.itheima.net/api/sys/login"
        r = requests.post(url, json={"mobile": "13800000002", "password": "123456"})
        json_data = r.json()

        # 编写schema：
        my_schema = {
            "type": "object",
            "properties": {
                "success": { "const": True },
                "code": { "const": 10000 },
                "message": {    "const": "操作成功！"     },
                "data": {     "type": "string"   }
            },
            "required": [ "success", "code", "message", "data"  ]
        }

        # 验证：
        jsonschema.validate(instance=json_data, schema=my_schema)
```



1. 掌握如何对接口响应数据进行全量字段校验

传智教育旗下高端IT教育品牌