你为什么需要mock-service

前后端分离已成为互联网项目开发的业界标准使用方式

传统的模式

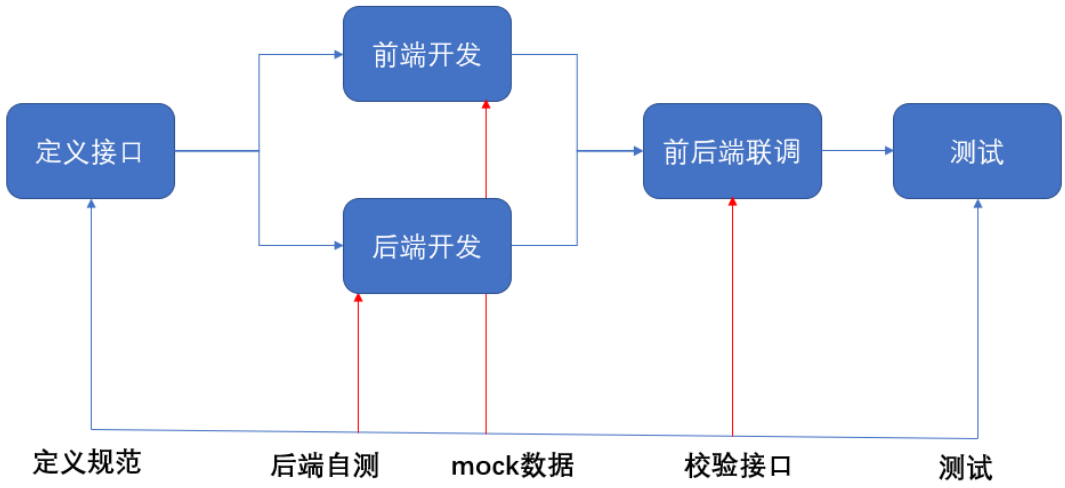
前端和后端进行调试，修改都非常麻烦。往往前端配合后端很痛苦，后端也嫌前端麻烦。

前后端分离

前后端分离并不只是开发模式，而是一种架构模式。

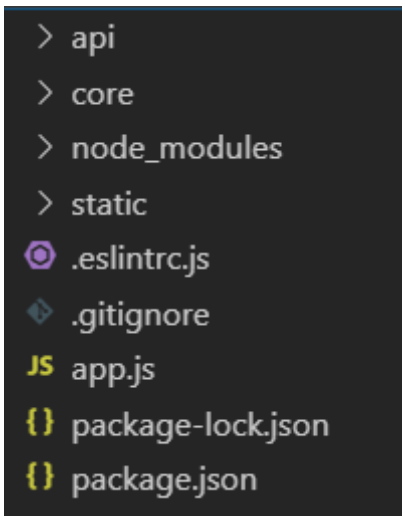
在开发阶段，前后端工程师约定好数据交互接口，实现并行开发和测试；在运行阶段前后端分离模式需要对web应用进行分离部署，前后端之前使用HTTP或者其他协议进行交互请求。

具体流程如下



基本了解了就不多说了

下面来讲讲怎么使用mock-service



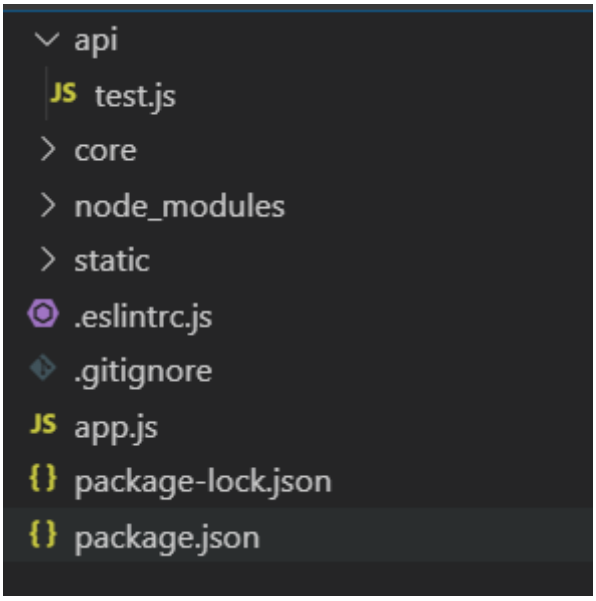
这是目录结构

我们需要用到的目录有两个

下面我们先讲讲 `api` 文件夹的功能

定义接口直接在 `api` 文件夹里面写就可以了

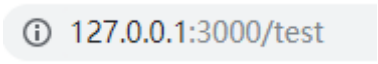
创建 `test.js` 文件




```
api > JS test.js > ...
1  module.exports = {
2    type: 'get',
3    mock: {
4      'list|10-20': [{
5        'id|+1': 1,
6        title: '时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”',
7        desc: /京雄城际铁路创造的世界、中国第一|世界第一条时速350公里列车下穿航站 楼的高铁{1,2}/,
8        'isShow|1': false
9      }],
10     msg: 'ok',
11     code: 0
12   }
13 }
14
```

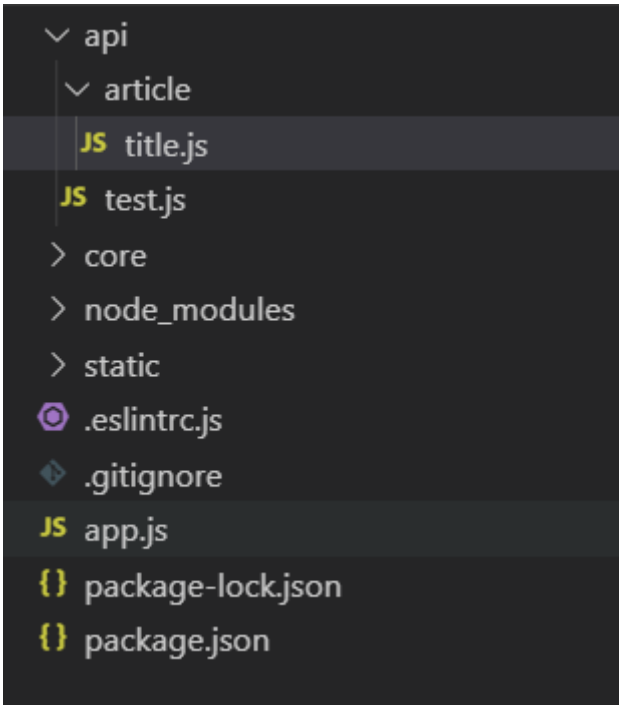
执行 `npm run serve`

在浏览器输入 `http://127.0.0.1:3000/test`



```
{
  "list": [
    {
      "id": 1,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "世界第一条时速350公里列车下穿航站 楼的高铁",
      "isShow": true
    },
    {
      "id": 2,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 3,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "世界第一条时速350公里列车下穿航站 楼的高铁铁",
      "isShow": true
    },
    {
      "id": 4,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 5,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 6,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 7,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    }
```

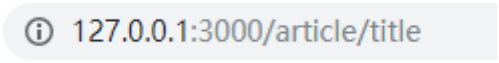

`url` 为什么是 `test`，因为你创建的文件夹或者文件会成为 `url` 路径，可能说的不是很清楚，没关系，在创建一个文件夹 `article` ，在文件夹下创建 `title.js` 文件



内容就随便点和上面一样

```
module.exports = {
  type: 'get',
  mock: {
    'list|10-20': [{
      'id|+1': 1,
      title: '时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”',
      desc: /京雄城际铁路创造的世界、中国第一|世界第一条时速350公里列车下穿航站 楼的高铁{1,2}/,
      'isShow|1': false
    }],
    msg: 'ok',
    code: 0
  }
}
```

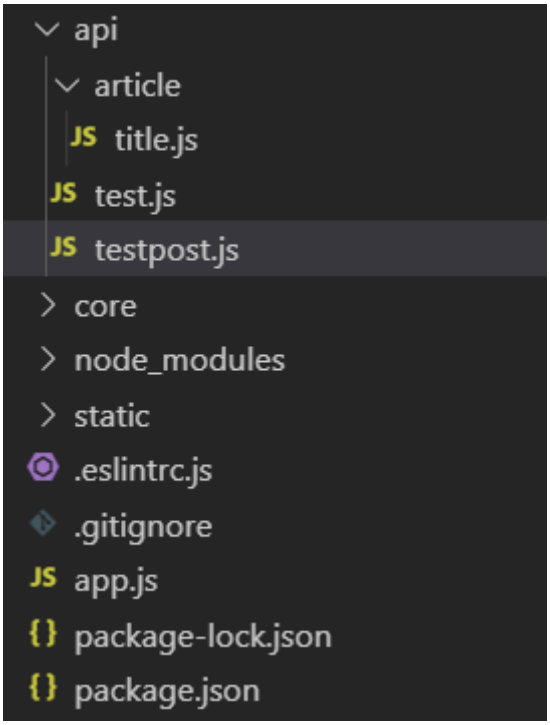
下面我们就很自然的在浏览器输入 `http://127.0.0.1:3000/article/title`



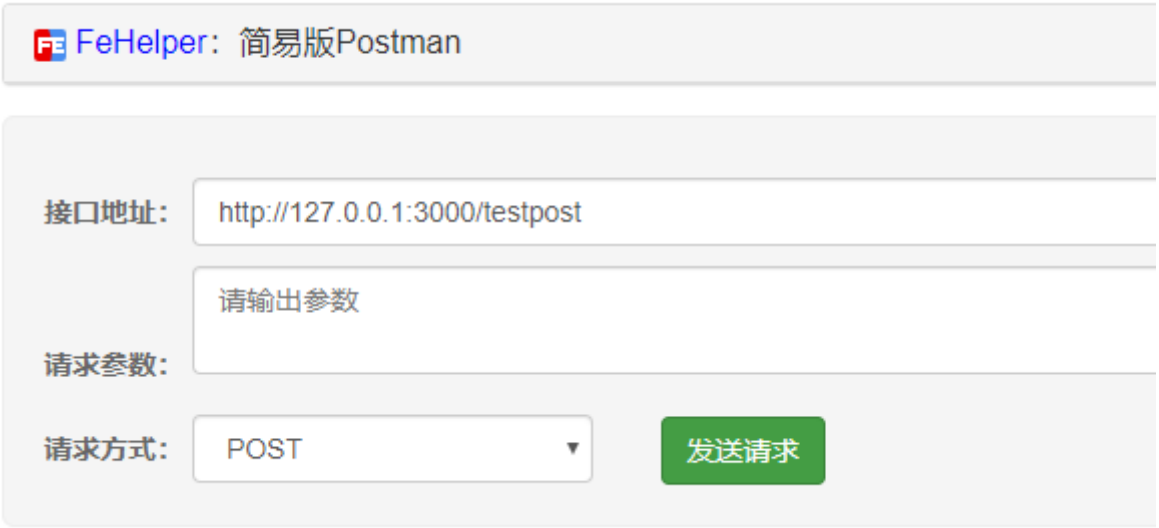
```
{
  "list": [
    {
      "id": 31,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 32,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "世界第一条时速350公里列车下穿航站 楼的高铁",
      "isShow": false
    },
    {
      "id": 33,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 34,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": false
    },
    {
      "id": 35,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "京雄城际铁路创造的世界、中国第一",
      "isShow": true
    },
    {
      "id": 36,
      "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
      "desc": "世界第一条时速350公里列车下穿航站 楼的高铁铁",
      "isShow": true
    },
    {
      "id": 37,
```


上面的都是 `get` 请求的，下面我们来个 `post` 请求的

在 `api` 文件夹下创建 `testpost.js` 文件



使用这个 `chrome` 插件



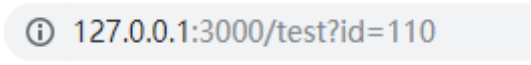
```
{
  "list": [
    {
      "id": 1
    },
    {
      "id": 2
    },
    {
      "id": 3
    },
    {
      "id": 4
    },
    {
      "id": 5
    },
    {
      "id": 6
    },
    {
      "id": 7
    }
  ],
  "msg": "ok",
  "code": 0
}
```


上面常用的请求都讲到了， 其实改变 `type` 属性的值就可以了

基本操作可以了，但是我们想处理请求参数怎么办，说再多也无用，还不如图文实在

直接修改 `test.js` 文件

```
module.exports = {
  callback: (req) => {
    const id = req.query.id
    return {
      id: id,
      title: '时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”',
      'desc|1-3': [`京雄城际铁路创造的世界、中国第一    我的id是 => ${id}`],
      'isShow|1': false,
      msg: 'ok',
      code: 0
    }
  }
}
```

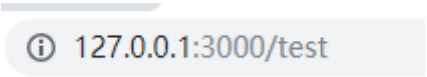


```
{
    "id": "110",
    "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
    "desc": [
        "京雄城际铁路创造的世界、中国第一 我的id是 => 110",
        "京雄城际铁路创造的世界、中国第一 我的id是 => 110"
    ],
    "isShow": false,
    "msg": "ok",
    "code": 0
}
```


大家可能奇怪，为什么没有 `type` 和 `mock` 选项了，其实 `type` 默认就是 `get` 请求，所以可以不用写，

那`mock` 选项是怎么回事，因为你要处理参数， `mock` 选项也不会使用到，为了简便所以也可以不用写了，

实际上面的也可以取代 不带参数的请求，可能没说清楚，还是来张图实在




```
{
    "title": "时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”",
    "desc": [
        "京雄城际铁路创造的世界、中国第一 我的id是 => undefined",
        "京雄城际铁路创造的世界、中国第一 我的id是 => undefined"
    ],
    "isShow": false,
    "msg": "ok",
    "code": 0
}
```


大家可能注意到上面的没有 `id` 这个选项，这是 `mockjs` 的机制，这样可以在判断用户使用是否传了参数，做对应的处理，或给默认值。

`callback` 选项是函数， `req` 是参数

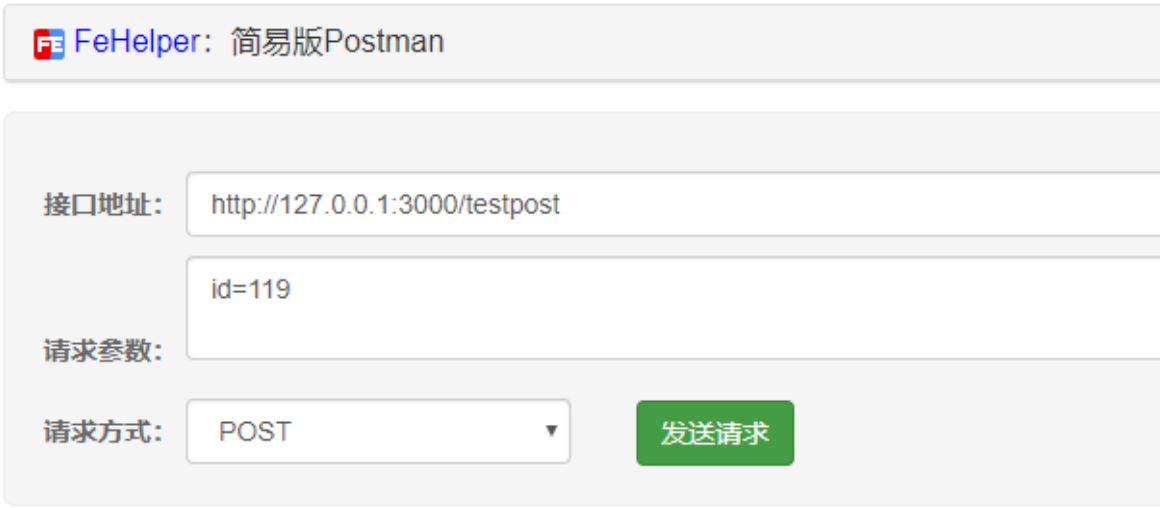
`get` 请求的参数通过 `req.query` 来获取

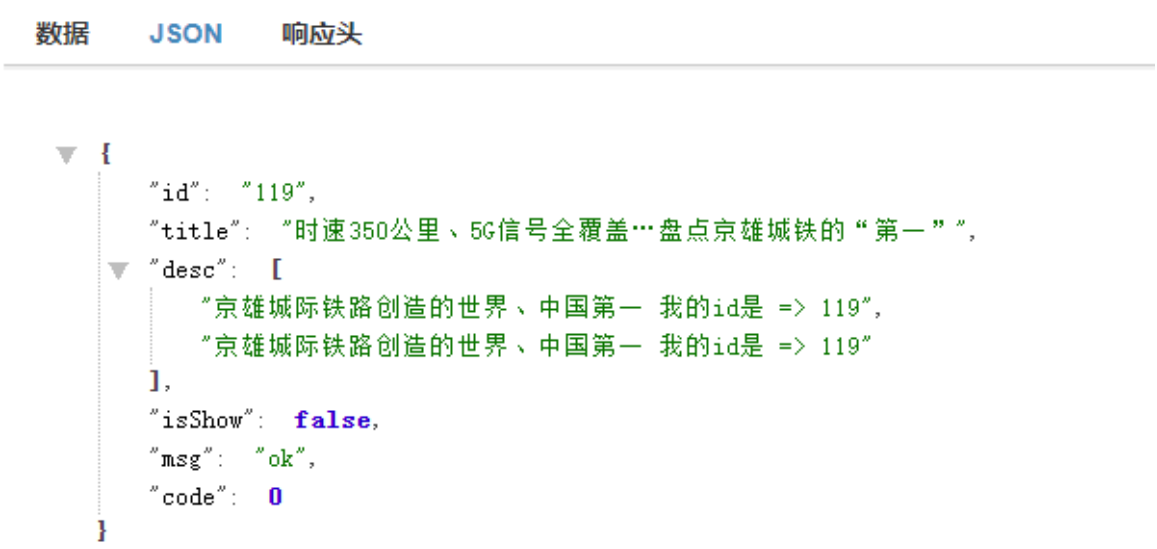
`post` 请求的参数通过 `req.body` 来获取

修改 `testpost.js` 文件

api > testpost.js > <unknown> > callback > id

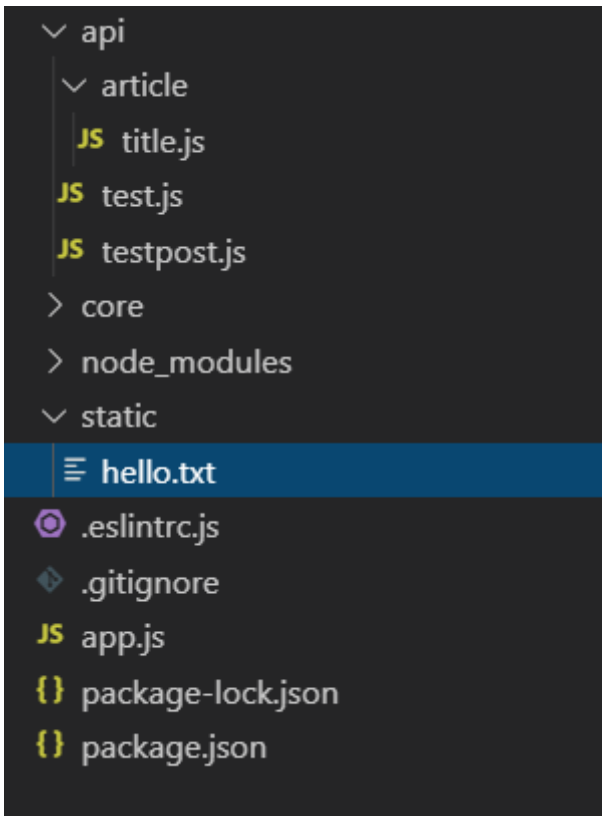
```
module.exports = {
  type: 'post',
  callback: (req) => {
    const id = req.body.id
    return {
      id: id,
      title: '时速350公里、5G信号全覆盖…盘点京雄城铁的“第一”',
      'desc|1-3': [`京雄城际铁路创造的世界、中国第一     我的id是 => ${id}`],
      'isShow|1': false,
      msg: 'ok',
      code: 0
    }
  }
}
```

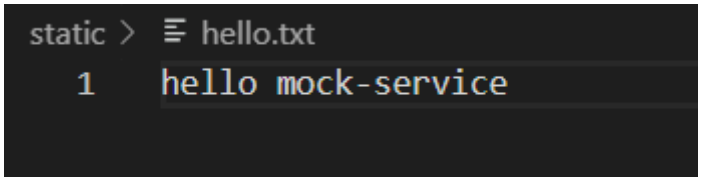




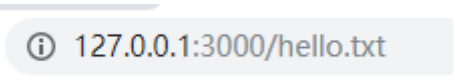
下面我们来看看 `static` 文件夹有什么功能

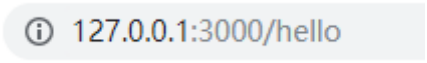
在 `static` 文件夹下创建 `hello.txt` 文件





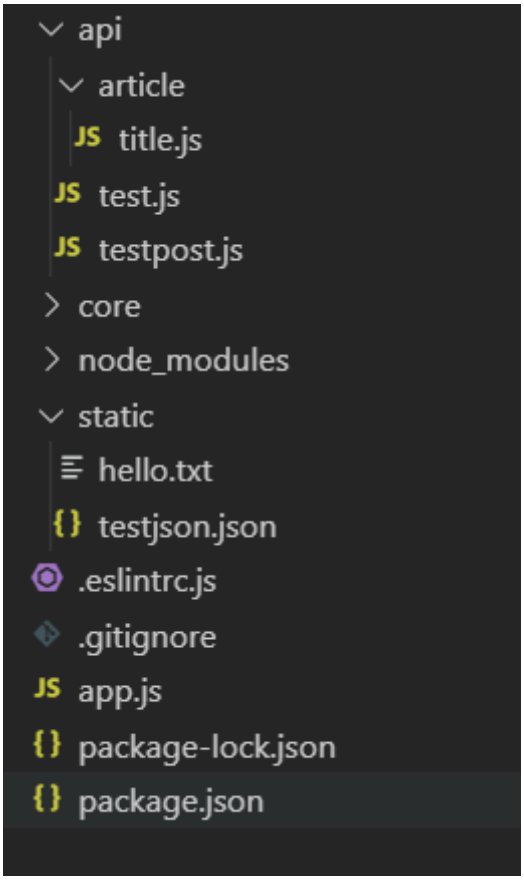
下面直接在浏览器输入 `http://127.0.0.1:3000/hello.txt` 或者 `http://127.0.0.1:3000/hello`

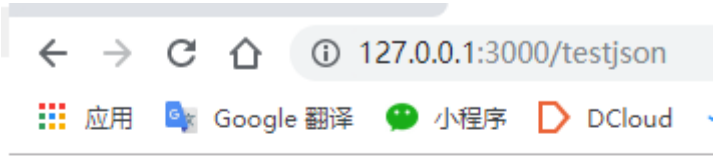


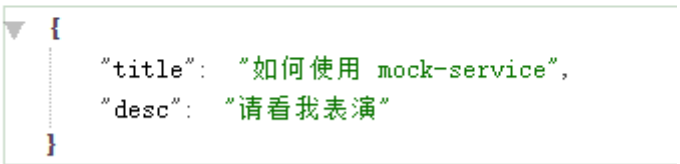


hello mock-service

在 `static` 文件夹下创建 `testjson.json` 文件

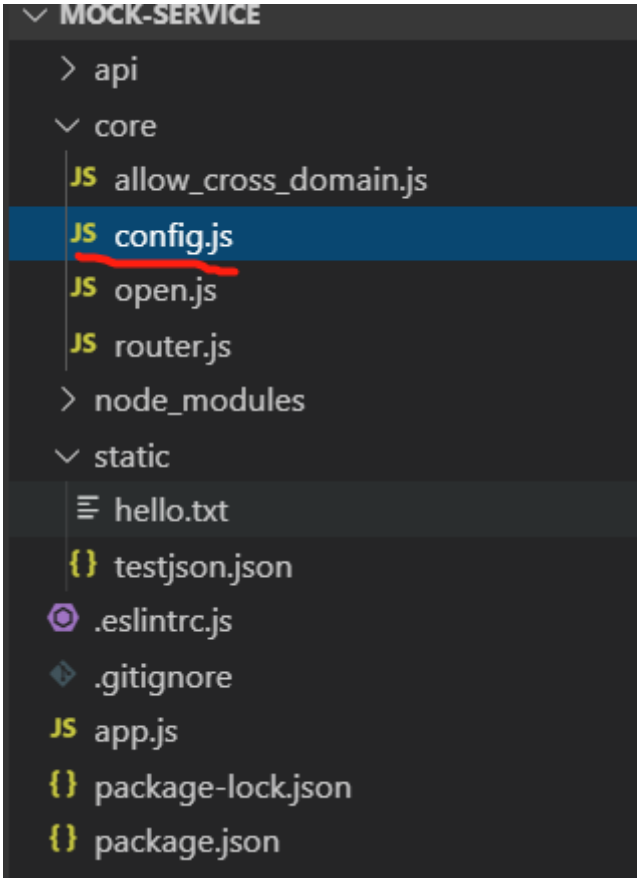






默认只支持 `.txt` `.json`

如果有其他需要可以在这个文件夹修改




```
core > JS config.js > [@] <unknown>
1   module.exports = {
2     // 静态文件是否需要后缀访问
3     extensions: ['json', 'txt']
4   }
5
```

到这里就完结了

项目地址 `https://github.com/134355/mock-service.git`

“如果觉得不错的话”

LJW的赞赏码