

第一章 接口测试的基础

在说接口测试之前，先重温两个概念，前端和后端。对于 web 端来说，前端就是我们所使用的网页，打开的网站，这都是前端，通过 html、css 等技术开发；对于移动端来说，就是我们所使用的 app，它的作用就是显示页面，以及做一些简单的校验，比如说非空校验。

而业务逻辑如打车、系统派单给司机这些功能是后端通过代码编程实现的，前端和后端是如何交互的？就是通过接口。

接口其实就是前端页面或 APP 等调用与后端做交互用的，简单来讲接口测试就是通过测试不同情况下的输入参数与之相应的返回结果，来判断接口是否符合或满足相应的功能性、安全性要求。

1.1.http 工作原理

在接口测试中，模拟浏览器发送 request 至服务器返回 response，既然要做接口测试，我们首先需要了解数据的传输过程。

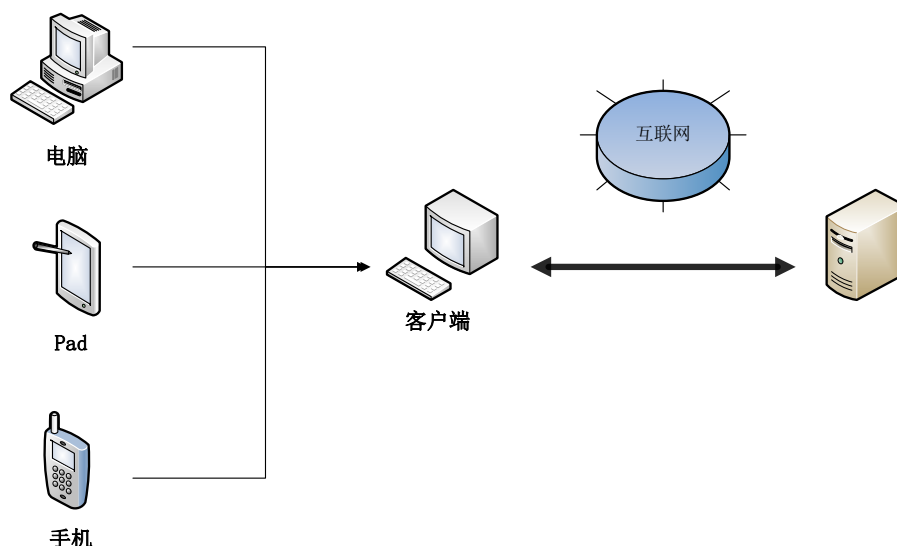


图 6-1 HTTP 协议原理

HTTP 协议工作于客户端—服务端的架构上，客户端通过 URL 向服务器发送请求，服务器根据接收到的请求，向客户端发送响应信息。

图 6-1 中有以下几点需要我们理解。

客户端主要有两个职能。

- (1) 向服务器发送请求
- (2) 接收服务器返回的报文并解释成友善的信息供我们阅读。

客户端大概有以下几类：浏览器、应用程序（桌面应用和 app 应用）等，我们在日常生活中使用比较频繁的就是浏览器。下面我们以浏览器为例，如图 6-2 所示。



图 6-2 chrome 浏览器访问网页

我们在地址栏输入网址并回车，浏览器会为我们做如下的处理：

- 当我们在浏览器输入 `www.baidu.com` 的时候，浏览器发送一个 Request 请求去获取 `www.baidu.com` 的 html 文件，服务器把 Response 文件对象发送回给浏览器。
- 浏览器分析 Response 中的 HTML，发现其中引用了很多其他文件，比如 Images 文件，CSS 文件，JS 文件。浏览器会自动再次发送 Request 去获取图片，CSS 文件，或者 JS 文件。
- 当所有的文件都下载成功后，网页会根据 HTML 语法结构，完整的显示出来了。

HTTP 协议详细规定了客户端与服务器之间互相通讯的规则，主要解决了两

个问题：

- 如何定位资源资源？ ---URL
- 客户端与服务器间如何进行信息传递？ ----报文

1.2. 用 firefox 浏览器抓取报文

测试人员可以使用 firefox 浏览器抓取 http 请求报文，以加深对通讯过程及 http 协议的理解，你可以通过以下步骤来抓取数据报文。

(1) 打开 firefox 浏览器，选择菜单栏【工具】→【Web 开发者】→【切换工具箱】，如图 6-3 所示。

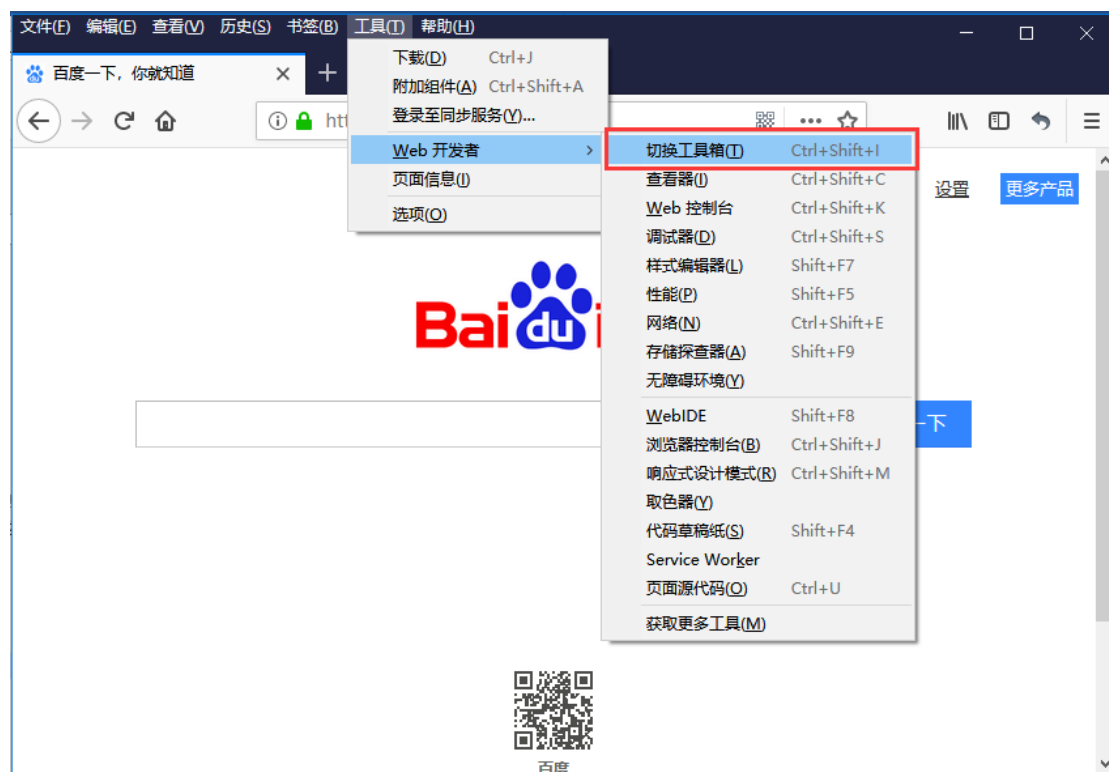


图 6-3 firefox 浏览器开启开发者工具

(2) 在浏览器的下方，将显示开发者工具窗口，如图 6-4 所示。



图 6-4 firefox 浏览器开发者工具展示

(3) 在开发者工具栏，切换至【网络】页签，如图 6-5 所示。



图 6-5 切换至网络页签

(4) 在地址栏中输入 www.baidu.com，并按回车键之后，浏览器发送一个 Request 请求去获取 www.baidu.com 的 html 文件如图 6-5 所示。

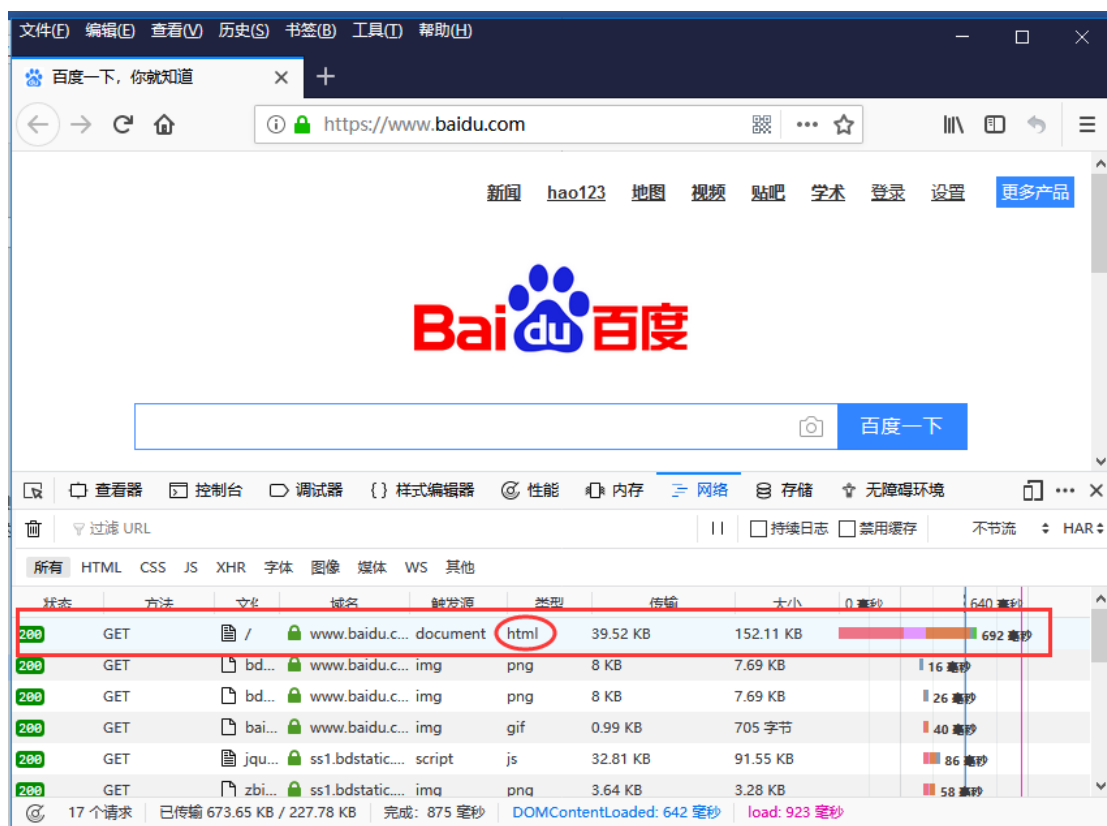


图 6-5 http 请求

在上图中，第一个请求就是主请求，也就是 www.baidu.com 的 HTML 文件，浏览器会分析 Response 中的 HTML，发现其中引用了很多其他文件，比如 Images 文件，CSS 文件，JS 文件。浏览器会自动再次发送 Request 去获取图片，CSS 文件，或者 JS 文件，所以大家可以看到后面的请求是 js、图片等资源。

(5) 选中第一个主请求，我们就可以看到请求的消息头，如图 6-6 所示。

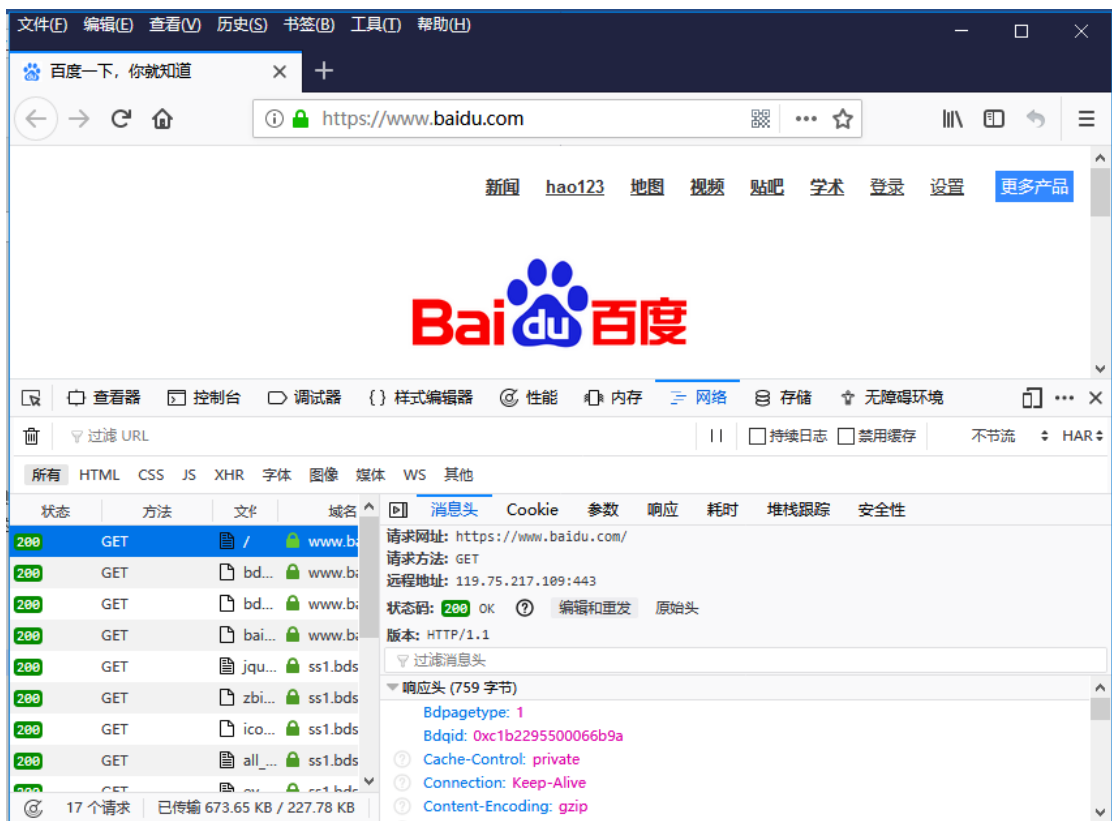


图 6-6 请求的消息头

(6) 点击“编辑和重发”，就可以看到完整的 http 请求报文，如图 6-7 所示。



图 6-7 完整 http 请求

在图中，你可以发现请求报文的结构分为 3 个部分：请求行、请求头、请求主体。

(7) 在上图中点击“取消”，我们来看响应报文，在 消息头部分，可以看到状态码和响应头，如图 6-8 所示。

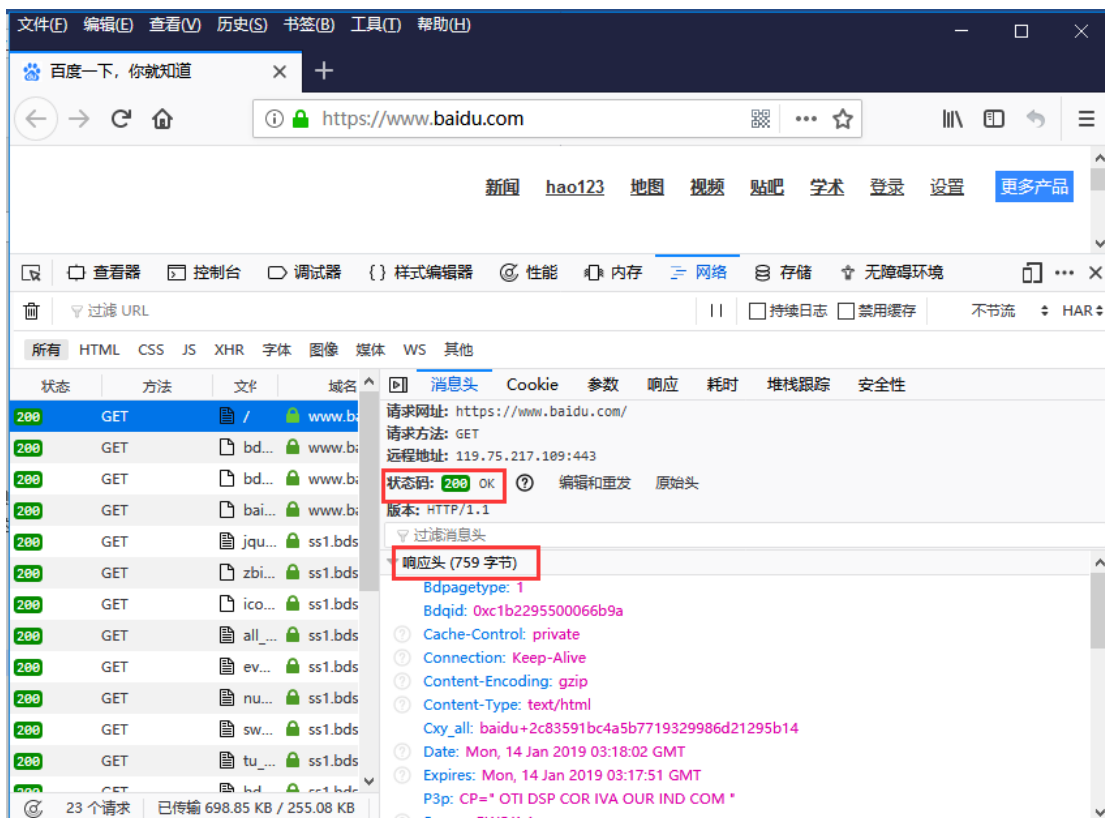


图 6-8 响应头

在图中，我们可以看到服务器返回的响应头：状态码为 200，请求成功，响应报文的结构分为 3 个部分：状态行、响应头、响应主体。

(7) 在图 6-8 中，点击“响应”，就可以看到服务器返回的 HTML 源码，如图 6-9 所示。

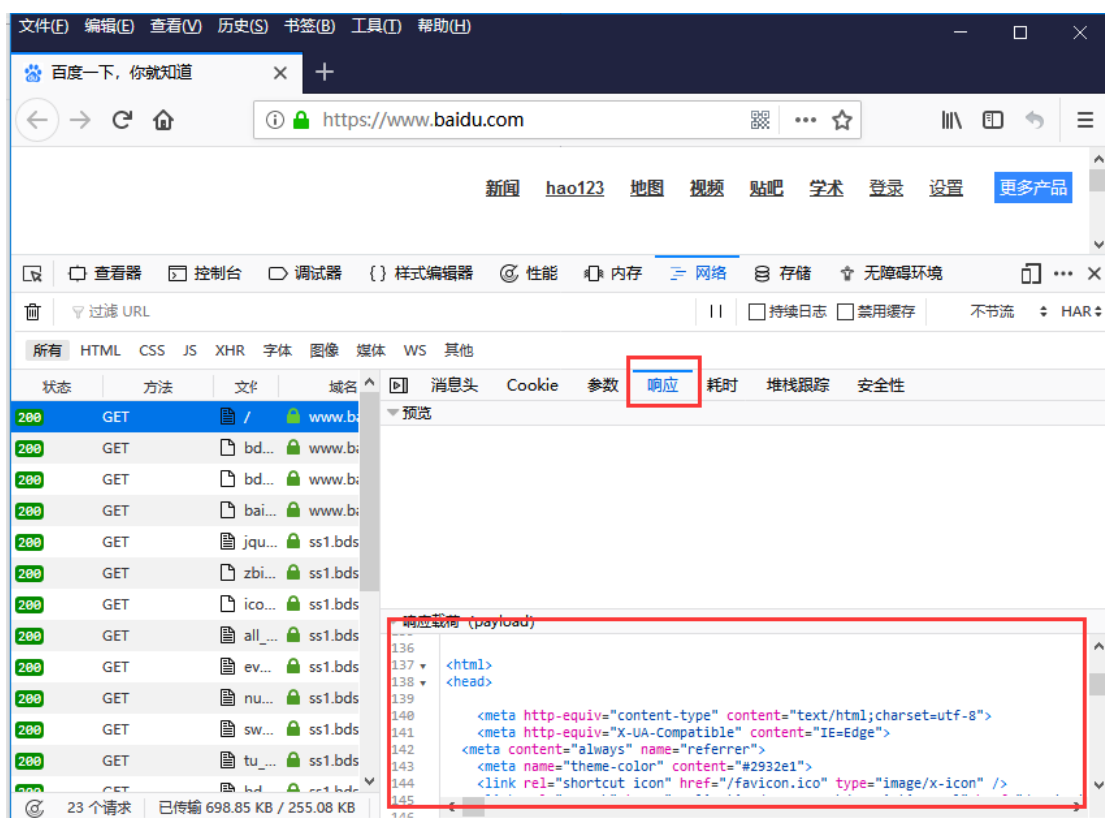


图 6-9 响应实体

请求被接收以后，服务器便可以根据请求返回对应的 HTML 源码，浏览器对 HTML 页面进行解析和渲染以后，我们就可以看到完整的页面了。

1.3. URL

怎样标识分布在整个因特网上的文档？

使用统一资源定位符 URL (Uniform Resource Locator) 来标识万维网上的各种资源，使每一个资源在整个因特网的范围内具有唯一的标识符 URL。

URL 的一般形式是：

http://<主机>:<端口>/路径

- http: 表示使用 HTTP 协议；
- 主机: 存放资源的主机，一个域名，或一个 IP 地址；
- 端口: HTTP 的默认端口号是 80，通常可省略；
- 路径: 访问资源的路径

1.4. 报文

客户端与服务端之间的信息传递使用的载体叫做报文。报文分为请求报文与响应报文。

- (1) 客户端向服务器发送一个请求报文
请求报文包含请求的方法、URL、协议版本、请求头部和请求数据
- (2) 服务器反馈给客户端一个响应报文
响应的内容包括协议的版本、成功或者错误响应码、服务器信息、响应头部和响应数据。

1.4.1. 请求报文 (request)

URL 只是标识资源的位置，而 HTTP 报文是用来提交和获取资源。客户端发送的 HTTP 请求消息，包括三个部分：请求行、请求头部、请求实体

图 6-10 给出了请求报文的一般格式：



图 6-10 请求报文格式

现在我们来详细分析上个章节所抓取的请求报文，如图 6-11 所示。

新请求

GET

https://www.baidu.com/

请求头:

Host: www.baidu.com

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Cookie: BIDUPSID=C5CE494E176F48A6AB42796B6FE0046F; PSTM=1509175985

Upgrade-Insecure-Requests: 1

请求主体:

图 6-10 请求报文

get 请求报文说明如下。

第一部分：请求行，用来说明请求类型、要访问的资源以及所使用的 HTTP 版本。

第二部分：请求头部，紧接着请求行之后的部分，用来说明服务器要使用的附加信息。

第三部分：请求数据也叫请求主体，可以添加向服务器提交的数据，这个例子的 get 请求数据为空。

(1) 请求方法

根据 HTTP 标准，HTTP 可以使用多种方法与服务器交互，如表 6-1 所示。

HTTP 1.0：定义了三种请求方法： GET, POST 和 HEAD 方法。

HTTP 1.1：在 1.0 基础上进行更新，新增了五种请求方法： OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

表 6-1 请求方法

序号	方法	描 述
1	GET	请求读取一个 Web 页面

2	HEAD	请求读取一个 Web 页面的首部
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件），数据被包含在请求体中。POST 请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	请求存储一个 Web 页面。
5	DELETE	删除 Web 页面
6	CONNECT	HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

（2）常用的请求头属性

表 6-2 请求属性

序号	请求头	描 述
1	Host	对应网址 URL 中的 Web 名称和端口号，用于指定被请求资源的 Internet 主机和端口号，通常属于 URL 的一部分
2	User-Agent	告诉服务器客户端使用的操作系统和浏览器的名称、版本。
3	Accep	指浏览器或其他客户端可以接受的文件类型，服务器可以根据它判断并返回适当的文件格式。 Accept: */*: 表示什么都可以接收。 Accept: image/gif: 表明客户端希望接受 GIF 图像格式的资源； Accept: text/html: 表明客户端希望接受 html 文本。
4	Accept-Language	指出浏览器可以接受的语言种类，如 en 或 en-us 指英语，zh 或者 zh-cn 指中文
5	Accept-Encoding	指出浏览器可以接受的编码方式。编码方式不同于文件格式，它是为了压缩文件并加速文件传递速度。浏览器在接收到 Web 响应之后先解码，然后再检查文件格式，许多情形下这可以减少大量的下载时间。
6	Referer	表明产生请求的网页来自于哪个 URL，用户是从该 Referer 页面访问到当前请求的页面。这个属性可以用来跟踪 Web 请求来自哪个页面，是从什么网站来的等。

		有时候遇到下载某网站图片，需要对应的 referer，否则无法下载图片，那是因为人家做了防盗链，原理就是根据 referer 去判断是否是本网站的地址，如果不是，则拒绝，如果是，就可以下载。
7	Cookie	浏览器用这个属性向服务器发送 Cookie，Cookie 是在浏览器中寄存的小型数据体，它可以记载和服务器相关的用户信息，也可以用来实现会话功能。
8	Connection	keep-alive：当一个网页打开完成后，客户端和服务端之间用于传输 HTTP 数据的 TCP 连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。 close：代表一个 request 完成后，客户端和服务端之间用于传输 HTTP 数据的 TCP 连接会关闭，当客户端再次发送 request 时，需要重新建立 TCP 连接。

1.4.2. 响应组成 (response)

HTTP 响应也由三个部分组成，分别是：状态行、响应头、响应正文，如图 6-11 所示。

```

HTTP/1.1 200 OK
Content-Type: text/html
Content-length: 59

<html>
<head></head>
<body>
<h1>Hello</h1>
</body>
</html>

```

图 6-11 响应报文

响应报文说明如下。

第一部分：状态行，由 HTTP 协议版本号、状态码、状态消息三部分组成。

第二部分：响应头部，紧接着请求行之后的部分，响应头用来描述服务器信息的说明。

第三部分：响应主体叫请求主体，服务器返回的实体数据。

(1) 状态码

状态码用来告诉 HTTP 客户端服务器是否产生了预期的 response。状态码总共只有三位，第一位表示状态类别，共分为五种，如表 6-3 所示。

表 6-3 状态类别

序号	分类	分类描述
1	1xx	进度通知类状态，表示“你的请求我正在处理”
2	2xx	表示“你的请求我已经成功处理了”
3	3xx	重定向，也就是服务器告诉客户端“你要的资源搬家了，你到某某地方再去找它吧”
4	4xx	客户端发来的响应报文里有些错误，比如语法错误或请求的资源不存在等。
5	5xx	服务器端有些问题，已经无法处理完成你的请求了

常用的状态码并不多，我们把常见的状态码含义列举如表 6-4。

表 6-4 常见状态码

状态码	名称	中文描述
200	OK	请求成功
301	Moved Permanently	资源被永久移动。请求的资源已被永久的移动到新 URI，返回信息会包括新的 URI，浏览器会自动定向到新 URI
302	Found	资源临时移动。资源只是临时被移动，客户端应继续使用原有 URI
403	Forbidden	没权限。服务器收到请求，但拒绝提供服务
404	Not Found	请求的资源不存在。遇到 404 首先检查请求 url 是否正确

500	Internal Server Error	服务器内部错误，无法完成请求
503	Service Unavailable	由于超载或系统维护（一般是访问人数过多），服务器无法处理客户端的请求，通常这只是暂时状态

（2）常用的响应头属性

表 6-5 响应头属性

序号	响应头	描 述
1	Date	生成消息的具体时间和日期
2	Server	指明 HTTP 服务器的软件信息
3	Content-Type	Web 服务器告诉浏览器自己响应的对象的类型和字符集
4	Content-length	指明实体正文的长度
5	Cache-Control	用来指定 Response—Request 遵循的缓存机制。 Public: 可以被任何缓存所缓存 Private: 指响应信息的全部或部分用于单个用户，而不能用一个共享缓存来缓存。这可以让源服务器指示，响应的特定部分只用于一个用户，而对其他用户的请求则是一个不可靠的响应。 No-cache: 所有内容都不会被缓存，请求头里的 no-cache 表示浏览器不想读缓存，并不是说没有缓存。一般在浏览器按 ctrl+F5 强制刷新时，请求头里也会有这个 no-cache，也就是跳过缓存，直接请求服务器
6	Set-cookie	用于把 cookie 发送到客户端浏览器，每一个写入 cookie 都会生成一个 set-cookie
7	Last-modified	用于指示资源的最后修改日期和时间
8	Content-encoding	Web 服务器表明自己使用了什么压缩方法 (gzip, deflate) 压缩响应中的对象

1.5.HTTP 协议应用场景

除了浏览器与服务器之间通信使用 HTTP，很多其他地方都使用，这些不同的场景的使用，所解决的问题不一样。

- 浏览器 / 服务器之间
- 手机 App / 服务器之间
- 服务器之间，如爬虫、搜索引擎、数据分析等应用
- 爬虫工具，典型如：搜索引擎等。

第二章 接口测试

理解 HTTP 协议是绝大多数接口测试的基础，在上一个章节中，我们详细介绍了 HTTP 协议，接下来就开始接口测试的内容。

2.1.为什么要做接口测试

大家都知道，接口其实就是前端页面或 APP 等调用与后端做交互用的，所以你可能会有这样的疑问，功能测试都测好了，为什么还要测接口呢？

除了之前我们所说的接口测试可以将测试工作前置之外，它还可以解决这样的问题，如用户注册功能，规定用户名为 6~18 个字符，包含字母（区分大小写）、数字、下划线。在做功能测试时肯定会对用户名规则进行测试时，比如输入 20 个字符、输入特殊字符等，但这些可能只是在前端做了校验，后端可能没做校验，如果有人通过抓包绕过前端校验直接发送到后端该如何处理？

试想一下，如果用户名和密码未在后端做校验，而有人又绕过前端校验的话，那用户名和密码就可以随便输了，如果是登录的话，还可能会通过 SQL 注入等手段拖数据库，甚至可以获取管理员权限。

所以，接口测试的必要性就体现出来了：

- 可以发现很多在页面上操作发现不了的 bug
- 检查系统的异常处理能力

- 接口测试相对 UI 测试也比较稳定，容易实现自动化持续集成，可以减少人工回归测试人力成本与时间，缩短测试周期，支持后端快速发版需求。

2.2. 接口测试的定义

接口测试主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。测试的重点是要检查数据的交换，传递和控制管理过程，以及系统间的相互逻辑依赖关系等。

简单来讲接口测试就是通过测试不同情况下的输入参数与之相应的返回结果，来判断接口是否符合或满足相应的功能性、安全性要求。

2.3. 接口测试实例分析

在做接口测试前，测试人员肯定会先拿到开发给予的接口文档。测试人员可以根据这个文档编写接口测试用例。

2.3.1. 接口文档解析

要做好接口测试，首先要学会解析接口文档，一般接口文档会包含接口的地址、使用的方法（get/post/put）等、必填参数、非必填参数、参数长度、返回结果，只有知道了这些信息才能设计测试用例。

举个抽奖接口的例子，如表 7-1 所示：

表 7-1 接口设计说明

接口名称	天天抽奖			
调用方式	RESTFUL			
接口地址	/appapi/luckDraw			
接口方法	Post			
● 输入参数定义				
列名	字段名	类型	必填	备注

用户手机	mobilePhone	Number	是	只接受长度为 11 的整数
活动 id	activityGuid	Number	是	只接受小于 1000 的整数
● 返回数据说明				
列名	字段名	类型	必填	备注
用户手机	mobilePhone	Number	是	
剩余抽奖次数	number	Number	是	每天限定 3 次抽奖机会
抽奖结果	successful	Varchar(10)	是	中奖—true；未中奖—false

从接口文档可以得到信息如下。

- (1) 接口的 URL 地址；
- (2) 接口的方法是 post；
- (3) 接口有 2 个必填的参数，一个是手机号，一个是活动；
- (4) 对于手机号参数数据类型是数字，且限定为 11 个数字；
- (5) 对于活动 ID 参数数据类型也是数字，且小于 1000 的数字；
- (6) 接口返回 3 个参数，用户手机号码、剩余抽奖次数、抽奖结果；
- (7) 返回的用户手机号码就是参与抽奖的用户手机号码；
- (8) 每天只有 3 次抽奖机会，抽一次少一次，当没有抽奖次数时，返回 Number 是 0，并且抽奖结果不能为 True；
- (9) 抽奖结果只能是 True 或者是 False。

解析完接口文档后，基本上可以明确测试点和预期结果，以便为之后测试用例做准备。

2.3.2. 测试用例设计

可以把接口测试当作是一个黑盒测试，测试各种输入情况，然后根据输出情况来判断是否符合预期结果，接口只是单一功能，不需要很复杂的测试用例，只要准备正常数据和异常数据，然后对应各种结果即可，先列一个大致的测试用例

如表 7-2 所示。

表 7-2 接口测试用例设计

输入数据	预期输出结果
不填写任何参数	报错，缺少手机号和活动 ID
只填写手机号	报错，缺少活动 ID
只填写活动 ID	报错，缺少手机号
填写非 11 位的手机号	报错，手机号码不正确
填写 11 位带符号的手机号	报错，手机号码不正确
填写非数字的活动 ID	报错，活动 ID 不正确
填写大于等于 1000 的活动 ID	报错，活动 ID 不正确
填写小于等于 0 的活动 ID	报错，活动 ID 不正确
填写 1000 以内的非整数的活动 ID	报错，活动 ID 不正确
填写符合要求的手机号和活动 ID	抽奖成功，返回手机号与输入手机号一致， 剩余抽奖次数为 2，抽奖结果 True/False
填写符合要求的手机号和活动 ID, 再次发送	抽奖成功，返回手机号与输入手机号一致， 剩余抽奖次数为 1，抽奖结果 True/False
填写符合要求的手机号和活动 ID, 第 3 次发送	抽奖成功，返回手机号与输入手机号一致， 剩余抽奖次数为 0，抽奖结果 True/False
填写符合要求的手机号和活动 ID, 第 4 次发送	抽奖失败，已经没有抽奖次数了
换一个符合要求的手机号和同样的活动 ID	抽奖成功，返回手机号与输入手机号一致， 剩余抽奖次数为 2，抽奖结果 True/False
换一个符合要求的活动 ID 和同样的手机号	抽奖成功，返回手机号与输入手机号一致， 剩余抽奖次数为 2，抽奖结果 True/False

至于是否中奖，还涉及到概率算法，需要调整概率并且配合使用并发来测试，完成了输入测试数据准备工作，接着就要使用接口测试工具执行测试用例了。

2.4. 接口测试工具

这里要重点推荐的是一款叫做 postman 的接口测试工具，Postman 一款非常流行的 API 调试工具。

2.4.1. 安装 Postman 工具

Postman 最早是作为 chrome 浏览器插件存在的，由于 2018 年初 Chrome 停止对 Chrome 应用程序的支持，Postman 提供了独立的安装包，不再依赖于 Chrome 浏览器了，推荐你使用这种方式安装。

(1) 访问 <https://www.getpostman.com/downloads/>，根据操作系统，选择 Windows 平台下对应版本的安装包，如图 7-1 所示。

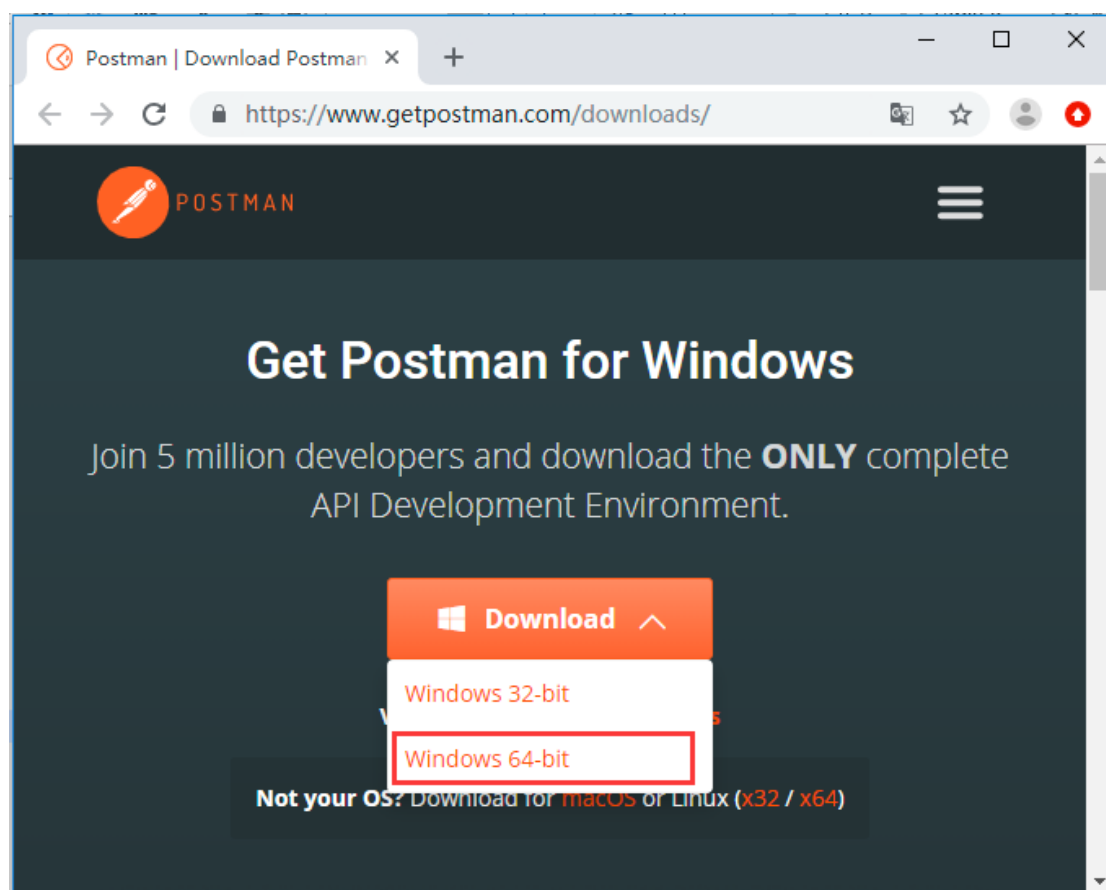


图 7-1 选择 Windows 平台的安装包

(2) 单击图 7-1 中的 Windows 64-bit 进行下载，下载后的文件名为“Postman-win64-6.7.1-Setup.exe”。双击该文件，进入安装 postman 的界面，

如图 7-2 所示。

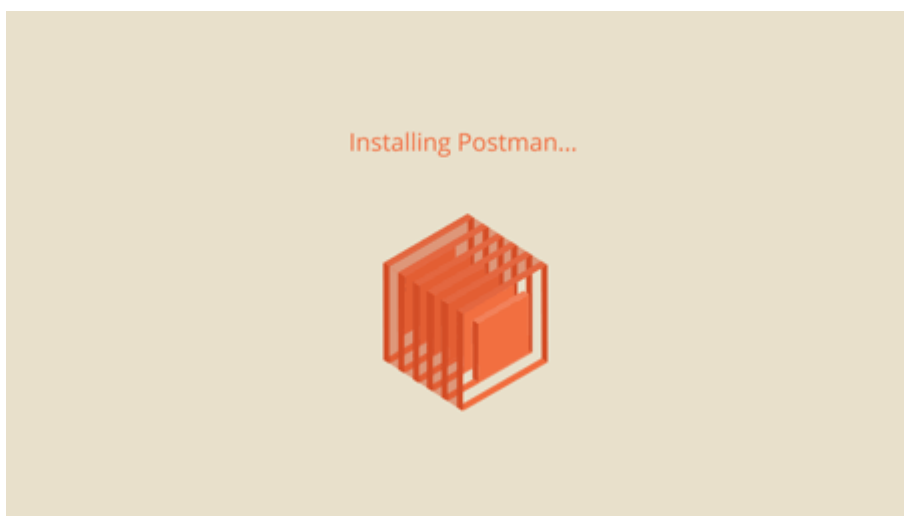


图 7-2 安装界面

(3) postman 的安装非常简单，现在已经安装成功了，在初次登录的时候很多人没有账号，所以说我们可以直接跳过注册，直接进入 Postman，如图 7-3 所示。

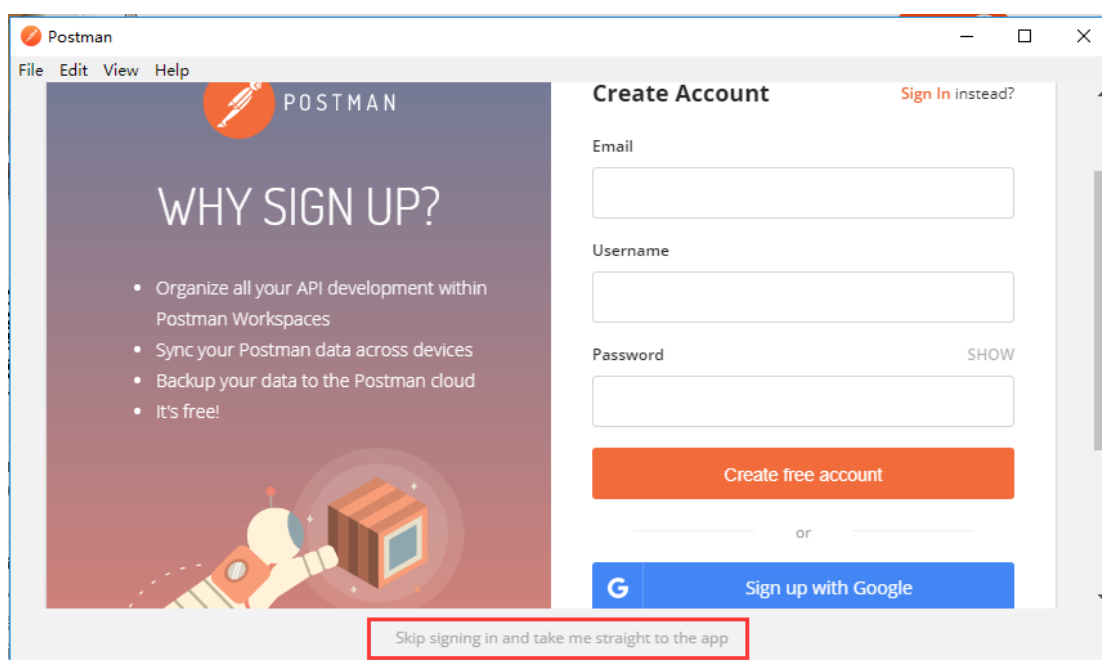


图 7-3 跳过注册直接进入 postman 界面

2.4.2. Postman 基础使用

(1) 首先我们选择创建 request 基础请求，如图 7-4 所示。

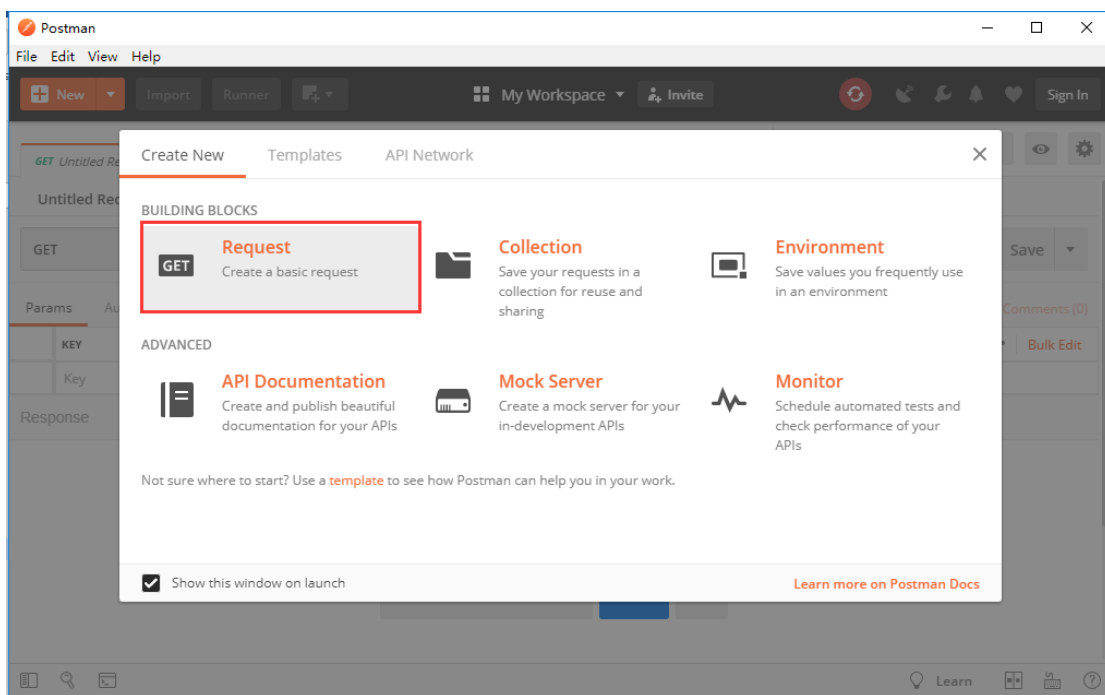


图 7-4 request 基础请求

除基础请求外，你还可以创建，Collection（请求集合文件夹）、Environment（环境变量）、API documents（API 文档）、Mock Server（模拟服务器）以及 Monitor（监视器）。

(2)在保存请求界面,输入请求名称“GET Request”、创建并选择 collection “Request Methods”，点击保存按钮，如图 7-5 所示。

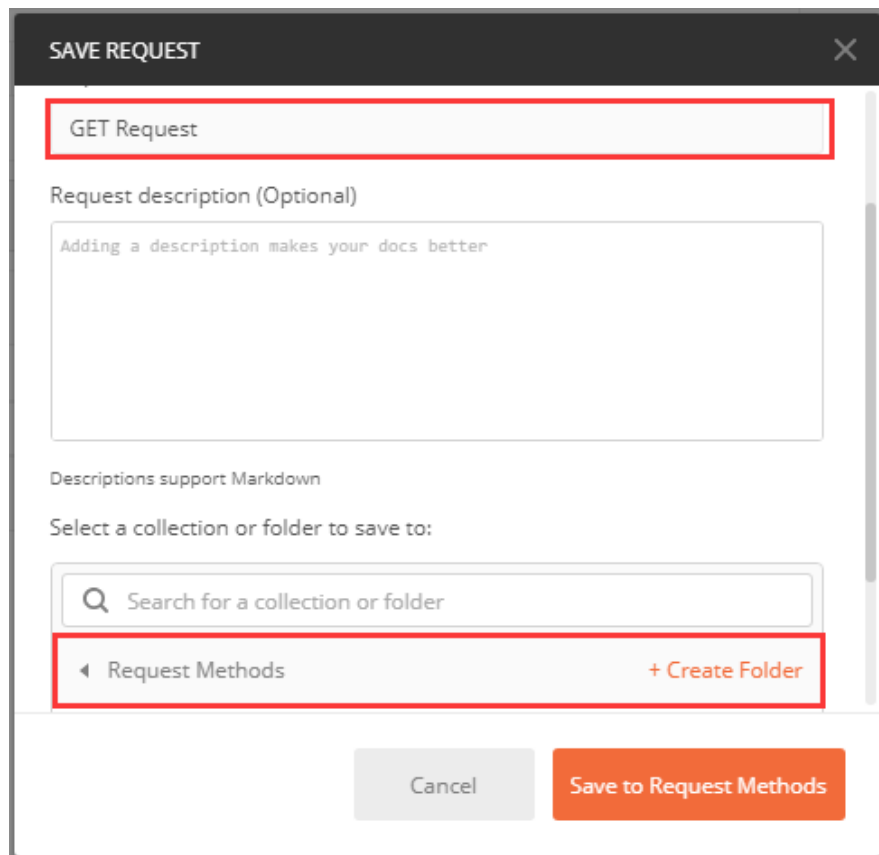


图 7-5 保存请求

(3) 请求保存后, 你可以添加 URL, 点击 Params 输入参数及 value, 可输入多个, 即时显示在 URL 链接上, 构建一个完整的 get 请求, 点击【send】按钮, 发送请求, 并得到响应, 如图 7-6 所示。

Postman Echo 提供了示例 API 调用, 你可以通过 <https://docs.postman-echo.com/> 来查看接口说明文档, 创建 http 请求, 在这里我们使用示例 API 中的 get 请求。

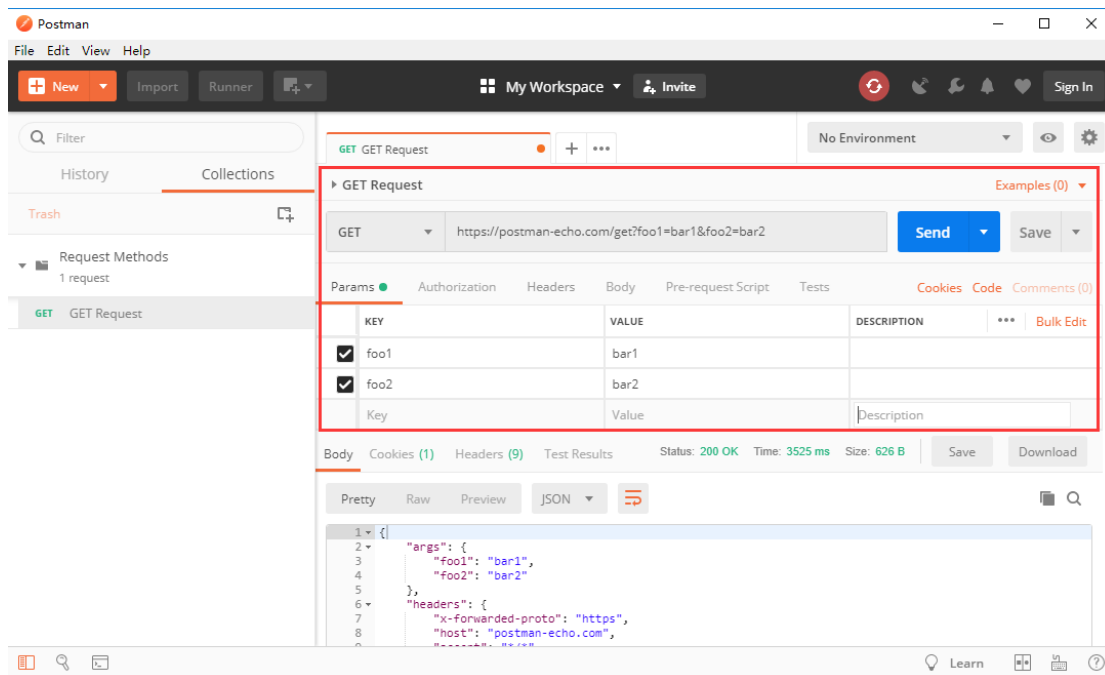


图 7-5 get 请求

(4) 一个完整的接口测试，包括：请求→获取响应正文→断言，我们已经知道了请求与获取响应，接下来使用 postman 进行断言，这个“Tests”就是我们需要处理断言的地方，postman 很人性化的帮我们把断言所用的函数准备好了，这里如图 7-6 所示。

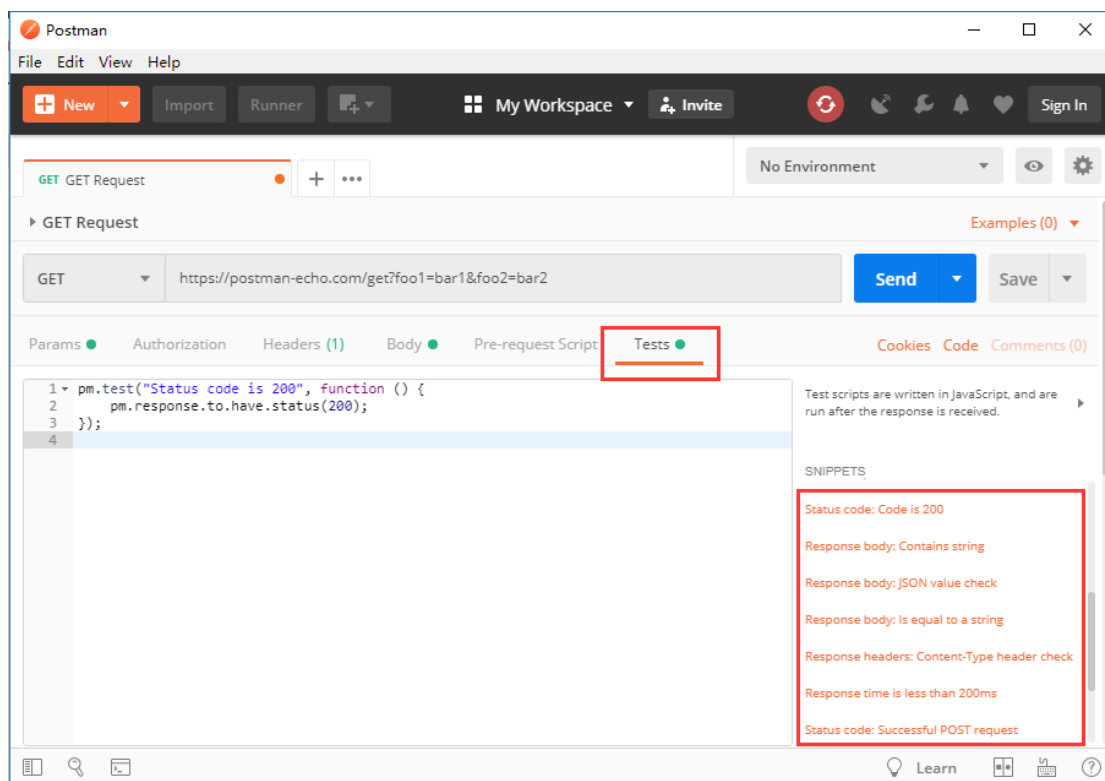
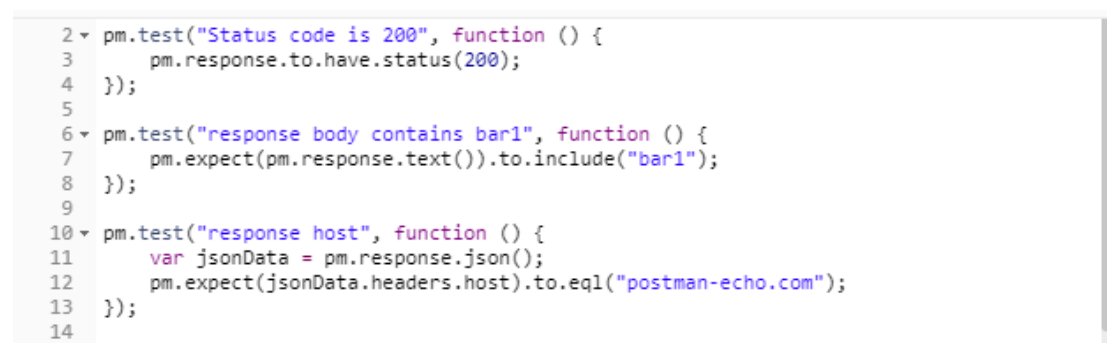


图 7-6 Tests 添加断言

在 Tests 中选择下面的常用断言，来实现断言场景，如图 7-7 所示。

1. Status code : Code is 200: 判断 http 返回状态码为 200
2. Response body: Contains string: 判断响应正文中是否包含“bar1”
3. Response body: JSON value check: 解析响应 json 数据，判断 host 的值是“postman-echo.com”



```
2 pm.test("Status code is 200", function () {
3   pm.response.to.have.status(200);
4 });
5
6 pm.test("response body contains bar1", function () {
7   pm.expect(pm.response.text()).to.include("bar1");
8 });
9
10 pm.test("response host", function () {
11   var jsonData = pm.response.json();
12   pm.expect(jsonData.headers.host).to.eql("postman-echo.com");
13 });
14
```

图 7-7 断言脚本

第一个断言代码的含义：名称为“Status code is 200”的断言中，判断响应状态码是否为 200，断言名称可以自行修改。

第二个断言代码的含义：判断响应文本内容中是否包含字符串 bar1。

第三个断言代码对 json 字符串进行解析，原始代码如下。

```
pm.test("Your test name", function () {
  var jsonData = pm.response.json();
  pm.expect(jsonData.value).to.eql(100);
});
```

其中 jsonData 变量是解析后的 json 对象，在 JS 中，一个 JSON 对象获取其属性的值，直接是用 jsonData.value，于是，我们把代码给修改一下，来判断第 3 条场景：

```
pm.test("response host", function () {
  var jsonData = pm.response.json();
  pm.expect(jsonData.headers.host).to.eql("postman-echo.com");
});
```


(5) 我们一共创建了 Tests 的断言 3 个，点击【send】，发送请求，在响应区内可以看到断言全部通过，如图 7-7 所示。

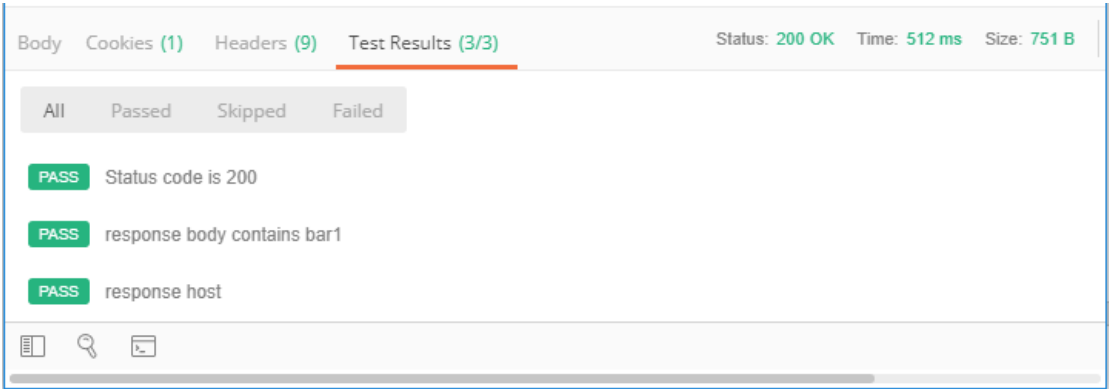


图 7-8 断言全部通过

需要大家注意的一点是，get 请求的参数随地址栏传递给服务器，post 请求相对于 get 请求多了个 body 部分，如图 7-9 所示。

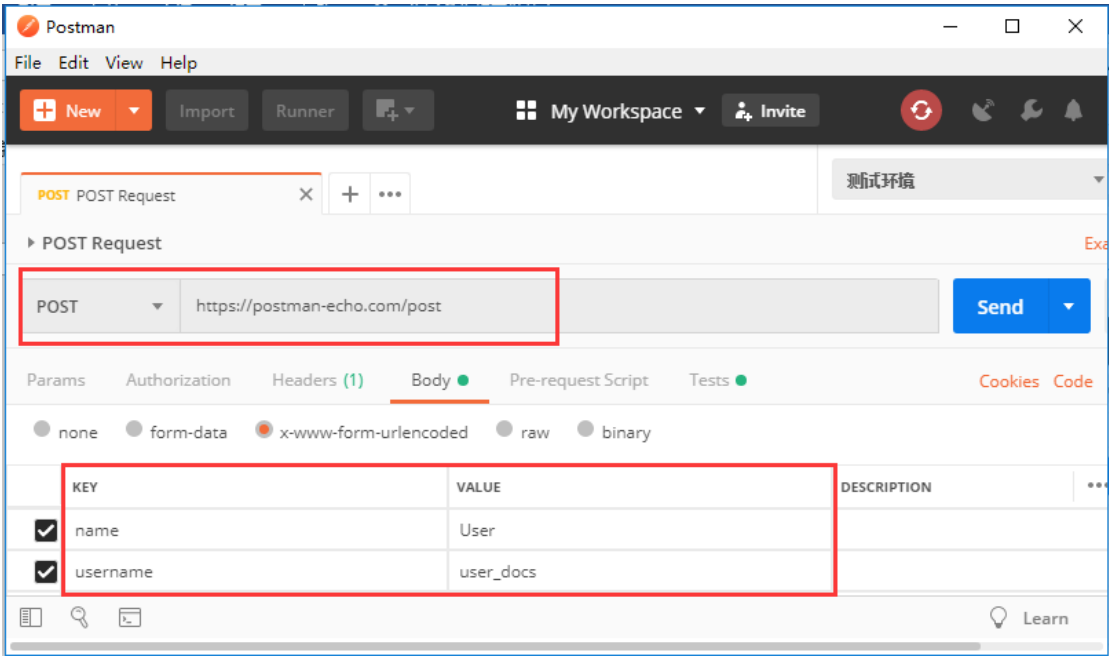


图 7-9 post 请求

Body：设置 POST 请求的参数：

- form-data： 是一种表单格式，它会将表单的数据处理为一条消息，如 form-data; name="file"; filename="chrome.png" 将数据传递给服务器。

- x-www-form-urlencoded: 浏览器的原生 form 表单, 以如下的数据格式
foo1=bar1&foo2=bar2 将数据传递给服务器。
- raw: 可以发送任意格式的接口数据, 可以 text、json、xml、html 等。
- binary: HTTP 请求中的 Content-Type:application/octet-stream, 只
可以发送二进制数据。通常用于文件的上传。

在使用 post 方法时, 需要注意 post 请求参数的格式。

2.4.3. Postman 进阶使用

设置环境变量

在实际执行接口测试时, 有些接口需要在测试环境、预热环境及生产环境均运行, 此时可以通过设置环境变量来动态选择。

(1) 点击左上角的【New】→Environment, 创建环境变量, 如图 7-10 所示。

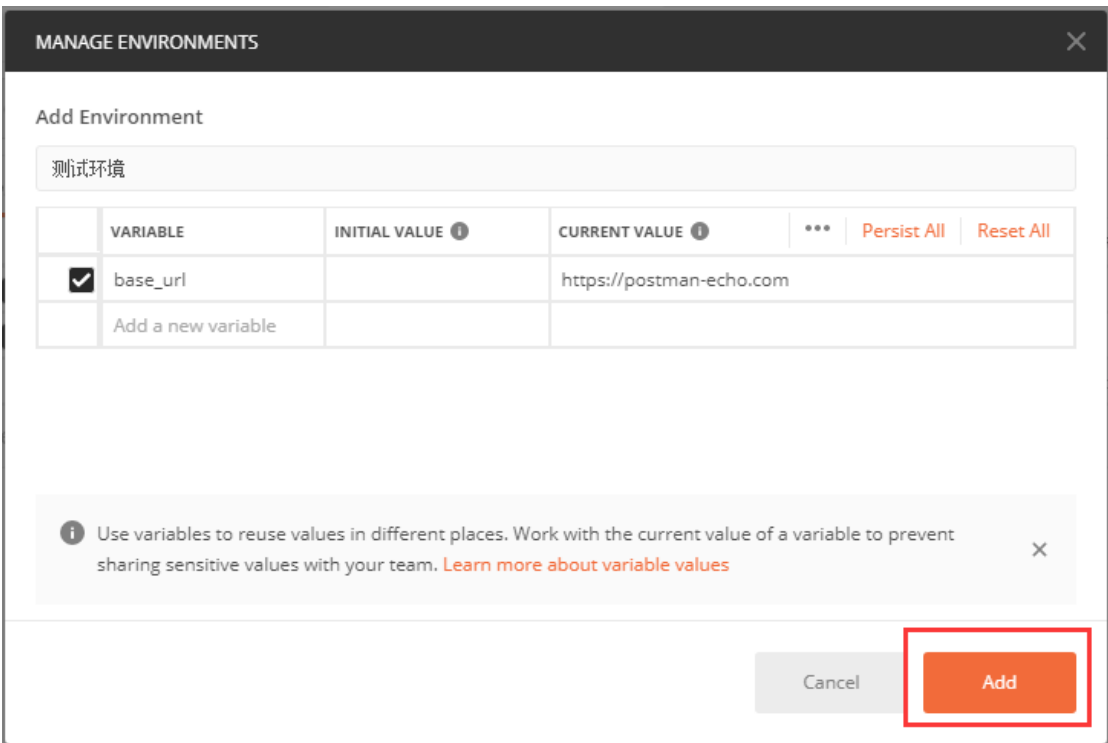


图 7-10 新建环境变量

(2) 使用这些键值的时候只需要加上两个花括号引用 key, 同时在右上角下拉列表选择需要的环境就可以了, 如图 7-11 所示。

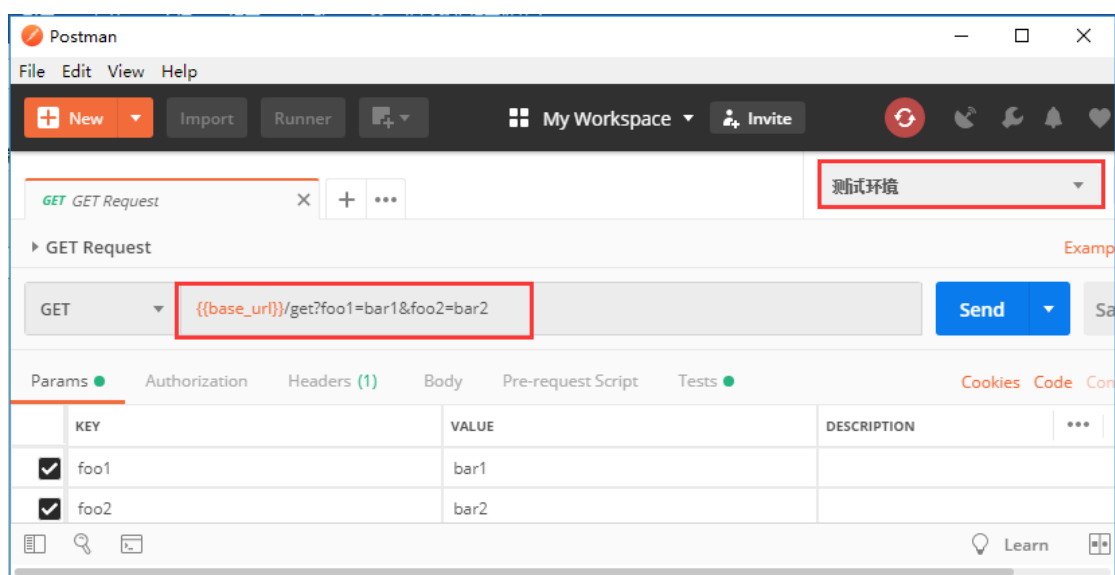


图 7-11 切换环境

建立多个环境时，不同的环境往往 key 通常都是相同的，只是 value 的值不同而已。

使用 Collections 管理用例

集合是一组请求，可以作为一系列请求在对应的环境中一起运行。当你希望自动化 API 测试时，运行集合非常有用。运行集合时，将逐个发送集合中的所有请求。

(1) 我们创建了一个名为 Request Methods 的集合，包含两个请求，如图 7-12 所示。

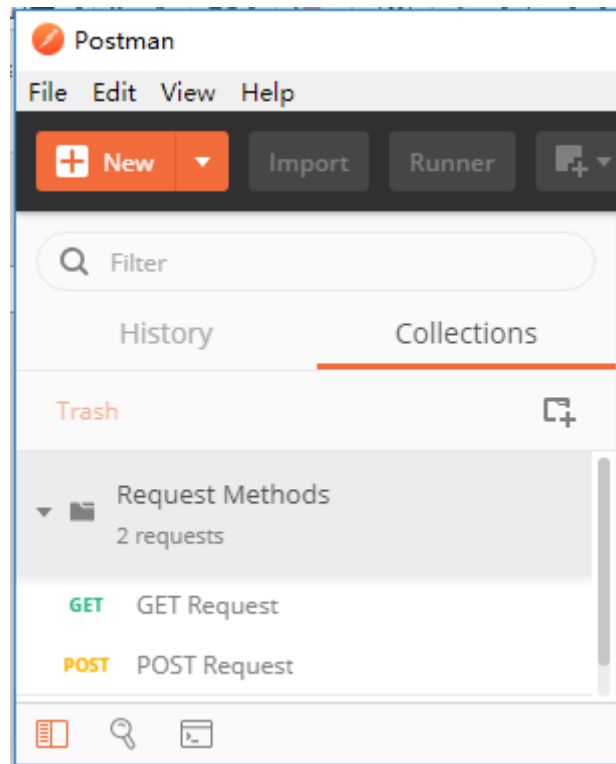


图 7-12 collections

(2) 运行 Collection, 一次执行整个 collection 里的用例, 点击图 7-13 中的 Run。

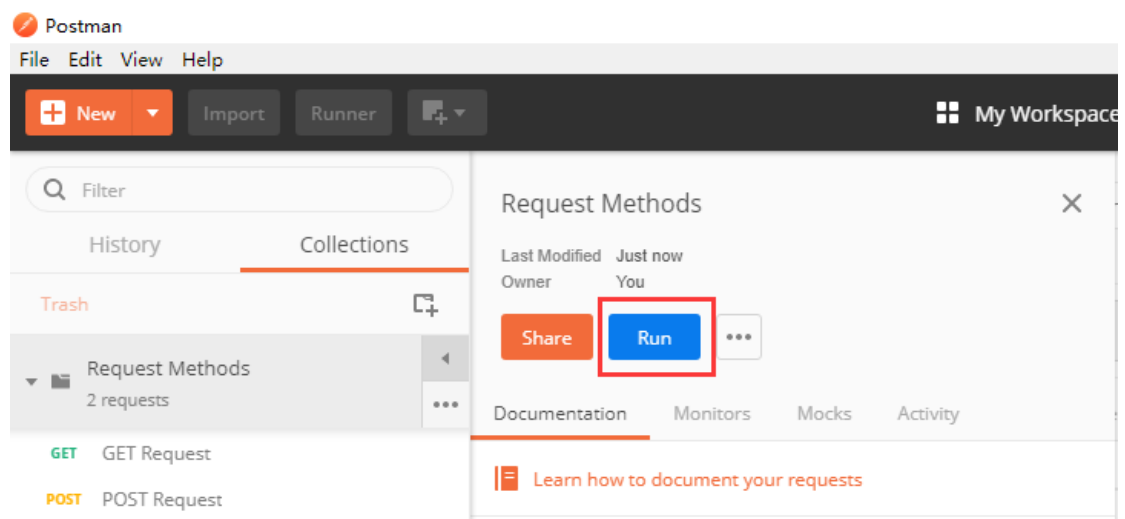


图 7-13 运行 collections

(3) 进入 Collection Runner 界面, 选择 Request Methods 集合, 如图 7-14 所示。

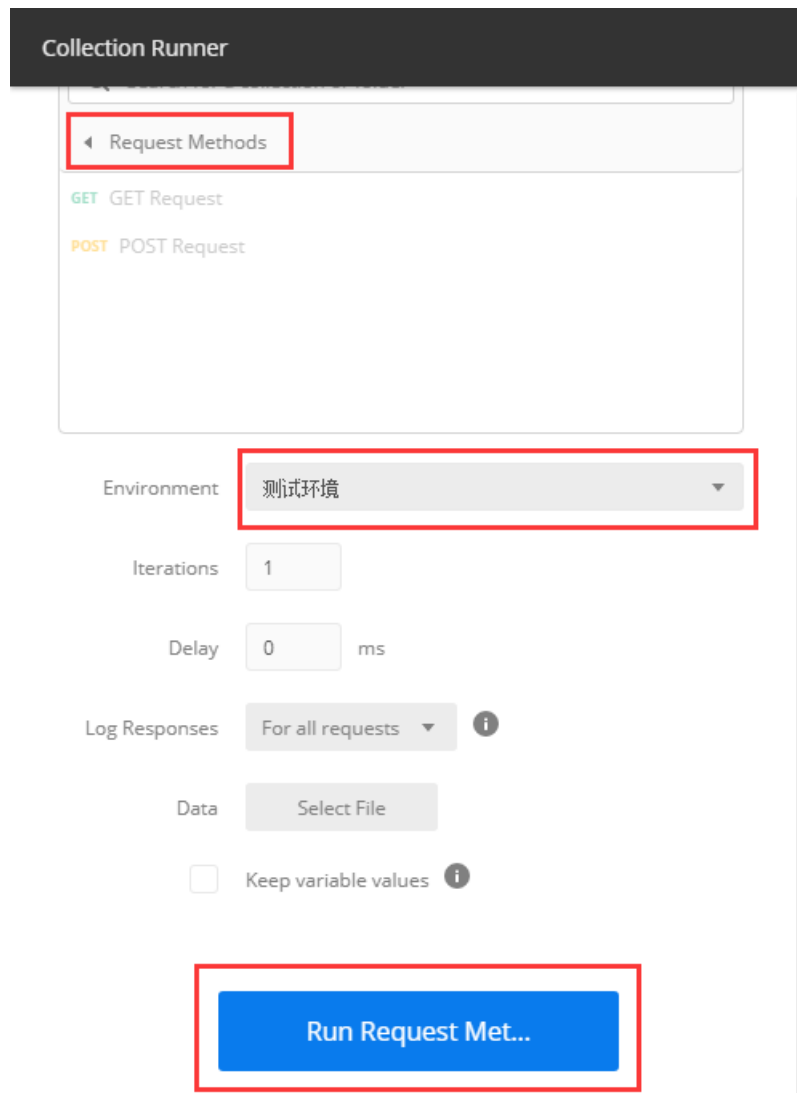


图 7-14 运行 collection runner 设置

- Environment，即运行环境，选择之前所创建的测试环境。
- Iterations，即重复运行次数。会将选择好的 collection 中 folder 重复运行。
- Delay，间隔时间。用例与用例间的间隔时间。
- Data，外部数据加载，即用例的参数化，可以与 Iterations 结合起来用，实现参数化，也就是数据驱动。

(4) 运行完成后，即可得出一个简易的聚合报告，如图 7-15 所示。

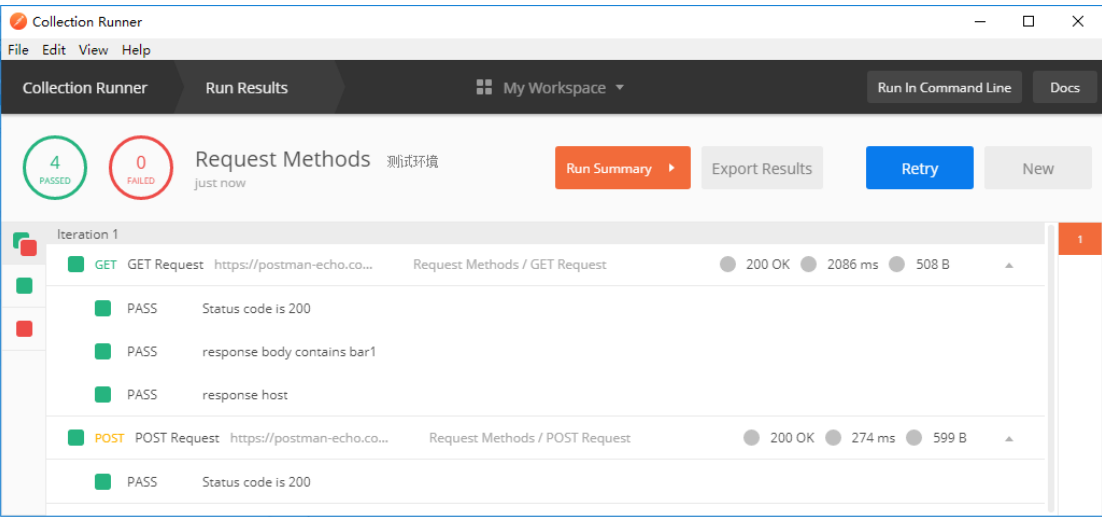


图 7-15 测试报告

选取外部文件作为数据源

在上一个章节中，我们对天天抽奖接口进行业务分析，共设计了 15 条测试数据，在重复运行某个接口时，如果希望每次运行，使用不同的数据，那么应该满足如下 2 个条件：

- 1、脚本中要用到数据的地方参数化，即用一个变量来代替，每次运行时，重新获取当前的运行数据。
- 2、需要有一个数据池，这个数据池里的数据条数，要与重复运行的次数相同。

（1）我们用 Postman Echo 中的 get 方法作为案例，首先创建一个名为“data.csv”的数据文件作为源数据，内容如表 7-3 所示：

表 7-3 数据文件

param1	Param2
test1	user1
test2	User2

在数据表中，创建两个参数分别为 param1 和 param2，每个参数分别有 2 个值。

（2）在 Request Method 下，添加 get 请求，将 URL 中的定值用 csv 文件中的参数来代替，如图 7-16 所示。

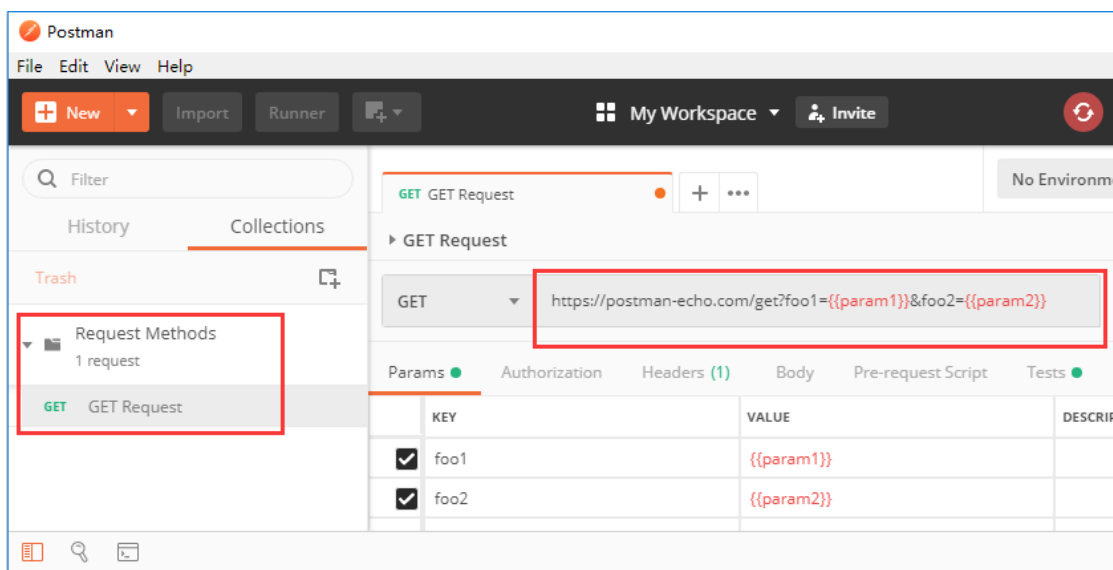


图 7-16 参数化

(3) 保存后，调用 Runner 模块运行此集合，选择外部数据文件，Iterations 运行次数会自动化匹配外部数据文件中的数据条数，如图 7-17 所示。

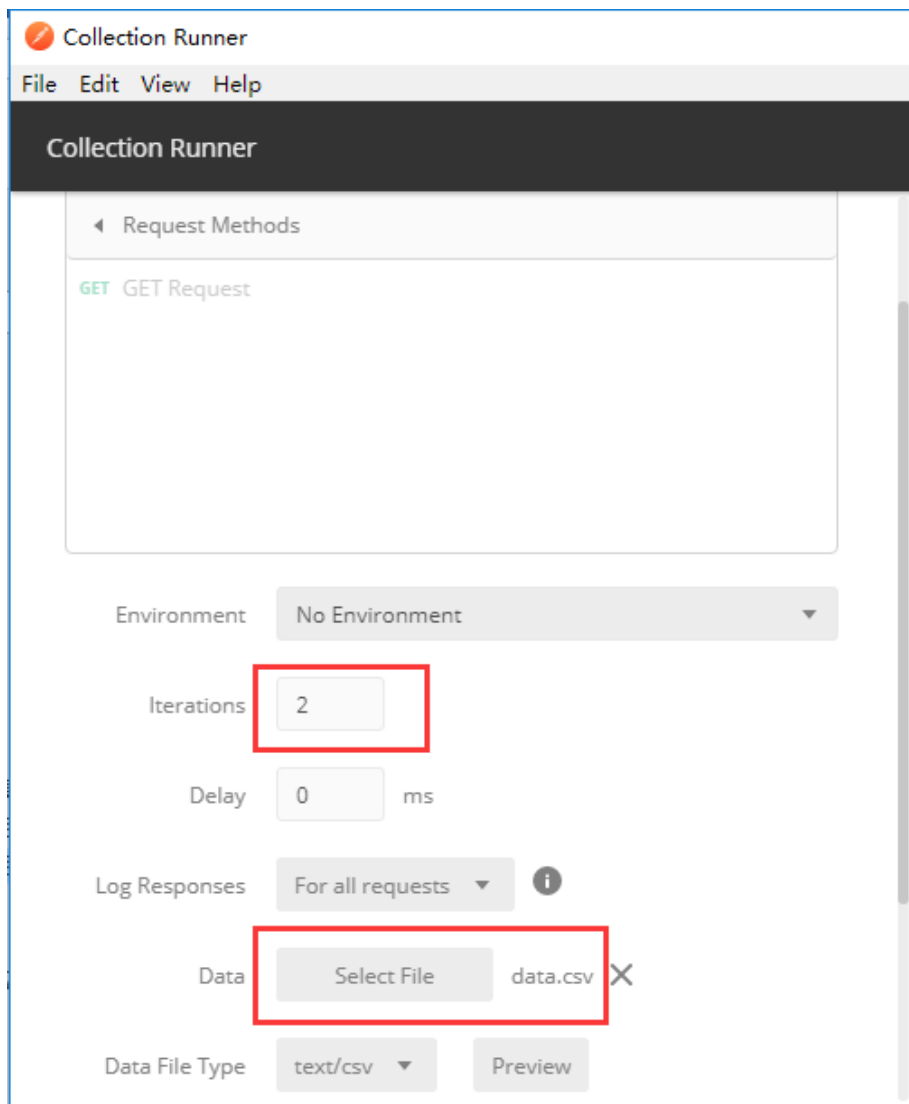


图 7-17 运行集合设置

(4) 运行完成后查看报告，可以看出一共循环两次，第一次循环取数据表中的第一条数据，第二次循环取数据表中的第二条数据，如图 7-18 所示。

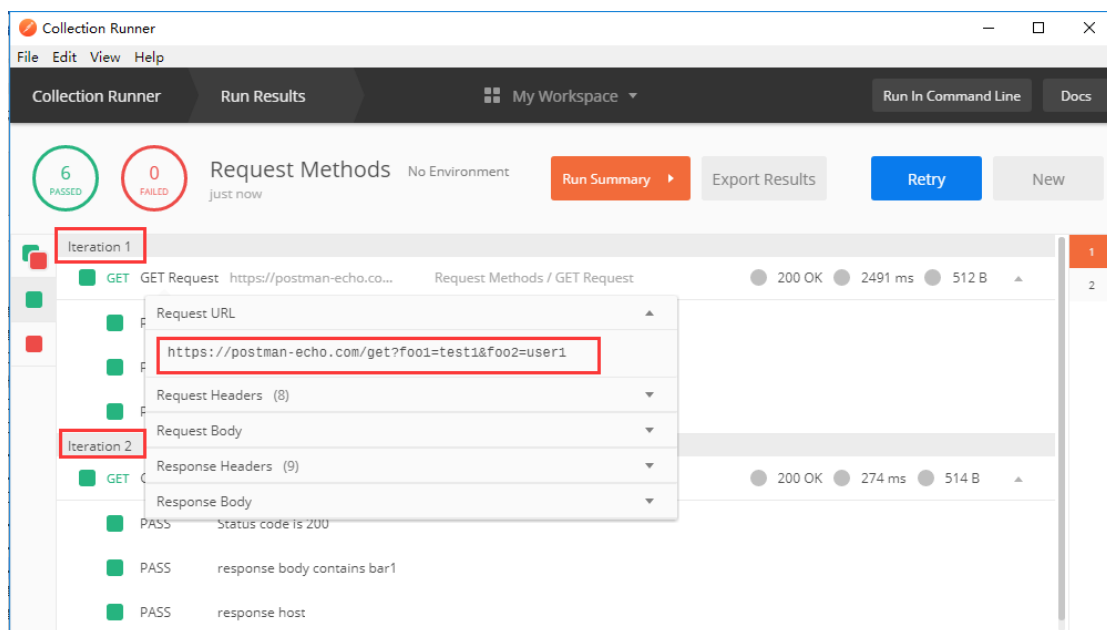


图 7-18 运行结果

Postman 的常用功能已经介绍完了，在实际项目中，你还会遇到一些其他的用法，这时候可以查看 postman 的帮助文档查看更多用法。

<https://learning.getpostman.com/docs/>

2.5. 本章小结

接口测试是近些年逐渐重视起来的测试类型，有人说接口测试属于黑盒测试，有人说接口测试属于白盒测试，还有人说其属于灰盒测试。不管其属于哪种测试类型，由于其用例开发周期短、缺陷命中率高、用例维护成本低等特点使得接口测试在实际工作中的效率和作用越来越被用户认可，本章通过一些常用的小工具和案例讲解接口测试的基本过程。

第三章 接口测试自动化测试

3.1.newman 的使用

通过 Postman 与 Newman 结合我们还可以批量运行 API 达到 API 自动化测试的目的。Newman 是为 Postman 而生，专门用来运行 Postman 编写好的脚本。

3.1.1. 安装 nodejs

(1) 访问 <https://nodejs.org/en/download/>，选择 Windows 平台下的安装包，如图 10-1 所示。

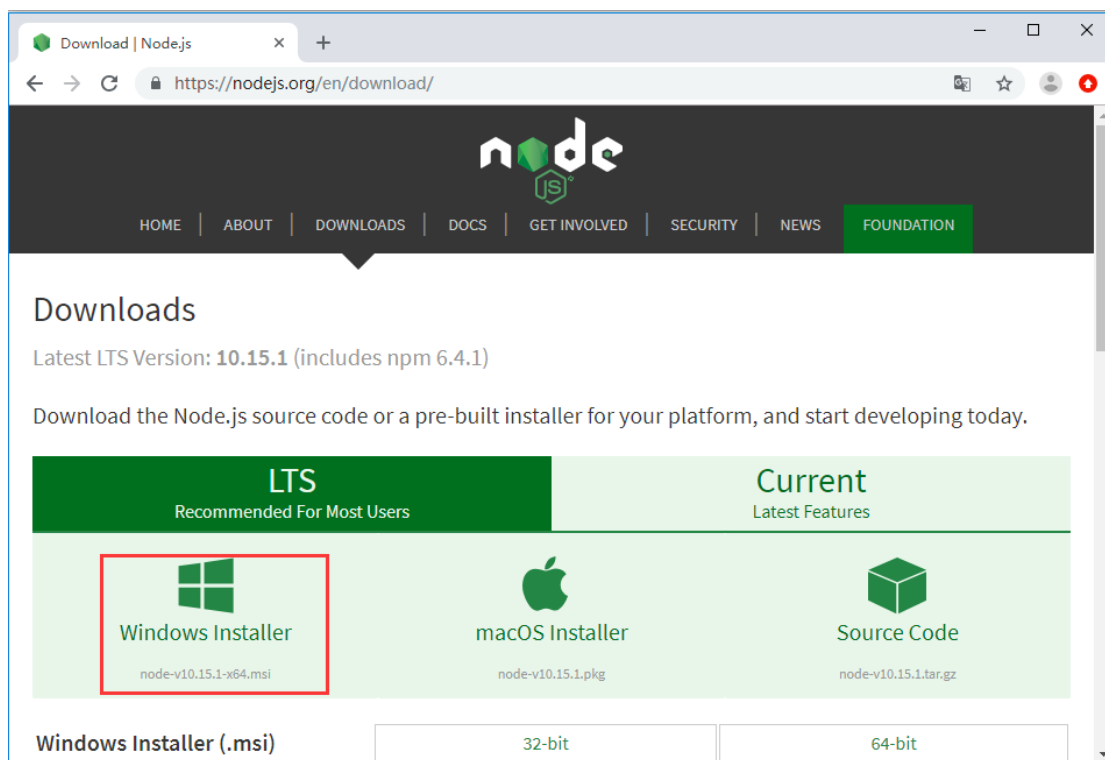


图 10-1 选择 Windows 平台的安装包

(2) 单击图 10-1 中的 Windows 进行下载，下载后的文件名为“node-v10.15.1-x64.msi”，双击该文件开始安装，如图 10-2 所示。

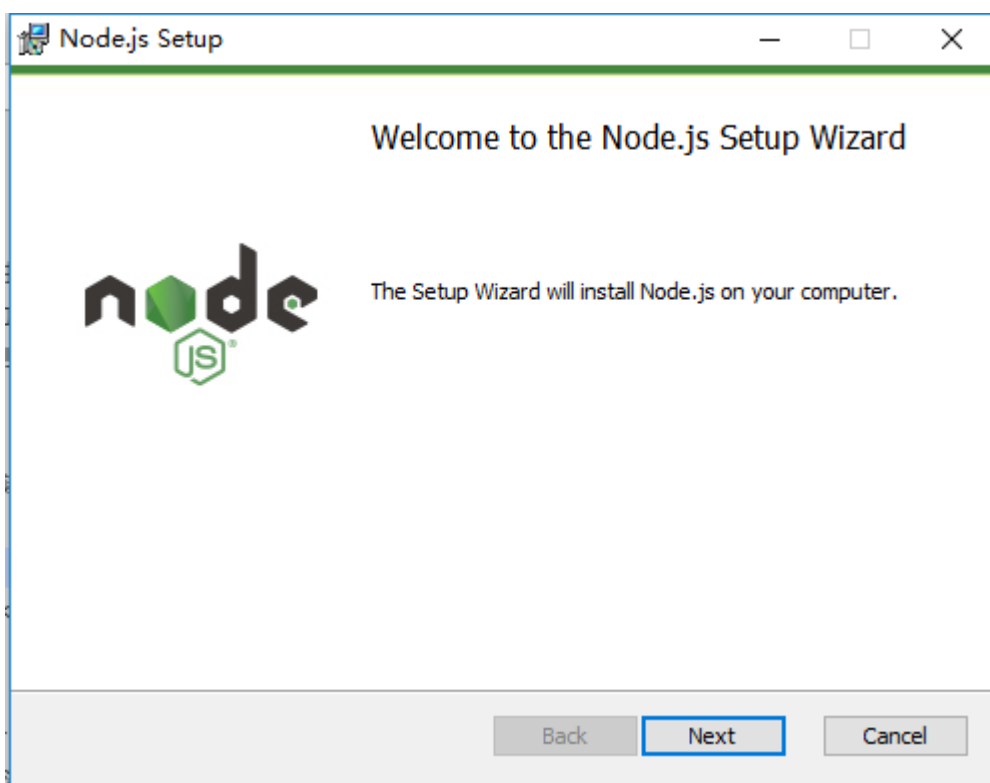


图 10-2 开始安装界面

(3) 单击图 10-2 中【Next>】按钮，接受 license 后，选择默认路径安装，如图 10-3 所示。

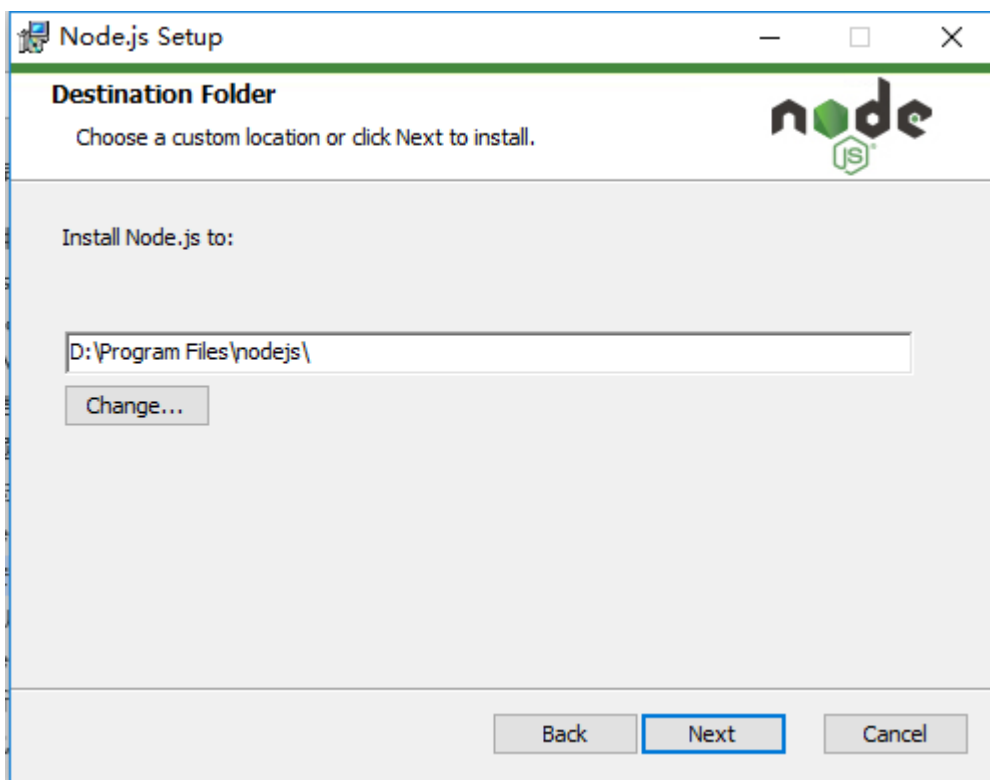


图 10-3 路径选择界面

(4) 单击图 10-3 中【Next>】按钮，选择默认设置，点击 install 后开始安装，如图 10-4 所示。

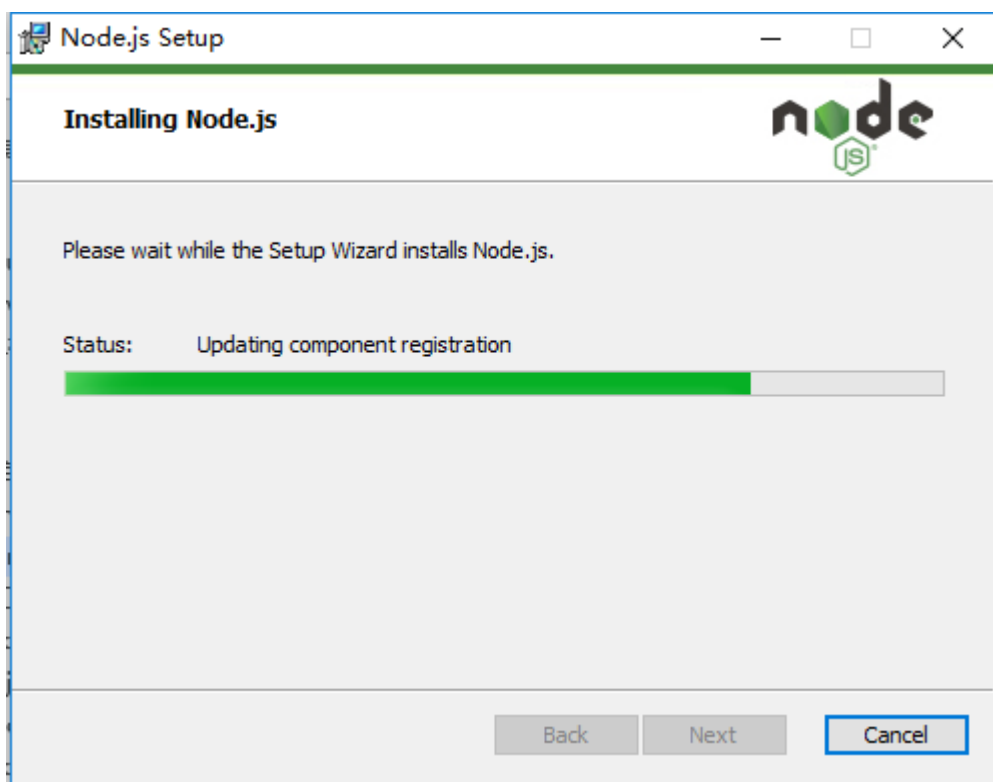


图 10-4 安装界面

(5) node.js 的安装进度非常快，安装成功后的界面如图 10-5 所示。

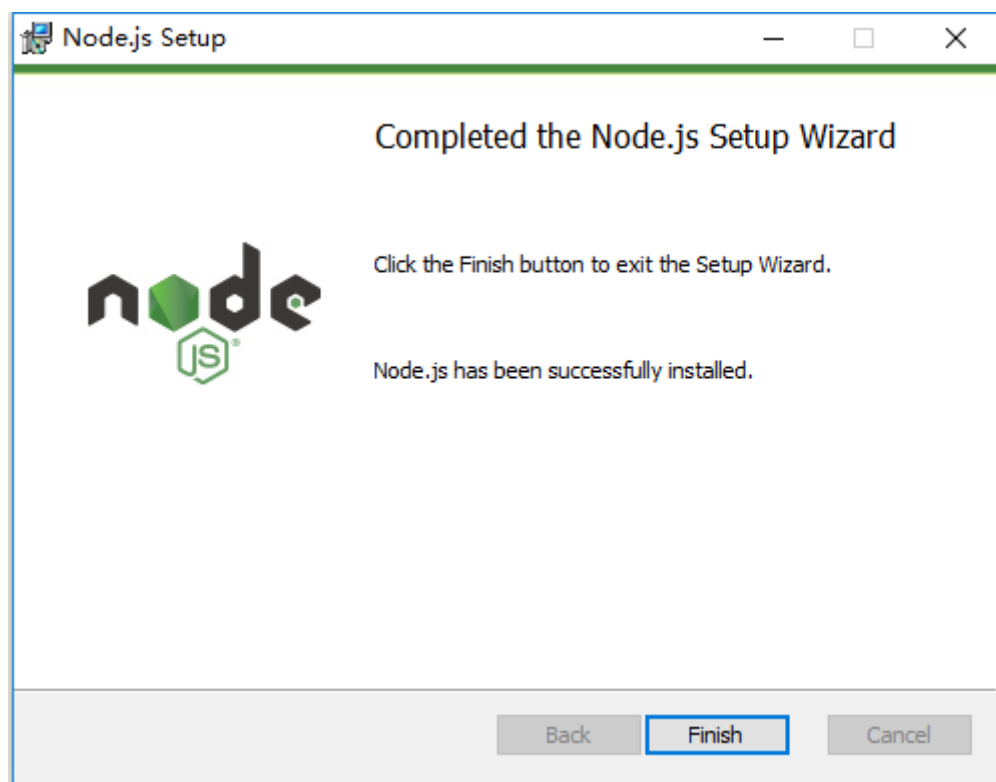
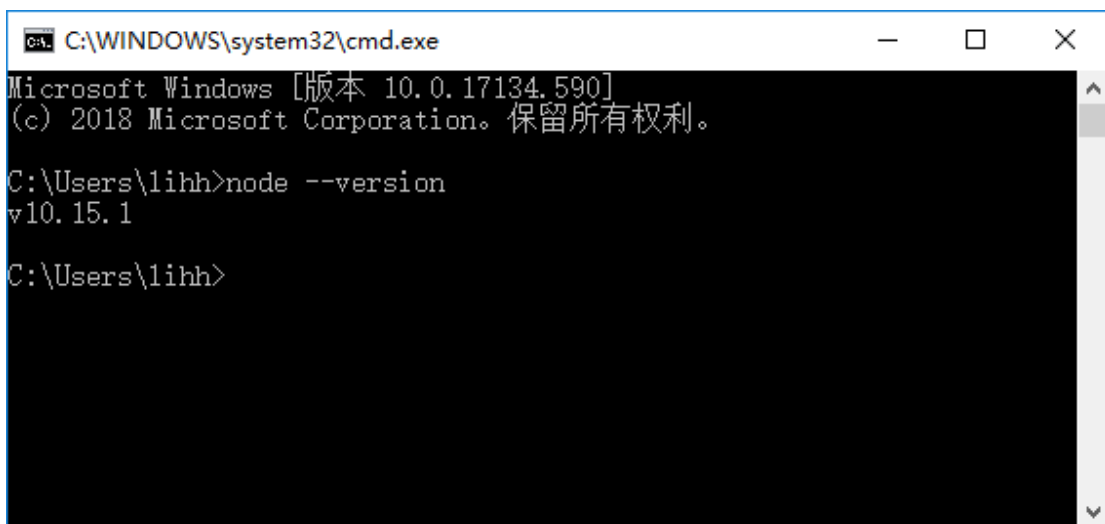


图 10-4 安装结束界面

(6) 此时，在控制台输入“node --version”，控制台会打印出 nodejs 的版本信息，如图 10-5 所示。



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.17134.590]
(c) 2018 Microsoft Corporation。保留所有权利。

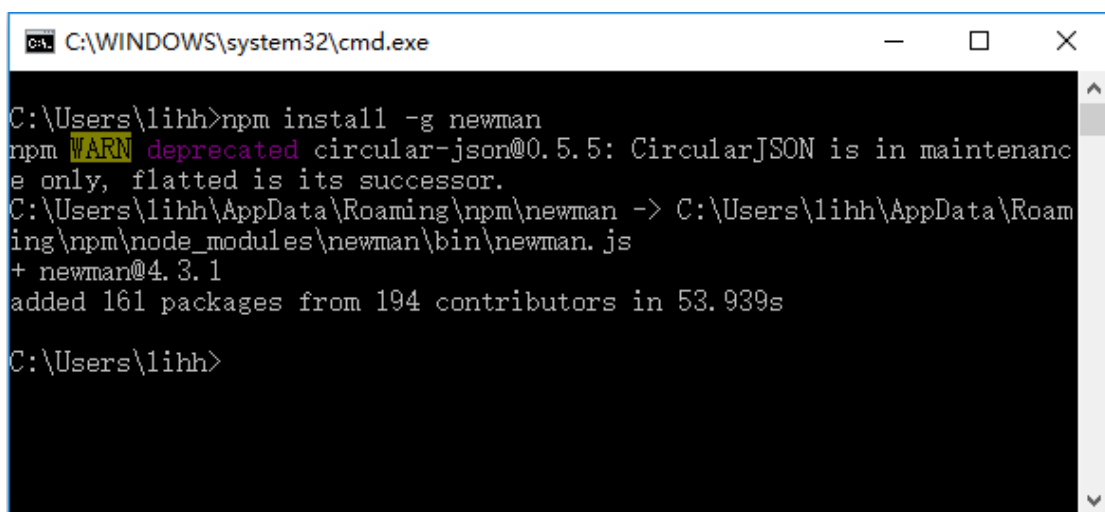
C:\Users\lihh>node --version
v10.15.1

C:\Users\lihh>
```

图 10-5 安装成功后的控制台输出

3.1.2. 安装 newman

(1) 在控制台输入“npm install -g newman”，安装 newman，如图 10-6 所示。



```
C:\WINDOWS\system32\cmd.exe

C:\Users\lihh>npm install -g newman
npm WARN deprecated circular-json@0.5.5: CircularJSON is in maintenanc
e only, flattened is its successor.
C:\Users\lihh\AppData\Roaming\npm\newman -> C:\Users\lihh\AppData\Roam
ing\npm\node_modules\newman\bin\newman.js
+ newman@4.3.1
added 161 packages from 194 contributors in 53.939s

C:\Users\lihh>
```

图 10-6 安装 newman

(2) 此时，在控制台输入“newman --version”，检验是否安装成功，安装成功后，控制台会输出 newman 版本信息，如图 10-7 所示。

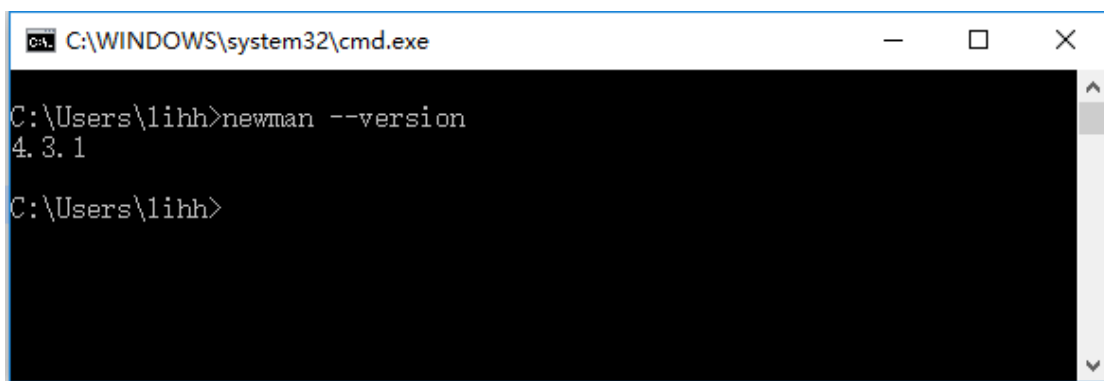


图 10-7 安装成功后的控制台输出

3.1.3. 通过 newman 执行脚本

Newman 通过命令来运行 postman 导出的 json 文件：

```
newman run <collection-file-source> [options]
```

run 后面是要执行的 json 文件，options 为可选项往往是一些参数，例如环境变量，测试报告，接口请求超时时间等等。

(1) 在这里我们导出之前章节中所添加的接口请求及环境变量，在控制台输入“newman run d:\test.json --environment d:\env.json”，控制台会输出执行结果，如图 10-8 所示。

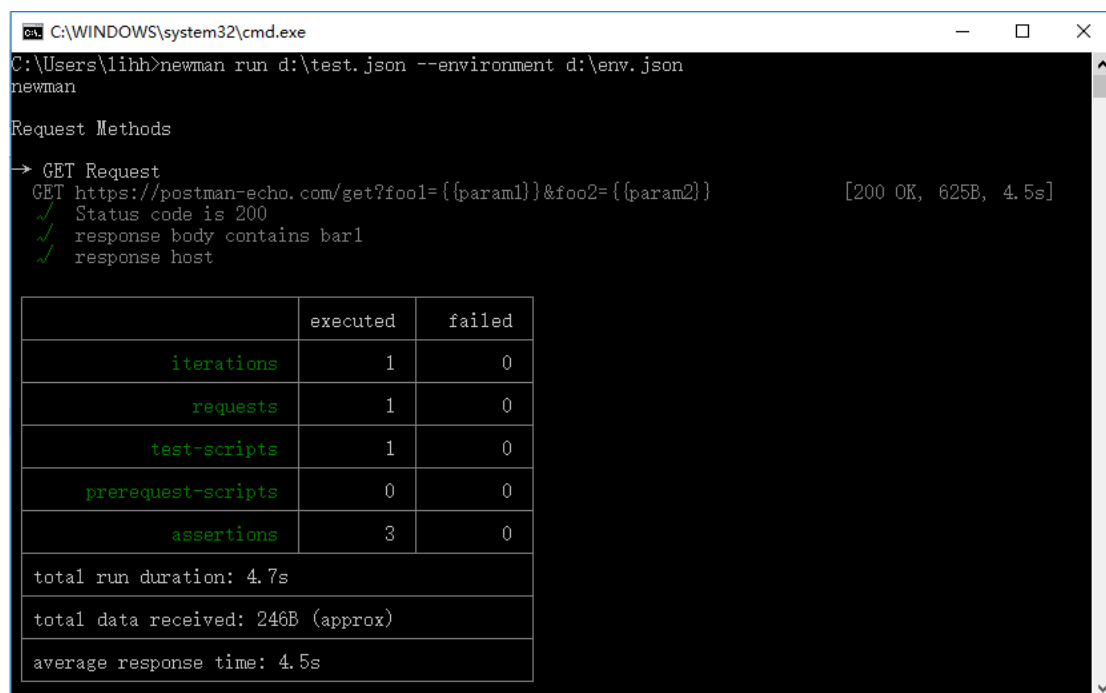
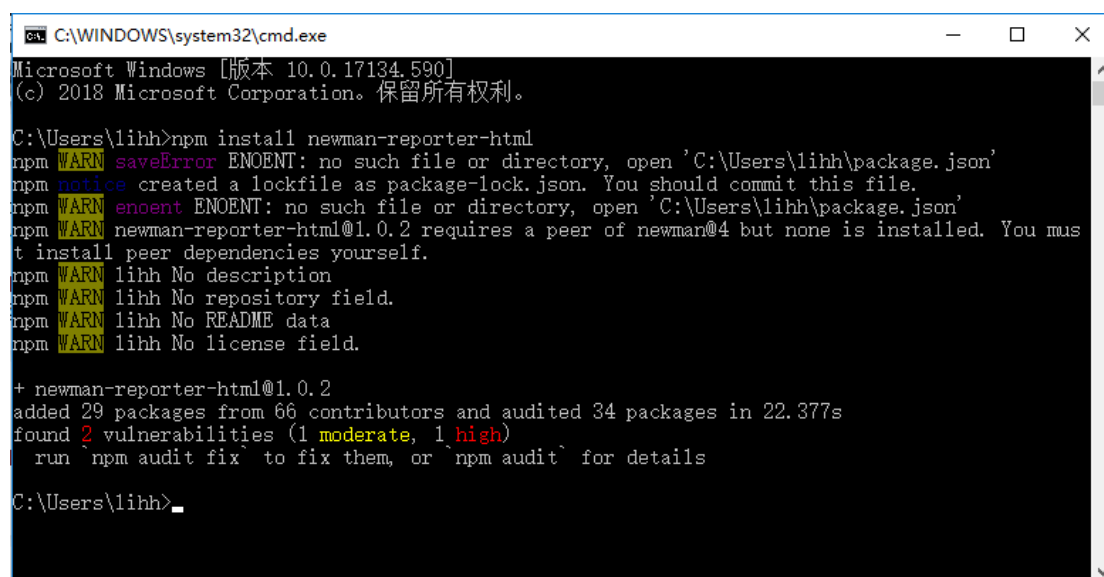


图 10-8 执行结果

(2) 如果希望 newman 执行完测试后，生成对应的 html 结果报告，则首先需要添加 html 插件，在控制台输入“npm install newman-reporter-html”命令安装 html 插件，如图 10-9 所示。



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.17134.590]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\lihh>npm install newman-reporter-html
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\lihh\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\lihh\package.json'
npm WARN newman-reporter-html@1.0.2 requires a peer of newman@4 but none is installed. You must install peer dependencies yourself.
npm WARN lihh No description
npm WARN lihh No repository field.
npm WARN lihh No README data
npm WARN lihh No license field.

+ newman-reporter-html@1.0.2
added 29 packages from 66 contributors and audited 34 packages in 22.377s
found 2 vulnerabilities (1 moderate, 1 high)
  run `npm audit fix` to fix them, or `npm audit` for details

C:\Users\lihh>
```

图 10-9 安装 html 插件

(3) 此时，在控制台输入“newman run d:\test.json --environment d:\env.json -r html --reporter-html-export d:\result.html”，生成 html 报告，如图 10-10 所示。

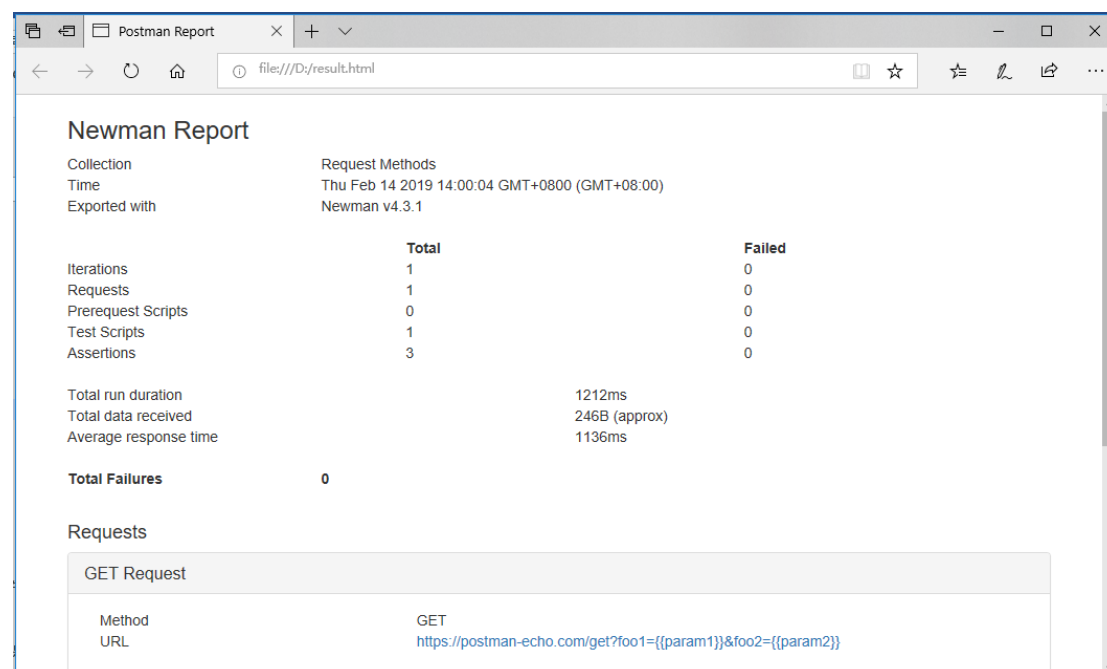


图 10-10 html 结果报告

3.2. 与持续集成工具 Jenkins 结合

平时做接口自动化，避免不了最后通过 Jenkins 做构建，Jenkins 可以用于搭建自动化测试环境，实现接口自动化测试在无人值守的情况下按照预定的时间调度运行，或每次代码变更提交至版本控制系统时实现自动运行的效果。

3.2.1. Jenkins 搭建

(1)访问 <https://jenkins.io/download/>，选择 Windows 平台下的安装包，如图 8-1 所示。

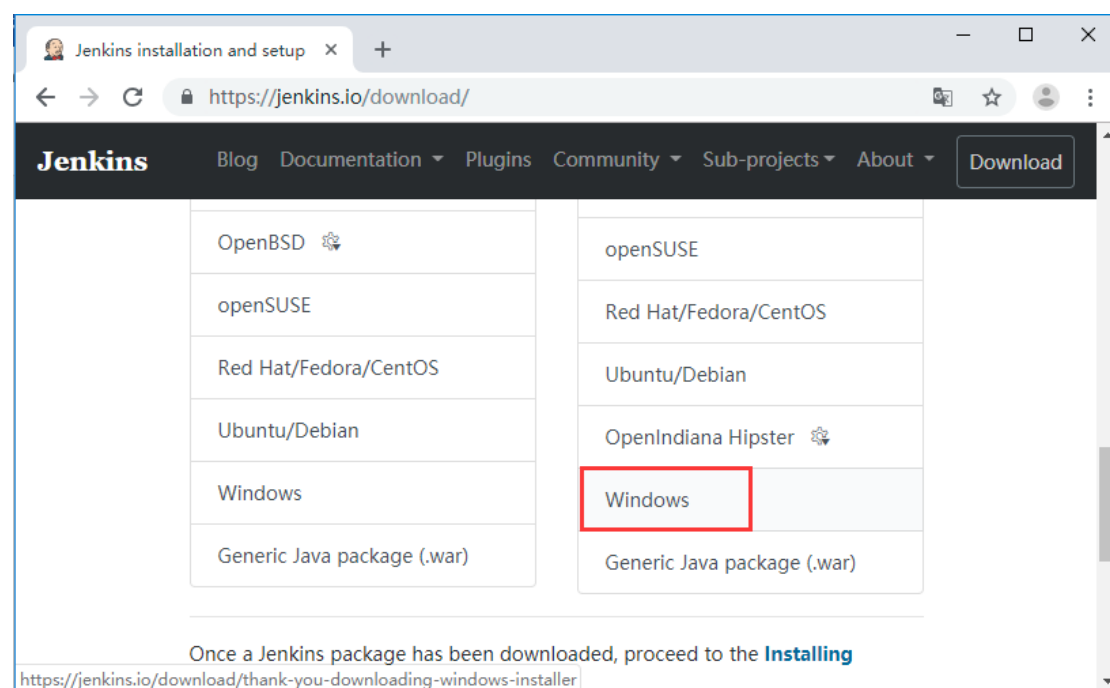


图 8-1 选择 Windows 平台的安装包

(2) 单击图 8-1 中的 Windows 进行下载，下载后的文件名为“jenkins-2.154.zip”，解压后双击该文件开始安装，在这里我们选择默认路径安装，如图 8-2 所示。

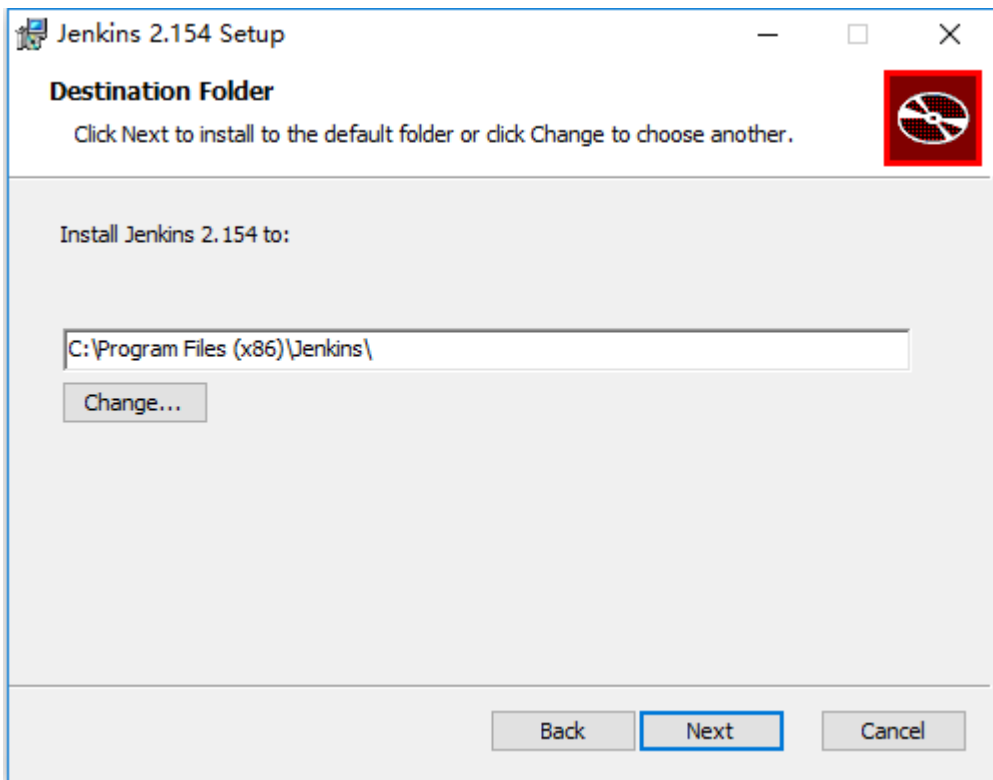


图 8-2 开始安装界面

(3) 单击图 8-2 中【Next>】按钮，开始安装，如图 8-3 所示。

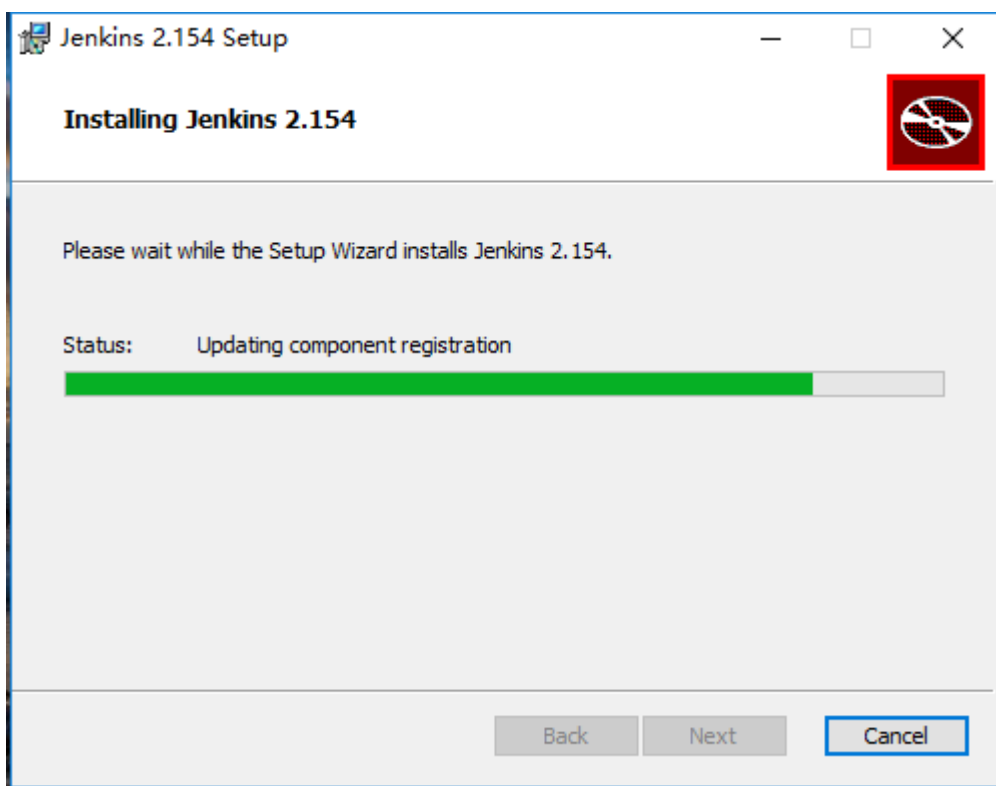


图 8-3 安装界面

(4) Jenkins 的安装进度非常快，安装成功后的界面如图 8-4 所示。

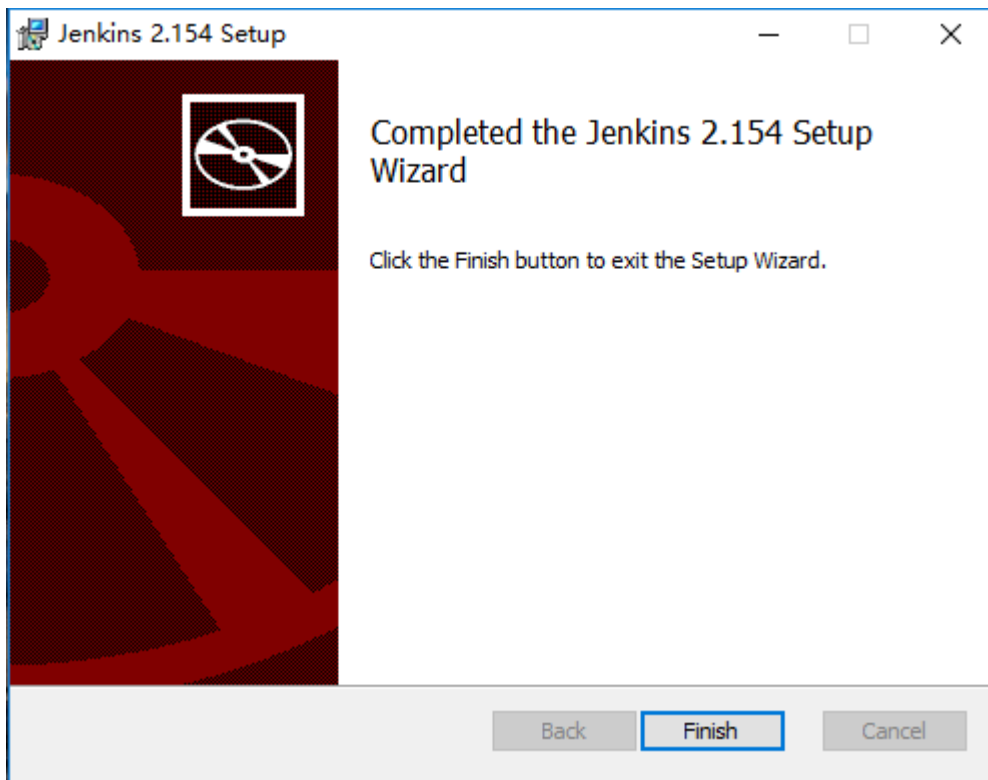


图 8-4 安装成功的界面

安装完成后，在浏览器地址栏输入 <http://localhost:8080/>，即可访问 Jenkins 界面,如图 8-5 所示，这个时候，我们需要获取 jenkins 自动生成的初始密码进行登录，密码文件地址如下：

C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword

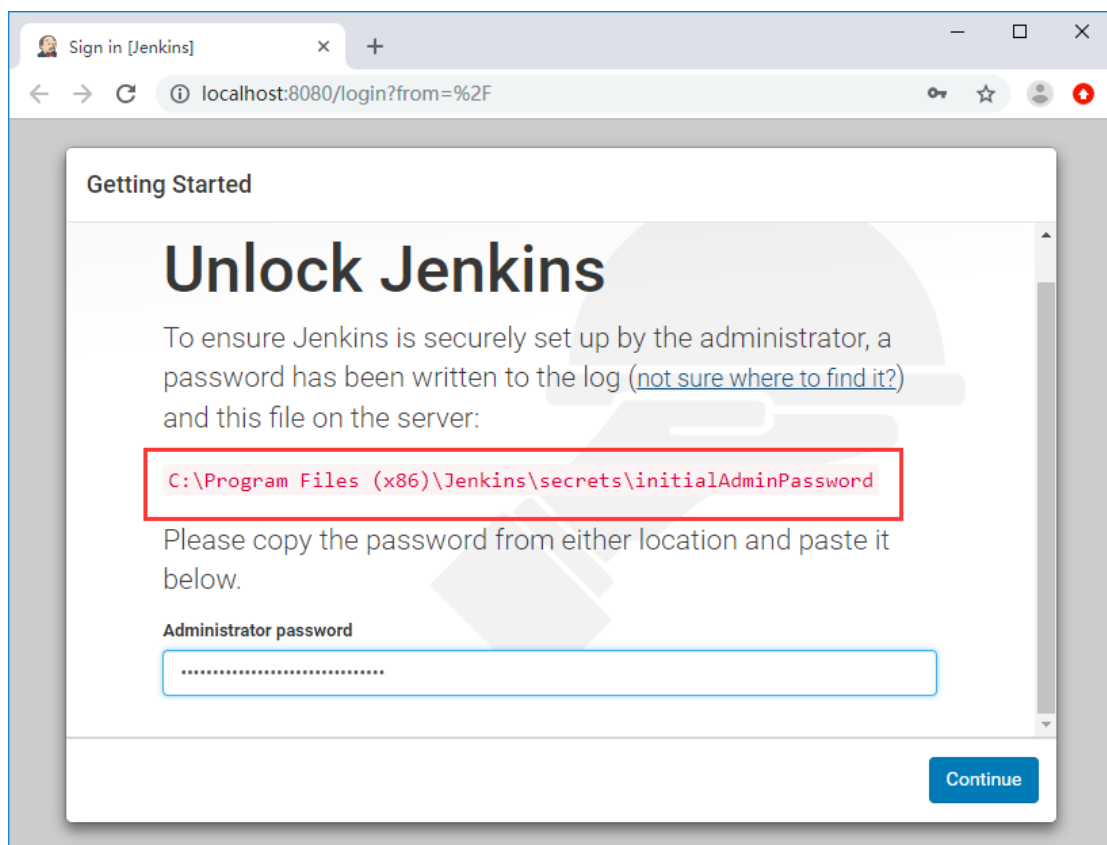


图 8-5 初始密码输入界面

(5) 在这里我们选择常见插件进行安装，如图 8-6 所示。

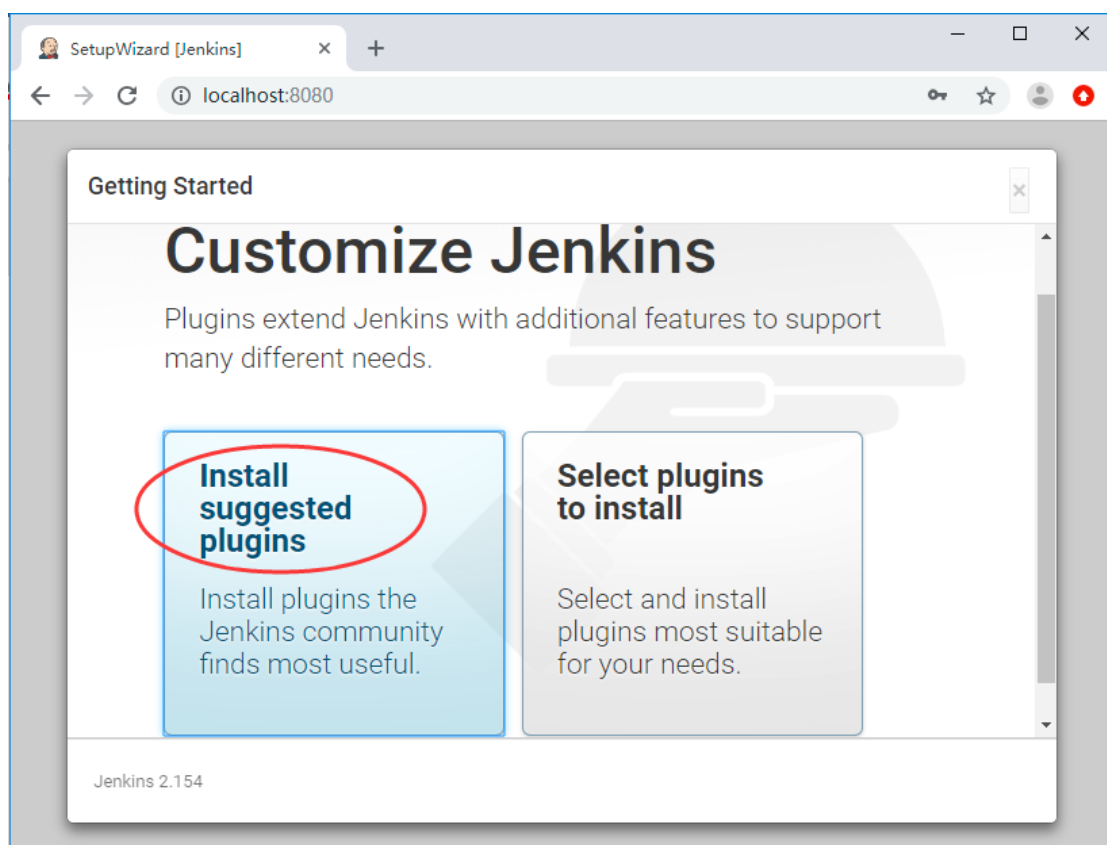
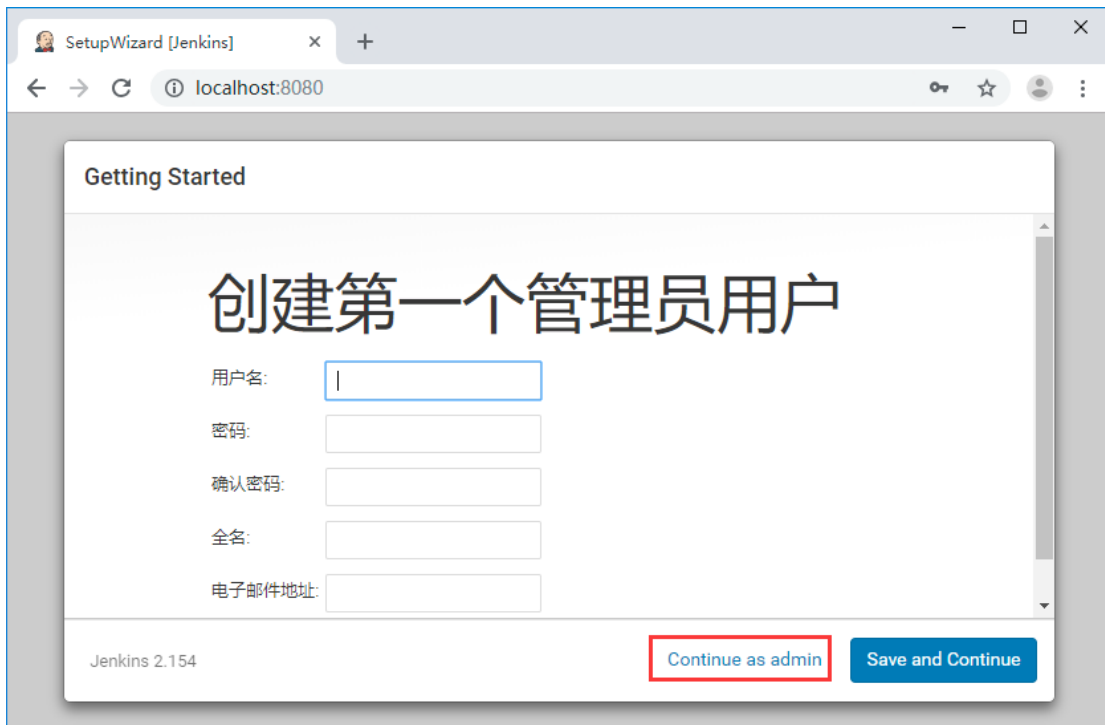


图 8-6 插件安装界面

(6) 插件安装完成后，进入创建管理员用户界面，如图 8-7 所示，目前我们暂不创建管理员，点击【Continue as admin】，以 admin 用户继续进行。



The image shows a web browser window with the title 'SetupWizard [Jenkins]'. The address bar shows 'localhost:8080'. The main content area is titled 'Getting Started' and features a large heading '创建第一个管理员用户' (Create the first administrator user). Below the heading are five input fields: '用户名:' (Username), '密码:' (Password), '确认密码:' (Confirm Password), '全名:' (Full Name), and '电子邮件地址:' (Email Address). At the bottom right, there are two buttons: 'Continue as admin' (highlighted with a red box) and 'Save and Continue'. The bottom left corner of the page displays 'Jenkins 2.154'.

图 8-7 创建管理员界面

(7) 在实例配置中，如图 8-8 所示，点击【Save and Finish】，相关 jenkins 配置已经完成。

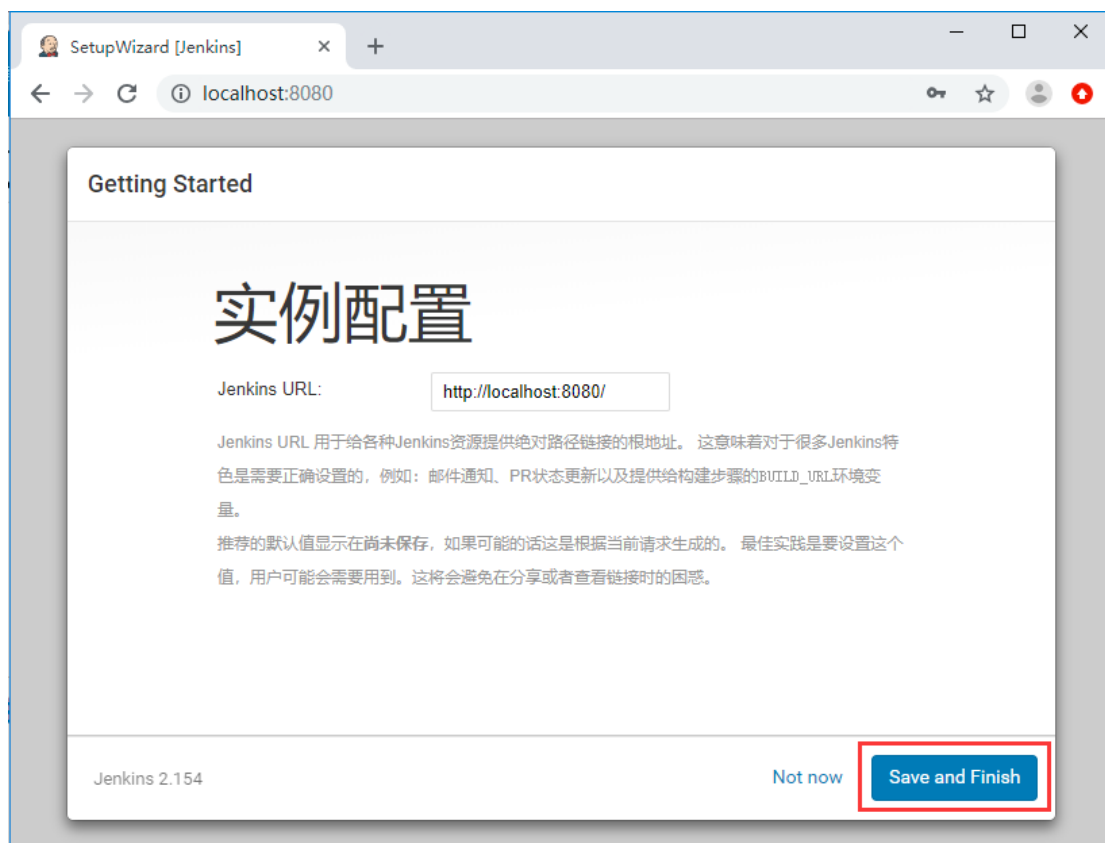


图 8-8 实例配置

(8) 安装成功后的界面如图 8-9 所示。

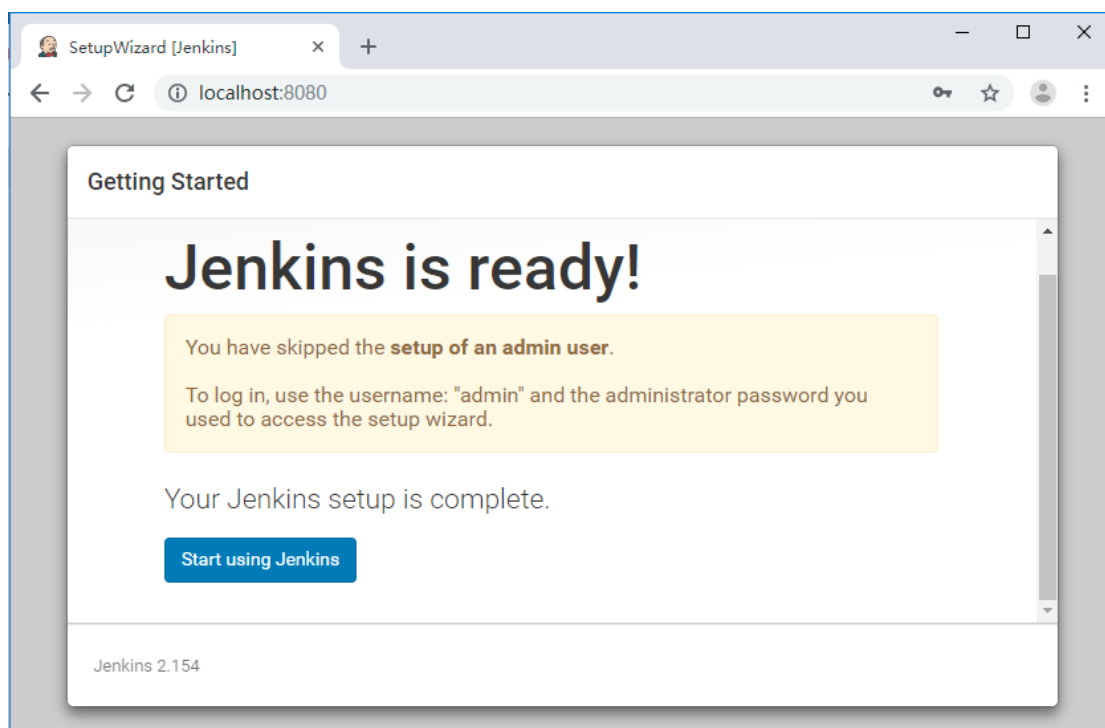


图 8-9 安装成功

(9) 最后，点击图 8-9 的【Start using Jenkins】，则会以 admin 用户登录 Jenkins，如图 8-10 所示。



图 8-10 安装成功

登录成功后，大家需要注意 admin 账户的密码目前还是默认初始密码，你可以在【系统管理】→【管理用户】→【用户列表】→【admin】→【设置】中进行密码的修改，如图 8-11 所示。

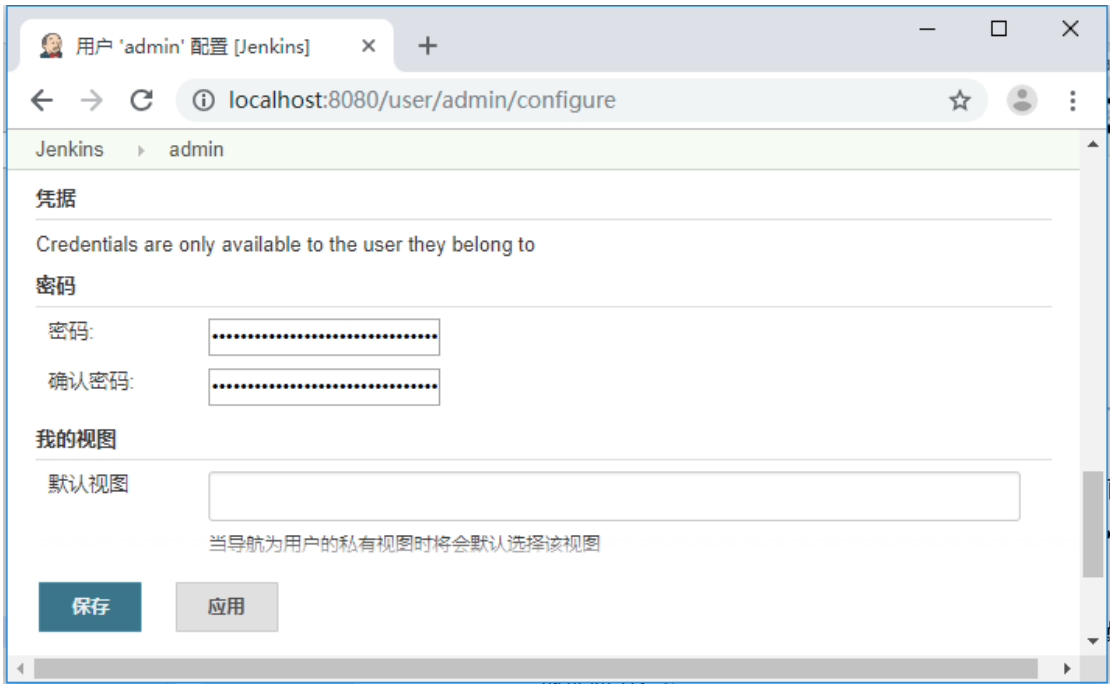


图 8-11 admin 用户修改密码

目前我们已经成功安装了 jenkins 平台，接下来将一步步构建任务，实现自动化测的定时执行。

3.2.2. 新建 job

任务(Job)是 Jenkins 的一个执行计划, 目前我们执行测试, 有两种方式, 通过 Pycharm 执行, 或者在命令行执行, 下面我们要一步一步达到测试脚本定时执行的目标。

(1) 新建一个 job, 如图 8-12 所示。



图 8-12 新建 job

(2) 选择构建一个自由风格的软件项目, 并起名为 “api_test_project”, 如图 8-13 所示。



图 8-13 job 信息

通过上面的操作，我们已经成功创建了一个任务(Job)，接下来需要通过配置使 job 完成下面几件事情。

- 通过 newman 运行接口测试用例
- 定时执行接口测试
- 生成并展示测试报告；

3.2.3. 执行 dos 指令

接下来，需要在构建模块下，dos 命令行来运行接口。

- (1) 添加构建步骤→“执行 Windows 批处理命令”，如图 8-17 所示。

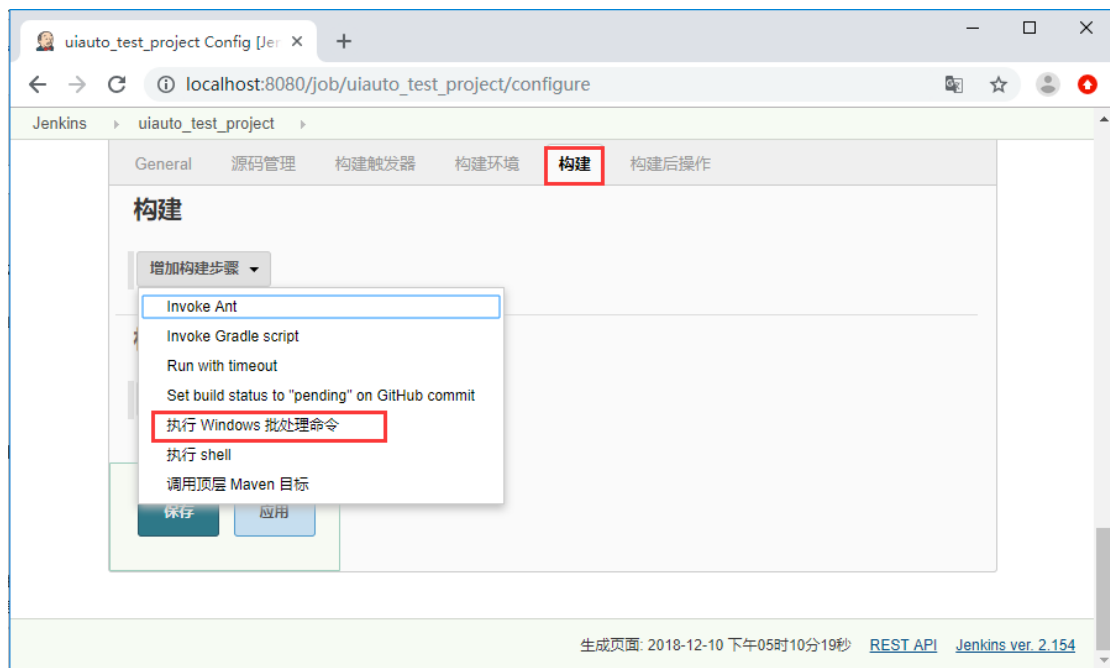


图 8-17 添加构建步骤

(3) 配置运行接口脚本的指令，如图 8-18 所示。

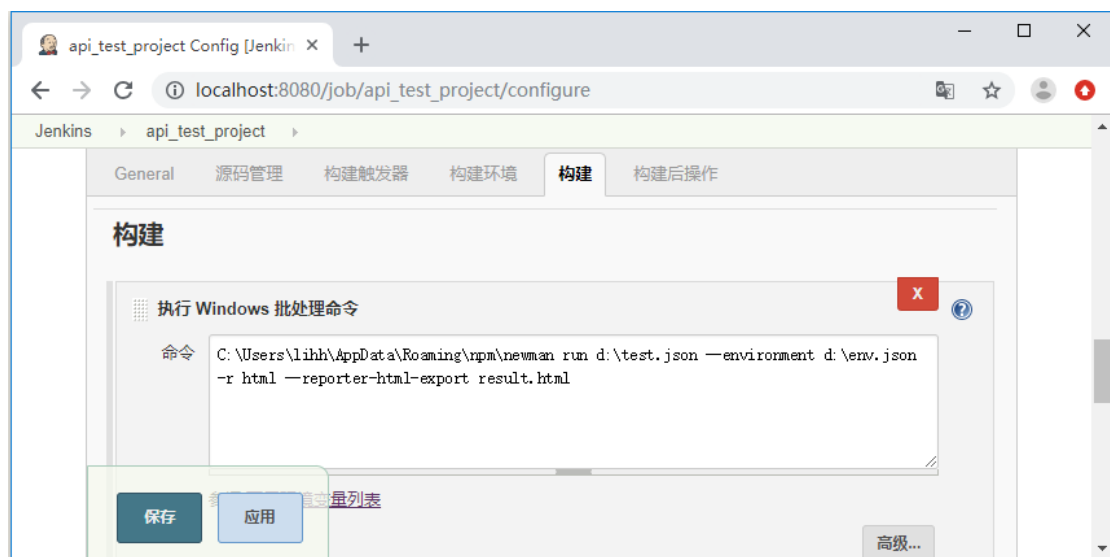


图 8-18 执行 windows 批处理命令

在上面的代码中，大家需要注意的是，我们更改了测试报告的存放路径，将结果存在该构建任务的目录下，即 Jenkins(安装路径)\workspace\api_test_project 下，为后面 jenkins 读取 html 报告做准备。

(4) 配置完成后，此时返回 Jenkins 工作台，找到该任务，点击“立即构建”，就可以一键运行所有的测试代码，当构建完成后，可以在文件中查看测试报告，如图 8-19 所示。



图 8-19 一键构建 job

注意，在这个过程中，如果出现构建失败的情况，需要通过查看控制台的报错信息，来定位错误原因，如图 8-20 所示。

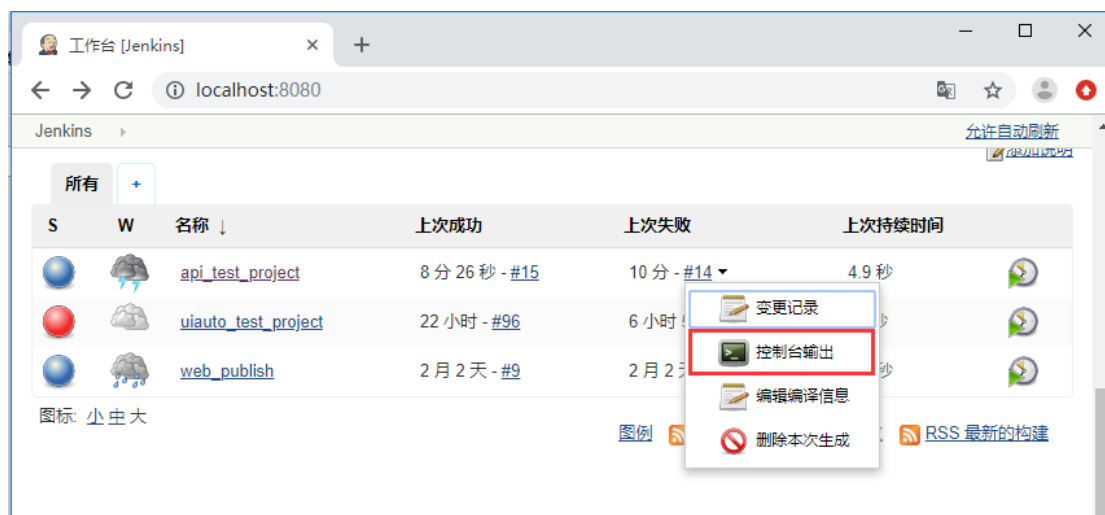


图 8-20 打开控制台

从控制台输出中，我们可以了解构建过程，在 jenkins 的安装路径的本任务目录下执行命令，在这里大家要注意，newman 一定要写完整的地址，不然会报如图 8-21 所示的错误：'newman' 不是内部或外部命令，也不是可运行的程序或批处理文件。



图 8-21 控制台信息显示

3.2.4. Jenkins 定制构建

跑自动化用例每次用手工点击 jenkins 触发自动化用例比较麻烦，我们希望能达到每天固定时间跑，坐等收测试报告的结果。

Jenkins 通过 5 颗星的语法结构表示时间，具体语法如下：

第一颗*表示分钟，取值 0~59

第二颗*表示小时，取值 0~23

第三颗*表示一个月的第几天，取值 1~31

第四颗*表示第几月，取值 1~12

第五颗*表示一周中的第几天，取值 0~7，其中 0 和 7 代表的都是周日

- 每 30 分钟构建一次：H/30 * * * *
- 每 2 个小时构建一次 H H/2 * * *
- 每天早上 8 点构建一次 0 8 * * *
- 每天的 8 点，12 点，22 点，一天构建 3 次 0 8,12,22 * * * （多个时间点，中间用逗号隔开）

3.2.5. 构建触发器

假如我们要在每天 9 点，17 点，朝九晚五各构建一次，如图 8-22 所示。

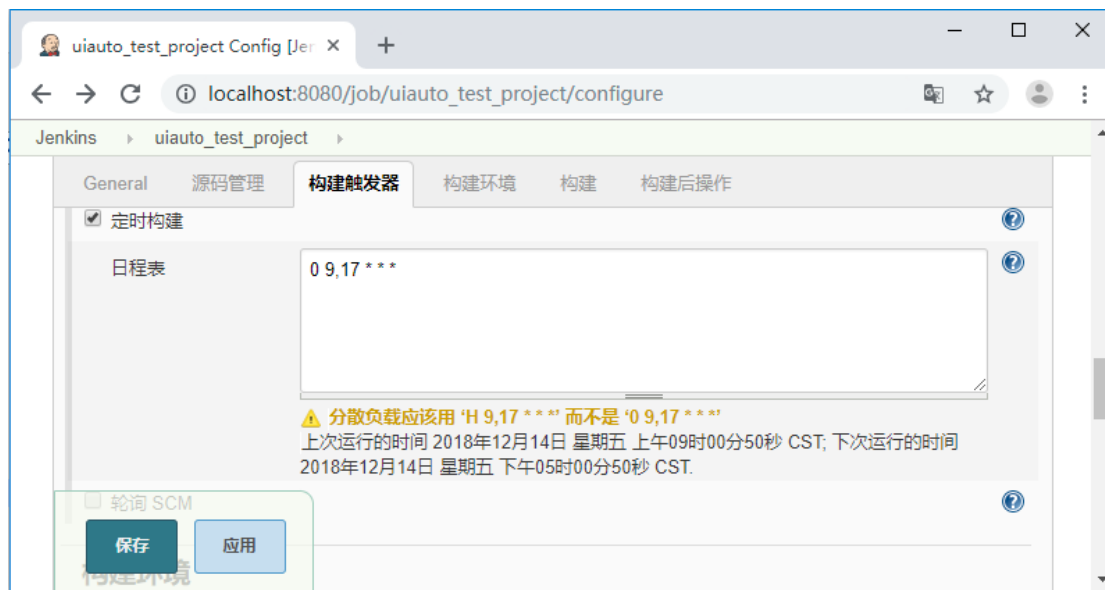


图 8-22 定时构建

注意，这时会推荐用 H 9, 17 * * * 语法，我们可以进行更改，如图 8-23 所示。



图 8-23 定时构建

3.2.6. Job 关联

举个案例场景，比如我下面 Job1 是 web 项目打包并发布的构建任务，我想每次打包发布后，然后触发接口测试 Job2 的构建。

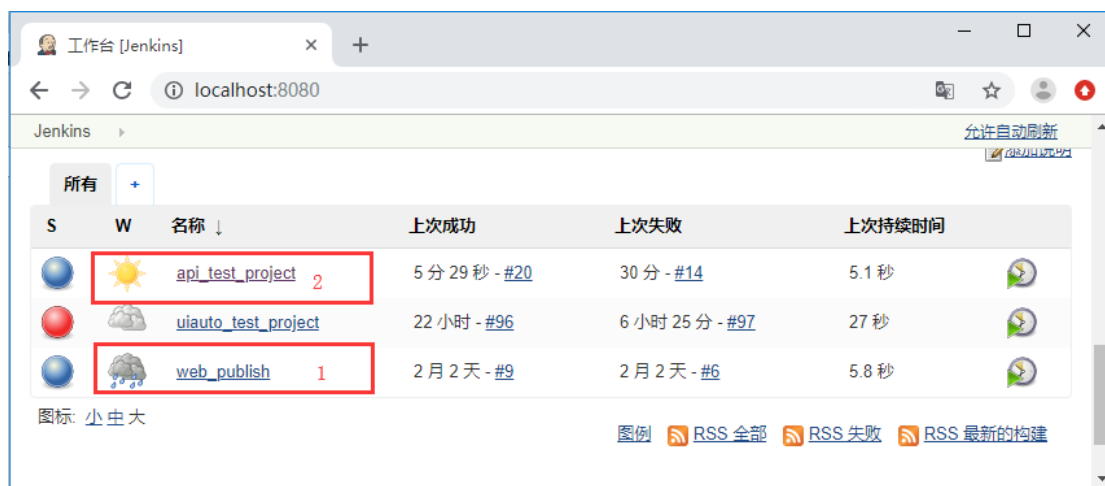


图 8-24 job 关联

(1) 构建触发器勾选 Build after other projects are built, Projects to watch 输入 Job1 的名称（这里可以输入多个依赖的 jobs，多个 job 中间用逗号隔开）

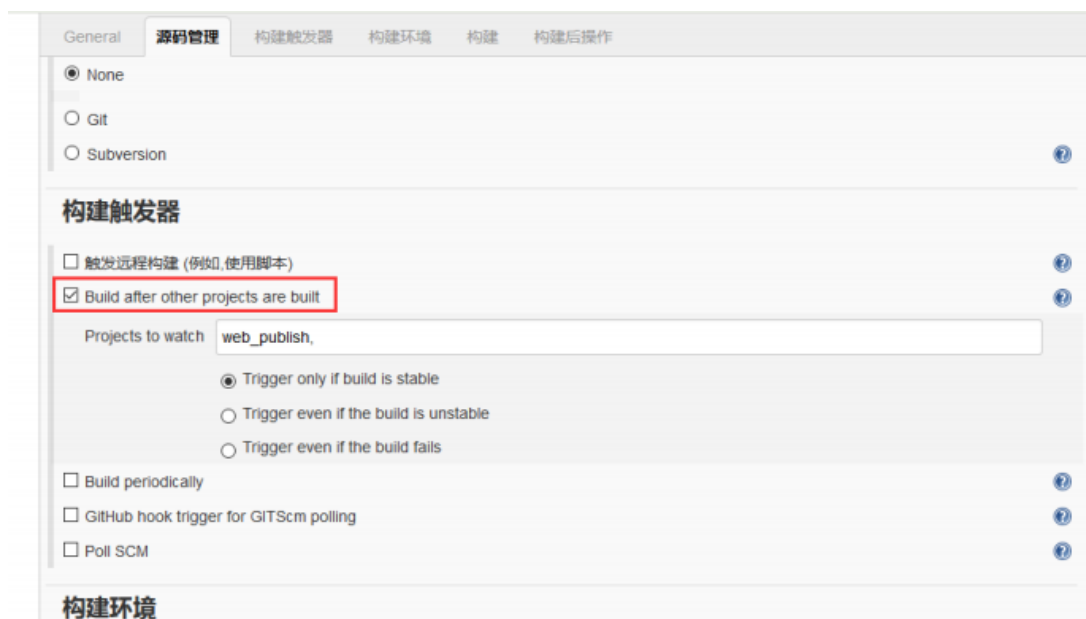


图 8-25 事件触发

(2) 下面有三个选择，一般默认第一个就行 Trigger only if build is stable: 构建稳定时触发 Trigger even if the build is unstable : 构建不稳定时触发 Trigger even if the build fails : 构建失败的时候触发。

上面设置好后，启动第一个 Job 完成后，就能接着启动第二个 Job 了

3.2.7. 添加 HTML Publisher 插件

在 Jenkins 上展示 html 的报告，需要添加一个 HTML Publisher plugin 插件，把生成的 html 报告放到指定文件夹，这样就能用 jenkins 去读出指定文件夹的报告了。

(1) 执行完测试用例后，可以用“添加构建后操作步骤”，读出 html 报告文件。

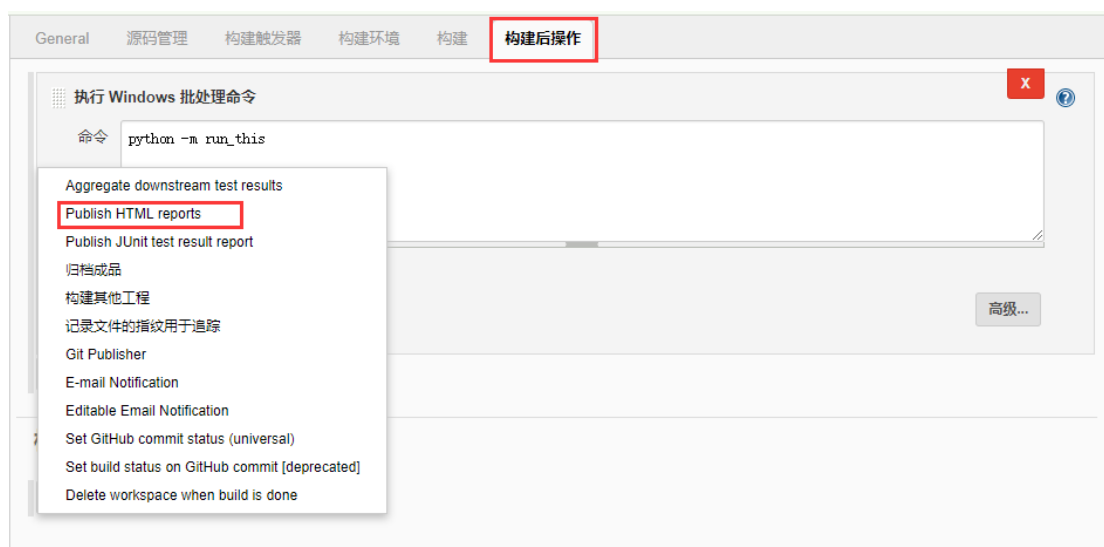


图 8-26 构建后的操作

注意：如果你的展开后有 Publish HTML reports 这个选项就能直接添加了，没有就看下一步如何添加插件。

(2) 添加 HTML Publisher plugin 插件，首先打开系统管理→插件管理，在【可选插件】页面，在右上角搜索需要安装的插件：HTML Publisher，如图 8-25。



图 8-27 搜索 HTML Publisher 插件

(3) 勾选后直接安装，安装完之后重新启动 Jenkins。

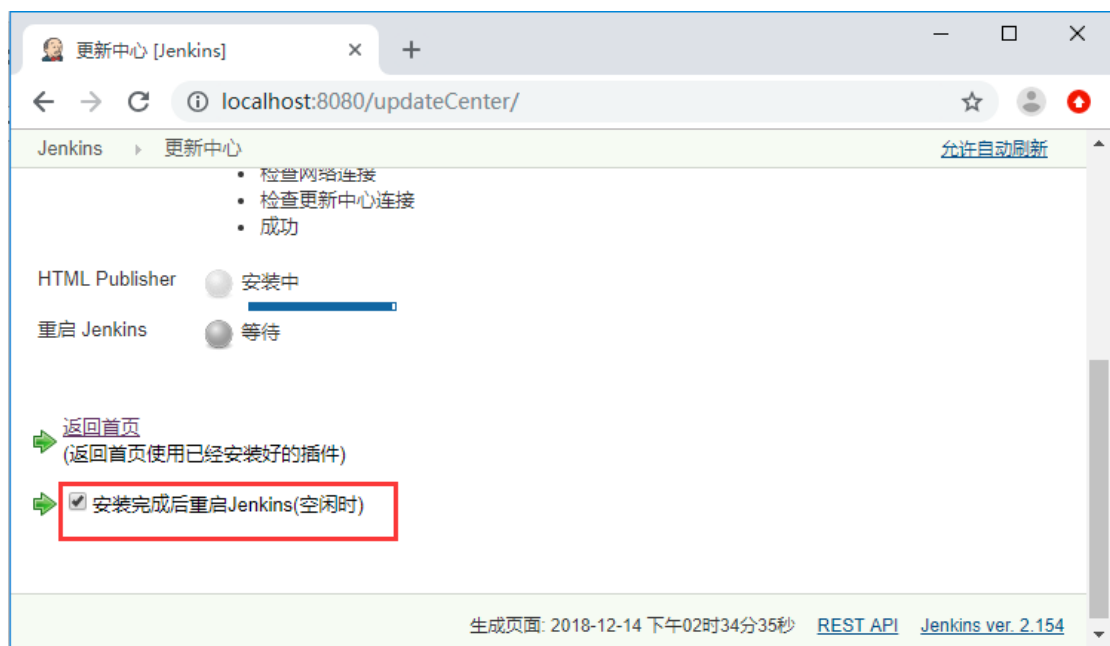


图 8-28 重启 jenkins

3.2.8. 添加 reports

(1) 安装完插件，并且重启 jenkins 后，在构建后的操作中，会看到【Publish HTML report】选项，如图 8-29 所示。

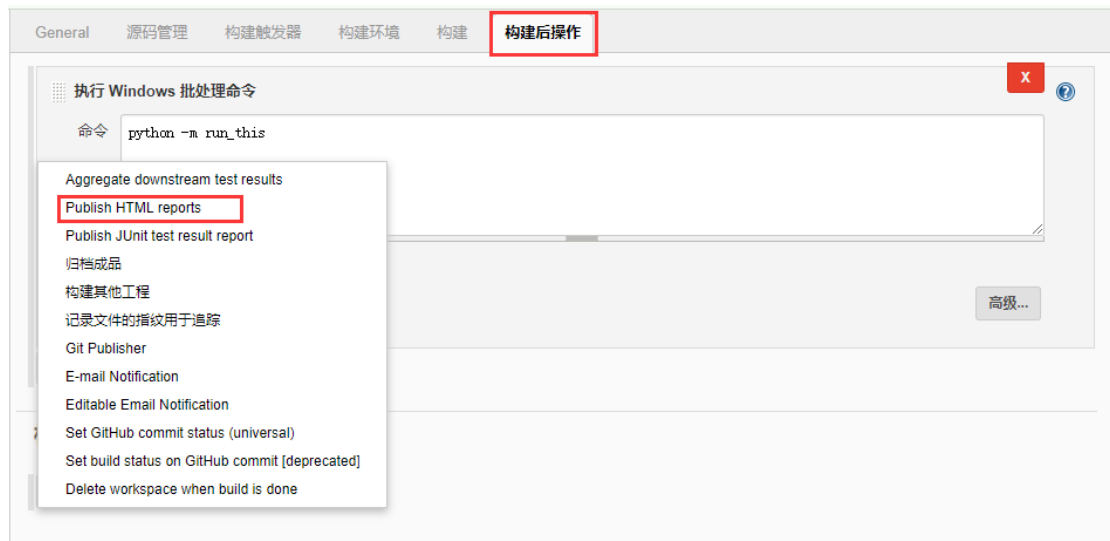


图 8-29 添加 reports

(2) 点开 Reports 后，界面显示如图 8-30 所示。

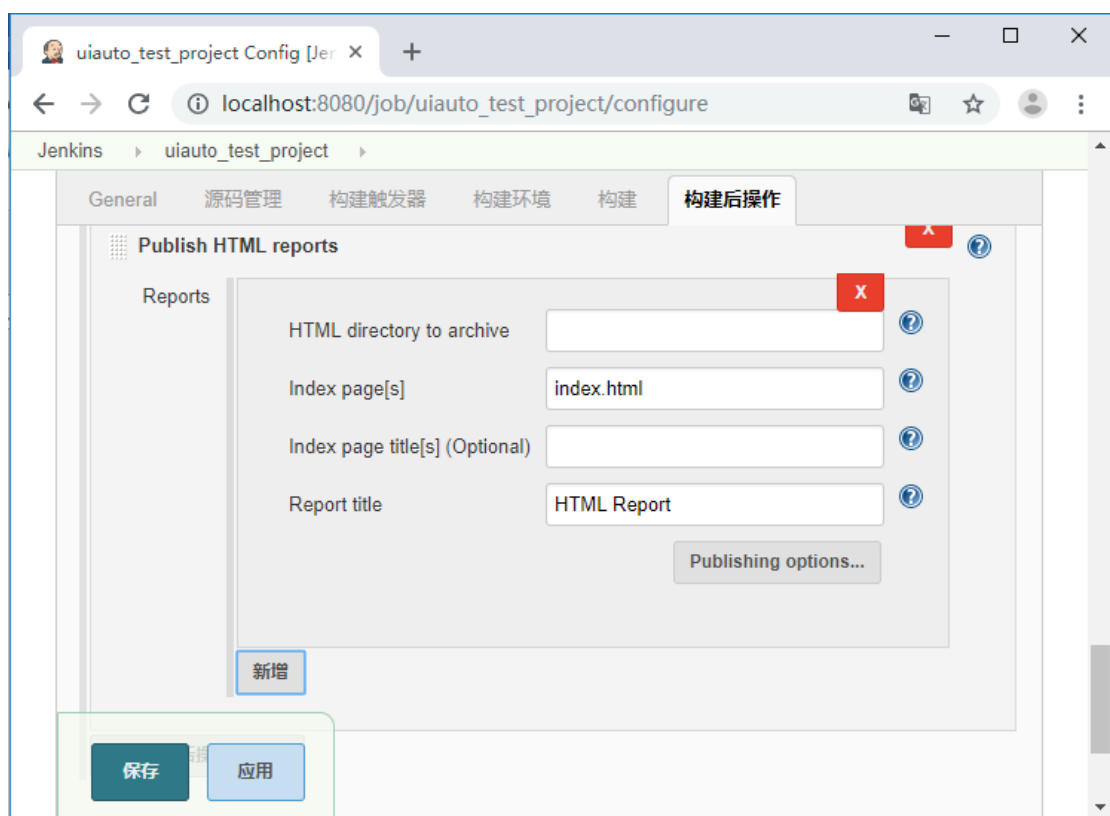


图 8-30 report 展示

其中 HTML directory to archive: 是以 workspace\api_test_project 作为参照路径，报告所存放的路径。

Index page 是运行完脚本后所生成的测试报告的名称。

Report title: 显示在 jenkins 上的名称，默认 HTML Report 就行，我们所生成的报告路径如图 8-30 所示。

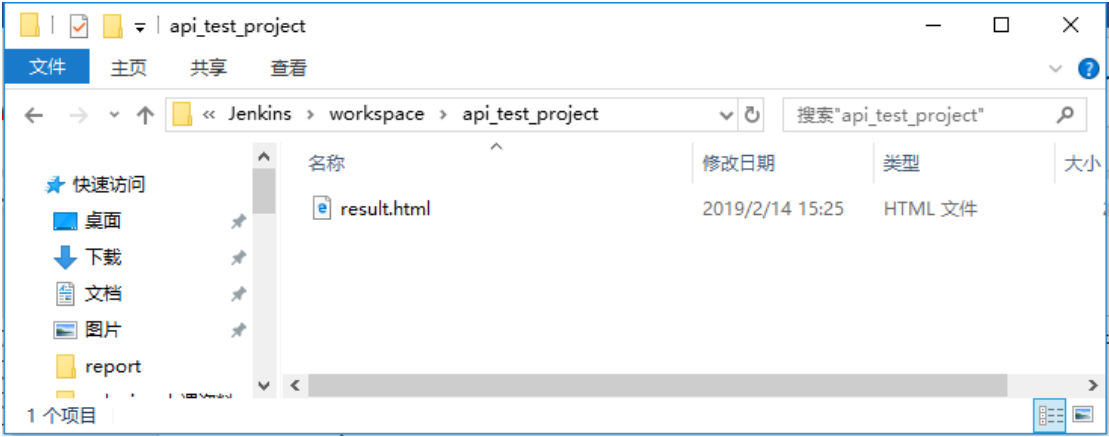


图 8-30 report 相对路径

(3) 结果报告的最终配置如图 8-31 所示，点击应用。

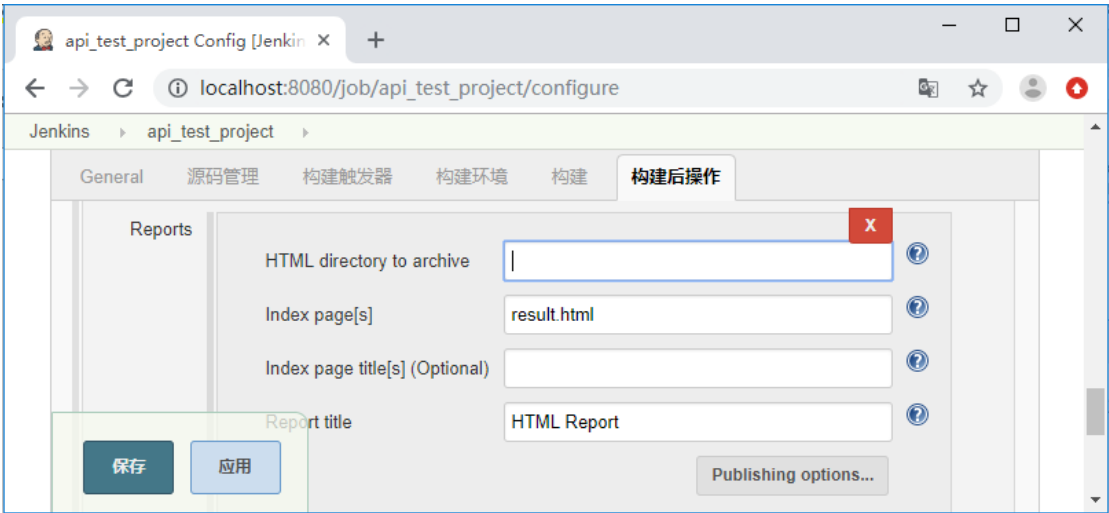


图 8-31 report 结果配置

3.2.9. 报告展示

(1) 运行完之后，在左侧工程下会生成一个 HTML Report 目录。



图 8-32 report 结果配置

(2) 点开查看详情，如图 8-33 所示。

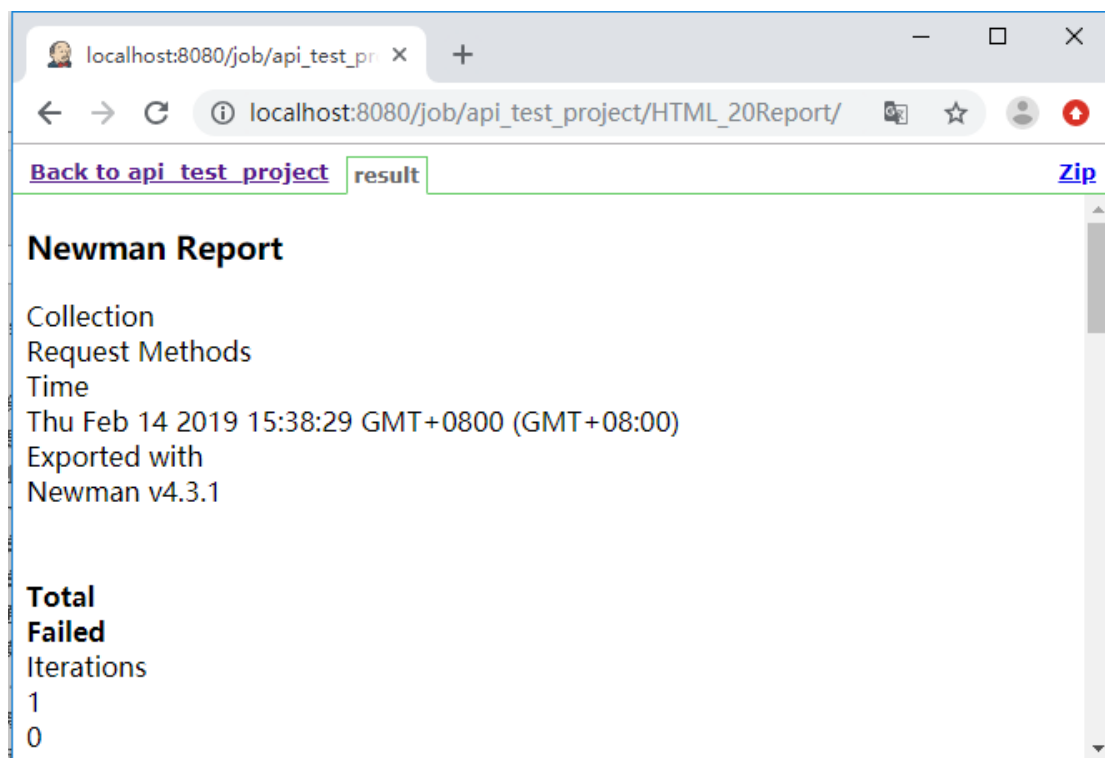


图 8-33 测试结果页面

点开，正常情况应该跟本地 html 浏览器打开是一样的。这里 html 格式丢失了，因为 jenkins 默认没有加载 css 样式。

3.2.10. jenkins 上 html 展示问题

我们遇到了展示出来的 HTML 没有正常加载相关的 CSS 和 JS，这是 Jenkins 的一个安全策略，只允许加载 Jenkins 服务器上托管的 CSS、图片，以防止 Jenkins 用户恶意 HTML/JS 文件的攻击。

可以通过下面的操作来解决这个问题。

(1) 打开系统管理→脚本命令行，输入如下命令，点击【运行】。

```
System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")
```

(2) 重新构建任务，在 jenkins 下查看结果报告，可以看到正常展示的 HTML 结果报告，如图 8-34 所示。

localhost:8080/job/api_test_pr

+

←

→

↺

localhost:8080/job/api_test_project/HTML_20Report/

🔍

☆

👤

+

Back to api_test_project

result

Zip

Mean time per request

1030ms

Mean size per request

362B

Total passed tests

3

Total failed tests

0

Status code

200

Tests

Name	Pass count	Fail count
Status code is 200	1	0
response body contains bar1	1	0
response host	1	0

图 8-34 测试报告展示

3.1. 本章小结

本章在接口基础测试的基础上，构建 postman+newman+jenkins 接口测试框架，能够达到定时批量执行所有的接口测试用例，也可以达到当被测系统更新版本后自动触发接口自动化测试的执行。

接口自动化测试在企业中应用广泛，有多种实现方式，使用本章框架时一种易上手的方式，你也可以通过 python 编写一套自动化测试框架，本章只作为基础章节向大家展示一种实现方式。