



CHAPTER

03

第3章

Web UI 自动化基础

在正式介绍 Web UI 自动化测试相关章节之前，针对一些初学者而言还需要额外补充下相关基础知识。这些基础知识不仅是 Web UI 自动化的知识，同时也是 Web 开发的基础知识。掌握了这些基础知识之后，在后续的自动化脚本开发时才能更好地发现问题和解决问题。

本章主要介绍的内容包括 HTML、DOM、CSS 等 Web 相关技术。

3.1 HTML 与 DOM 简介

HTML 全称为超文本标记语言，是网页制作与 Web 开发所使用的语言。其特点是简单易学、平台无关且通用性较好。HTML 的标准是由 W3C 组织提出的，目前最新的规范是 HTML5。

HTML 不是编程语言，而是一个标记语言；它有自己的一套标记标签，并且使用这些标记标签来描述网页的内容。浏览器接收到这些标记文本后，按照预定的行为来解析并展示到页面上。HTML 标签的规则如下。

- 由尖括号包围关键字组成，例如 <html>。
- 通常都是成对出现，例如 <div> 和 </div>。
- 成对出现时第一个为开始标签，第二个为结束标签。
- 有时会是单标签，例如 <input name="kw"/>。
- 标签可以嵌套但不能交叉嵌套。

- 不同的标签有它自己的功能和定义，例如，**** 标签可以使文本加粗。

HTML 页面就是由这些特定含义的标签对与文本内容所组成的，开发人员会根据不同的需求选用对应的标签来开发网页。下面就是一个最简单的 HTML 页面源码。

```

<html>                                # 网页根标签
  <head>
    <title> 测试页面 </title>
    <meta charset="utf-8">
  </head>
  <body>                                # 网页主体标签
    <h1> 我是一级标题 </h1>                # 标题标签
    <p> 我是一个段落。 </p>                  # 段落标签
  </body>
</html>

```

上述的 HTML 源码在浏览器中被加载之后的效果如图 3-1 所示。

从上述代码中可以看出，HTML 的基本结构是 **<html>** 与 **</html>** 作为最外层标签，其下有 **<head><body>** 标签。**<head>** 标签下可以包含 **<title><meta>** 等标签。**<body>** 标签下则可以包含更多的 HTML 标签，例如 **div**、**p**、**table**、**input** 等。

其中，**<html><head><body>** 这三者的结构是固定的，并且 **<head>** 标签下的内容也基本为固定形式。只有 **<body>** 标签下内容与形式不是固定的，因为 **<body>** 中的内容是需要在页面上显示的元素；而不同的页面显示内容各有所需，所以 **<body>** 下的内容与形式会根据需要定制成各种形式。具体而言，**<body>** 下的标签可以分为如下几种类型。

- **格式标签：**用于规范页面格式显示的标签。例如，块区域标签 **div**、段落标签 **<p>**、换行标签 **
** 等。
- **文本标签：**用于约束文本内容显示的标签。例如，**** 加粗标签、**<i>** 斜体标签、**** 字体标签等。
- **图像标签：**用于显示图片的标签。主要为 **** 标签。
- **超链接标签：**用于连接和跳转页面的标签。通常为 **<a>** 标签。
- **表格标签：**用于回显表格内容的标签。通常为 **<table>** 标签。

除了上述举例的这些标签之外，HTML 还有很多同类型不同功能的标签。想要全部了解的读者可以参见 HTML 规范 http://www.w3school.com.cn/tags/html_ref_func.asp。

在正常的网页开发过程中，就是使用各种不同功能的 HTML 标签来组装成我们需要的页面内容。而浏览器在接收到 HTML 内容后并不是直接显示 HTML 本身，而是显示了其对应的网页效果。这是因为浏览器自身拥有解析 HTML 并渲染页面的功能，在渲染页面的过程

我是一级标题

我是一个段落

图 3-1 HTML 效果

中，浏览器还会创建一个叫作 DOM 的对象。

DOM 全称为文档对象模型，是 W3C 组织推荐的处理可扩展语言的标准编程接口。它是浏览器在解析 HTML 页面的过程中生成的一个内部对象。在 DOM 对象中把页面（或文档）的对象都组织在一个树状结构中，其树状结构中的每一个对象都是与源 HTML 中的节点一一对应的。例如，前段代码中的 HTML 内容其对应的 DOM 对象结构如下。

```
|--html
  |
  |--head
  |   |
  |   |--title
  |   |--meta
  |
  |--body
    |
    |--h1
    |--p
```

浏览器最终在渲染和显示网页内容的时候正是基于 DOM 对象的内容而来的，并且 DOM 对象一旦被修改浏览器将会重新渲染页面内容。而另一方面 DOM 本身就是一个可编程的接口，即我们可以通过编程的方式调用它。简而言之，我们可以通过编程来动态改变页面显示的效果。

大部分基于 Web 的自动化测试工具都是通过操作 DOM 来控制浏览器行为的；而我们在进行自动化测试脚本开发的时候，其中一个重要的知识点就是如何定位 DOM 中的元素；在定位到元素之后自动化工具就可以对其 DOM 节点进行相关操作，来实现自动化测试 Web 页面的效果。接下来将学习如何定位一个 Web 元素。

3.2 学习元素定位方式

Web 的 UI 自动化测试步骤有元素定位、元素操作、再次元素定位、元素信息获取、结果检查等这几个步骤。其中，元素定位是奠定整个测试过程的基础，如果找不到元素我们既不能操作页面，也不能获取页面信息，自然就无法进行自动化测试。因此元素的定位技术是我们学习 Web 自动化测试所必须掌握的一项技能。

所谓的元素定位，即根据元素的特定属性对元素进行定位描述的一项技术。也就是说，对元素进行定位时需要先了解元素具有哪些属性，然后依据这些属性就可以编写出针对该元素的定位符。那么元素具体有哪些属性呢？下面列出了在 Web 自动化测试过程中经常用来进行元素定位的常用属性。

- ID 属性。能唯一定位一个元素的属性，优先选取的定位属性。

- Class 属性。广泛使用的定位属性。
- Name 属性。
- TagName 属性。HTML 节点的标签名，如 input、a 等。

图 3-2 中标注出每一个属性在 HTML 页面中的具体形式。

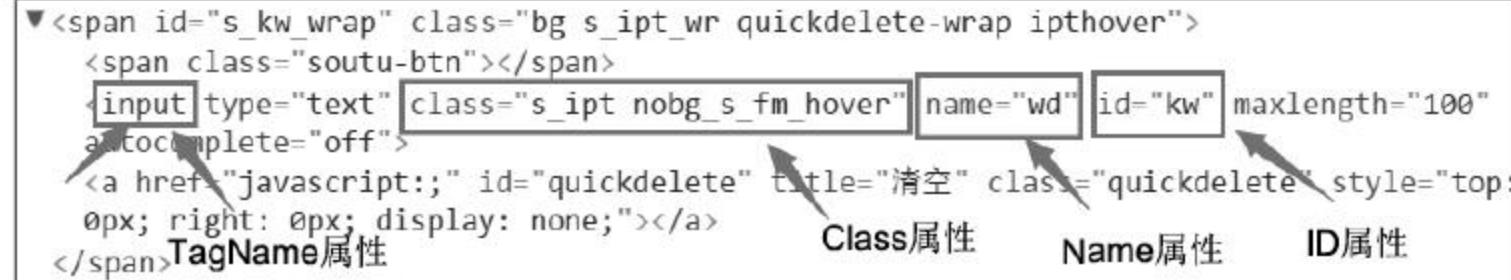


图 3-2 HTML 属性分析

这些属性中只有 ID 属性的值在 HTML 文档中是唯一的，而其他属性的值在 HTML 文档中可能会出现一次以上。因此，通过 ID 属性找到的元素都是非常精准的，而其他属性定位元素时有可能会匹配到多个目标元素。例如，某页面中 class 值为 fly 的元素有两个，这种情况下如果使用 class=fly 的属性来进行定位则会匹配到两个元素，导致元素定位无法精确匹配从而影响后续测试工作，因为不能保证定位到的那个元素就是我们真正需要操作的元素。

提示 虽然只有 ID 属性是唯一的，但是我们仍然可以考虑使用 Class、Name 属性进行元素定位，因为并不是所有的 Class 和 Name 属性都有多个值存在，我们通过在 HTML 页面中搜索一下指定的 Class 或者 Name 属性，如果只搜索到一个结果，那么该属性就可以唯一定位这个元素了。而 TagName 则不建议单独用来进行定位，因此 TagName 在页面中只出现一次的可能性很小。

上面提到的是利用元素自身基本属性来定位，而除了这些基本属性之外，还可以通过其他技术手段来进行定位，目前业内普遍使用的定位技术为 XPath 和 CSS。这两项定位技术的特点是支持的定位方式更多元化，支持的功能更强大，几乎可以唯一定位任何一个元素，可以轻松解决一些日常工作中的常见问题，具体如下。

- 匹配多个元素时可以通过 Index 获取指定元素。
- 可以使用更多的元素属性，例如，value、type 等。
- 可以综合使用多个元素属性，例如，同时使用 Class、Name 和 TagName 属性来定位。
- 可以分层进行定位，例如，先定位到一个确定的父节点元素，再定位可以唯一确定的子元素。

提示 本书推荐使用 CSS 的定位技术来进行元素定位学习，关于 XPath 的定位技术感兴趣的读者可以通过在线教程进行学习。

3.3 CSS 定位技术

其实 CSS 定位元素的基础依然是 HTML 文档中元素的属性，只是通过 CSS 语法我们可以组合成条件和层次都更加丰富的定位语句。在具体学习复杂的定位之前我们先来学习一下 CSS 的基本定位语法，如表 3-1 所示。

表 3-1 CSS 定位语法

定位资源	语法	样例	说明
ID	#ID	#kw	匹配 id=kw 的元素
Class	.Class	.fly	匹配所有 class=fly 的元素
TagName	Element	input	匹配所有的 input 元素
Attribute	[attribute]	[name]	匹配所有具有 name 属性的元素
	[attribute=value]	[name=su]	匹配所有 name=su 的元素

上面是 CSS 常用的基本语法，通过这些基础语法就可以组合出更多的定位语句。关于 Selenium 支持的 CSS 更多的定位语法可以访问 <http://www.testdoc.org> 进行学习。接下来针对前面所列出的几种复杂情况进行 CSS 定位。

- 获取匹配多个元素中的指定一个。

```
table>tr:nth-child(1)          ## 定位 table 元素下的第 1 个 tr 元素
```

- 使用更多的元素属性。

```
input[type=text][value=1]      ## 定位 type 为 text, value 为 1 的 input 元素
```

- 综合使用不同的属性。

```
input.fly[name=wd]           ## 定位 class 为 fly, name 为 wd 的 input 元素
```

- 分层定位元素。

```
form>table>a[class=dot]     ## 定位 form 下的 table 下的 class 为 dot 的 a 元素
```

- 定位特定文本内容的元素。

```
label:contains('userName')   ## 定位包含 userName 文字的 label 元素
```

通过上面的 CSS 定位示例语句，可以大体先了解下针对不同定位场景，如何利用 CSS 定位技术去解决。在后面会进一步通过代码进行练习和掌握。

3.4 使用工具帮助定位

通过上面的介绍掌握了如何在 HTML 页面中进行元素定位，方法虽然可行，但是每次都是手动地在 HTML 页面的源码里寻找，效率自然就会非常低，所以本节中介绍如何快速高

效地对目标元素进行定位，即使用定位工具来进行元素定位。

3.4.1 IE 的 Developer Tool

IE 的开发者工具可以通过单击元素的方式来定位页面上元素所对应的 HTML 节点，具体的步骤如下。

- (1) 打开 IE 浏览器。
- (2) 打开一个页面，如 <http://www.baidu.com>。
- (3) 右击页面选择“检查元素”或者按 F12 键打开开发者工具，如图 3-3 所示。



图 3-3 IE 开发者工具

- (4) 单击“选择元素”图标或者按 Ctrl+B 快捷键。
- (5) 将光标移动到页面上的目标元素上并单击。
- (6) 查看开发者工具有背景色的元素节点，即被单击元素的对应节点。

通过上面的步骤，可以轻松定位某个页面元素的对应 HTML 节点内容，而无须手动查看源码和查找关键字，既准确又快速。

3.4.2 Firefox 的 Web 开发者工具

同样地，Firefox 浏览器也提供了相似的工具，快捷键 F12 或者右击选择“查看元素”即可打开该工具，打开后的界面如图 3-4 所示。



图 3-4 FireFox 开发者工具

查看元素的方式与 IE 基本一致，首先单击“用鼠标选择元素”按钮 ，然后使用鼠标单击目标元素，工具中有背景色的节点即为被单击元素对应的 HTML 节点。

3.4.3 Chrome 的开发者工具

Chrome 浏览器中也提供了相应的工具，按 F12 快捷键或者右击选择“检查”命令即可打开该工具，使用方法同 IE 和 Firefox，具体界面如图 3-5 所示。

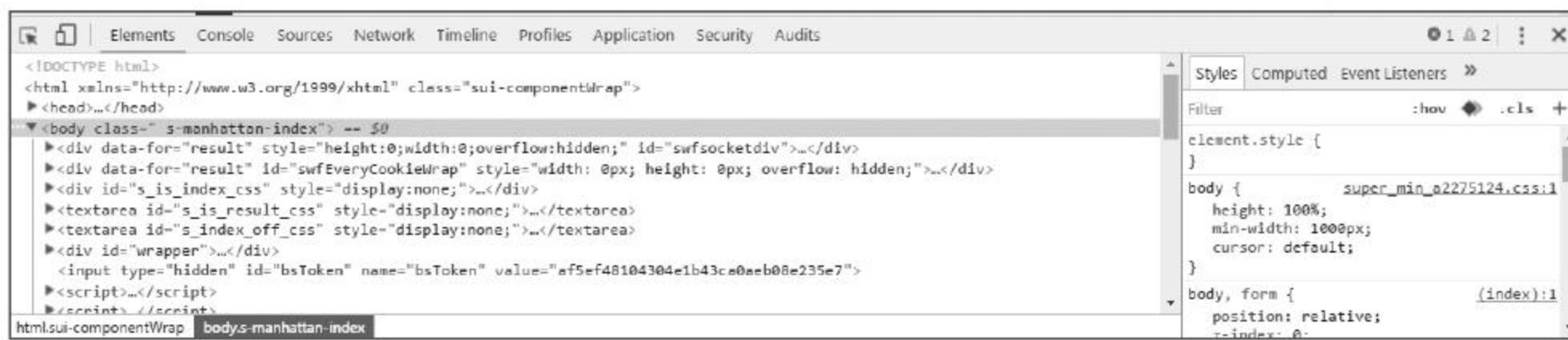


图 3-5 Chrome 开发者工具

3.4.4 Firefox 的 XPath Checker 插件

除了使用开发者工具来查看元素节点的属性，还可以通过 XPath 工具来查看元素的 XPath 路径，该工具可以自动生成被单击元素的 XPath 路径，为我们的元素定位提供了另一种方便。具体的安装和使用步骤如下。

- (1) 安装 Firefox 的 XPath Checker 插件。
- (2) 打开一个页面，如 <http://www.baidu.com>。
- (3) 右击目标元素后选择 View XPath 子项。
- (4) 查看 XPath Checker 界面中的 XPath 路径，如图 3-6 所示为百度首页搜索框的 XPath 定位路径。

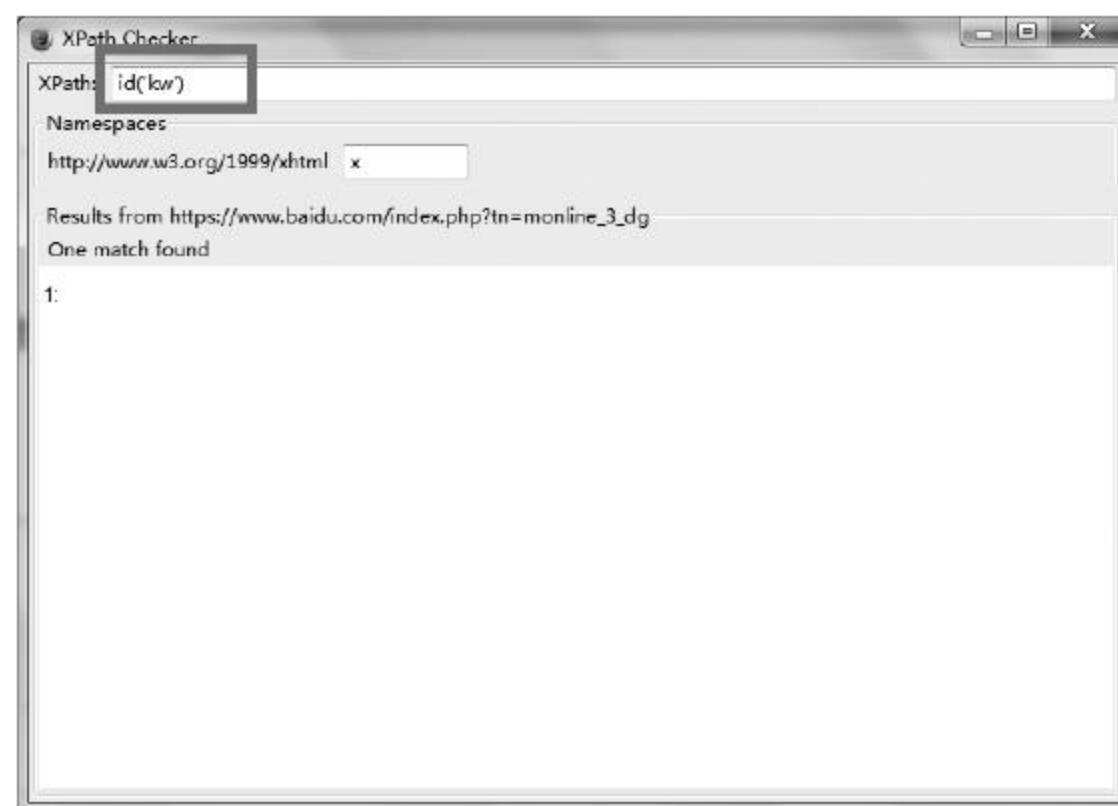


图 3-6 Firefox 下生成 XPath

3.4.5 Chrome 的 XPath 工具

Chrome 中也提供了 XPath 的定位工具，而且已经直接集成在它的开发者工具中了，获取元素 XPath 路径的具体操作步骤如下。

- (1) 打开开发者工具。
- (2) 定位到具体的元素节点。
- (3) 在开发者工具中右击该 HTML 节点。
- (4) 选择 Copy → Copy XPath，如图 3-7 所示。



图 3-7 Chrome 下生成 XPath

通过上面的步骤之后，元素节点对应的 XPath 路径就被复制到系统的粘贴板中了，可以通过粘贴的方式把 XPath 路径直接复制出来，同样对于百度首页的输入框我们得到的 XPath 路径为：//*[@id="kw"]。

3.4.6 Firefox 的 CSS 插件

同样，对于元素的 CSS 路径 Firefox 也有相应的工具可以帮助我们生成。在火狐中想要使用该功能，需要同时安装 Firebug 和 FirePath 两个插件，具体步骤如下。

- (1) 安装好 Firebug 和 FirePath 插件。
- (2) 打开一个网页，如 <http://www.baidu.com>。
- (3) 右击搜索框并单击 Inspect in FirePath。
- (4) 在打开的 FirePath 工具中选择 CSS，如图 3-8 所示。
- (5) 单击 按钮后用鼠标单击目标元素。

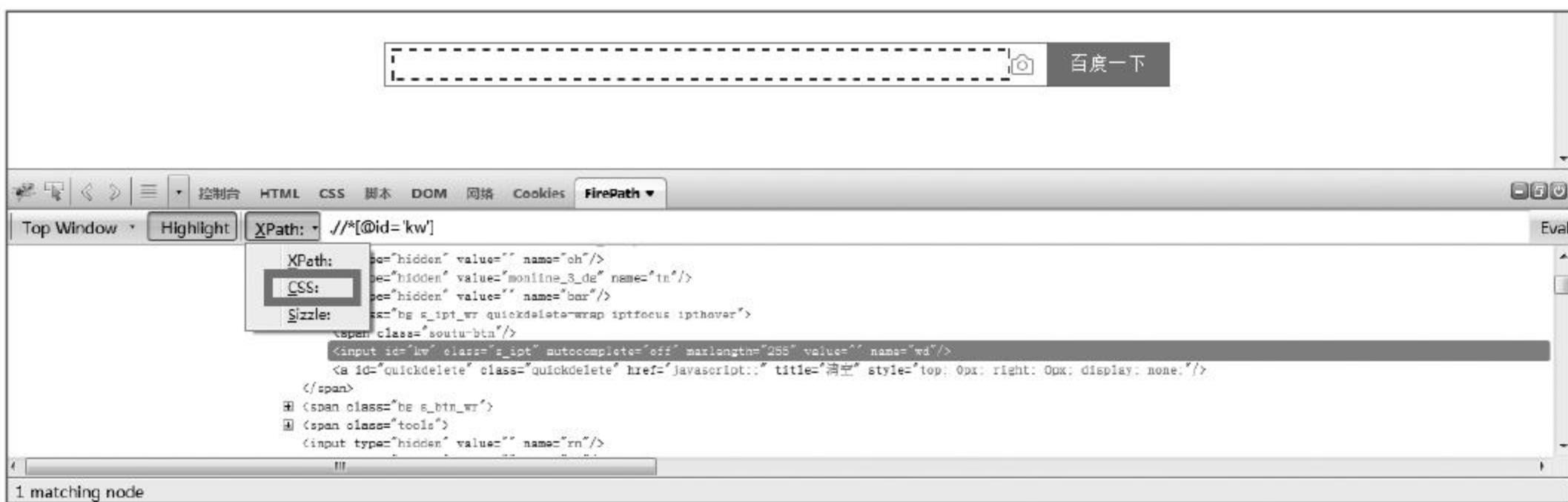


图 3-8 Firefox 生成 CSS 选项

(6) 查看 CSS 后面的输入框内容即为所单击元素的 CSS 定位路径, 如图 3-9 所示为百度搜索框的 CSS 定位路径。



图 3-9 Firefox 生成 CSS 路径

3.4.7 Chrome 的 CSS 工具

Chrome 作为主流的浏览器之一, 也是支持 CSS 定位生成功能的, 同样地也是集成到了它的开发者工具中, 具体的使用步骤如下。

- (1) 打开开发者工具。
- (2) 定位到具体的元素节点。
- (3) 在开发者工具中右击该 HTML 节点。
- (4) 选择 Copy → Copy Selector 即可复制该节点的 CSS 路径。

3.4.8 Firefox 的 WebDriver Element Locator 插件

Firefox 的插件库是相当丰富的, 对于定位它还有一个 WebDriver 友好的插件叫作 WebDriver Element Locator。它不仅提供了定位的路径, 还提供了生成 WebDriver 的代码, 而且还支持 WebDriver 的多种语言。具体的操作步骤如下。

- (1) 打开 Firefox 浏览器，进入 [https://addons.mozilla.org/en-US/firefox/。](https://addons.mozilla.org/en-US/firefox/)
- (2) 在搜索框里输入“WebDriver Element Locator”。
- (3) 找到插件并单击 Download Now 按钮，如图 3-10 所示。



图 3-10 WebDriver 插件下载

- (4) 会有一个弹出框，单击 Install Now。
- (5) 在 Firefox 里打开 <http://www.baidu.com>。
- (6) 右击百度搜索框后查看菜单内容，如图 3-11 所示。



图 3-11 WebDriver 插件演示

- (7) 选择对应的生成代码子项即可生成该语言的 WebDriver 脚本。
该工具支持生成脚本的语言有 C#、Java、Python 和 Ruby，根据所用的语言不同可以选择对应的子项，在这里可以直接选择 Python Locators 子项。

注意 可以看到 WebDriver Element Locator 目前仅支持生成 XPath 路径的 Selenium 定位脚本，不支持 CSS 路径的 Selenium 定位脚本生成。

3.5 Selenium 中进行元素定位

前面学习了 HTML 中元素定位的方法和工具，本节学习如何在 Selenium 中进行元素的定位。

3.5.1 获取一个定位元素

在 Selenium 中的元素定位是通过几个定位接口对开发人员开放的，主要的几个定位方法的名称如下。

- `find_element_by_class_name`。
- `find_element_by_css_selector`。
- `find_element_by_id`。
- `find_element_by_link_text`。
- `find_element_by_name`。
- `find_element_by_partial_link_text`。
- `find_element_by_tag_name`。
- `find_element_by_xpath`。

上述定位元素的接口分别是通过元素的 `ClassName`、`CSS` 路径、`ID`、链接文字、`Name`、部分链接文字、标签名以及 `XPath` 路径来进行定位的。我们要做的是在调用具体方法的时候传递过去正确的参数即可。接下来通过一个示例节点来说明下如何使用每个定位方法。具体节点代码如下。

```
<input type="text" class="s_ipt nobg_s_fm_hover" name="wd" id="kw" >
```

上述节点可以通过以下几个方法分别进行定位，如下。

- `find_element_by_css_selector("#kw")`。
- `find_element_by_id("kw")`。
- `find_element_by_xpath("//*[@id='kw']")`。
- `find_element_by_name("wd")`。
- `find_element_by_tag_name("input")`。

其中，以 `id` 作为定位参数的通常都可以精确定位到该元素，而以 `Name`、`TagName` 作为定位参数的则可能定位到多个符合条件的节点，而 Selenium 则会返回第一个匹配到的节点元素。另外，对应 `link text` 之类的方法只适用于 `a` 元素，所以上面的 `input` 元素不可使用这类方法进行定位，接下来就看看如何通过链接文字来定位链接，示例节点代码如下。

```
<a href="https://www.python.org" target="_blank">python 官网 </a>
```

对于上面的元素节点，就可以使用 link text 的方法进行定位，具体如下。

- `find_element_by_link_text("python 官网")`。
- `find_element_by_partial_link_text("python")`。

上面两个方法都会匹配到目标元素，第一个方法会匹配到链接文字为“python 官网”的第一个链接，第二个方法会匹配到链接文字包含“python”的第一个链接，所以如果上述代码中的节点在整个 HTML 页面中是唯一的或者最先出现的，那么将会被匹配成功。

注意 如上所述，`find_element_by_XXX` 方法返回的永远是第一个匹配到的元素，而实际情况下并非所有元素都有 `id` 属性，也并非所有想得到的元素都是最先出现的；那么如何获取匹配节点中的非第一个元素？

3.5.2 获取一组定位元素

为了解决上面提到的问题，Selenium 提供了另外一套定位方法，具体名称如下。

- `find_elements_by_class_name`。
- `find_elements_by_css_selector`。
- `find_elements_by_id`。
- `find_elements_by_link_text`。
- `find_elements_by_name`。
- `find_elements_by_partial_link_text`。
- `find_elements_by_tag_name`。
- `find_elements_by_xpath`。

可以看到这一套方法和前面提到的方法是相对应的，由原来的 `find_element_by_XXX` 变成 `find_elements_by_XXX`，即返回一组所有匹配到的元素，这样开发者就可以通过这个方法来处理匹配到多个元素时的情况。例如，HTML 的部分代码如下所示。

```
<ul id="mylist">
    <li class="red">java</li>
    <li class="red">python</li>
    <li class="red">c#</li>
    <li class="red">ruby</li>
</ul>
```

如果想要获取 Python 所在的 li 元素，则需要通过如下代码来获取此元素。

```
webdriver.find_elements_by_class_name("red")[1]
```

即先通过 `find_elements_by_class_name` 获取到所有匹配的元素，然后再通过索引下标获取第 2 个元素（默认下标从 0 开始）。

3.5.3 匹配非第一个元素

除了通过上面的方法来处理同时匹配多个元素的情况，还可以使用 CSS 和 XPath 的语法功能来支持选择多个匹配元素中的指定元素；同样以 3.5.2 节的 HTML 为例看看 CSS 和 XPath 是如何支持多元素选择的，首先来看 CSS 的代码脚本如下。

```
webdriver.find_element_by_css_selector('#mylist li:nth-child(1)')
```

代码中关键定位符为 `:nth-child`，即定位元素中的第几个孩子节点。上面的代码的意思是定位 id 为 mylist 元素下的第 1 个 li 孩子节点。再来看看 XPath 的代码脚本如下。

```
webdriver.find_element_by_xpath('//*[@id="mylist"]/li[1]')
```

代码中通过中括弧中的数字来描述具体定位第几个孩子节点，上面的代码同样是定位 id 为 mylist 的元素下第 1 个 li 孩子节点。

注意 通常情况下需要定位的元素通过上面提到的方法都可以直接定位到；而另一些时候需要定位的元素却没有任何属性，并且它们都分布在在整个 HTML 页面内，无法通过以上方式在整个 HTML 页面内定位第 N 个子元素；这时通常使用的方法就是寻找它的可定位的父类元素，然后在这个父类元素的范围内进行子元素定位。



第4章

Selenium IDE

CHAPTER
04

Selenium IDE 是 Firefox 的一个插件，它可以支持对页面上的测试步骤进行录制与回放。即可以通过这个工具进行页面自动化脚本的录制，而无须手动开发脚本。

默认情况下，Selenium IDE 录制生成的是 HTML 表格形式的测试脚本；回放时默认也是以 HTML UNIT 的方式回放脚本。

此外，Selenium IDE 还有一个特点就是，它可以把 HTML 形式的测试脚本转换成其他支持 Selenium 的语言脚本。例如，转换为 Python 语言的脚本之后，再通过执行 Python 脚本来达到回放的效果。Selenium IDE 支持转换的脚本语言有：C#、Java、Python、Ruby。

4.1 Selenium IDE 安装

Selenium IDE 的安装过程包括 Firefox 浏览器安装、Selenium IDE 火狐插件的安装。如果本机已经安装了 Firefox 浏览器，则可以直接跳至 4.1.2 节安装 Selenium IDE 插件。

4.1.1 Firefox 安装

由于 Selenium IDE 是 Firefox 的插件，所以在安装 Selenium IDE 之前需要先安装一下 Firefox 浏览器。具体的安装过程如下。

(1) 进入 Firefox 官网下载页面 (<http://www.firefox.com.cn/download/>)。

(2) 单击下载对应的 Firefox 版本。

- (3) 双击下载的 exe 文件。
- (4) 默认安装或选择安装目录。
- (5) 依次确认完成安装。

4.1.2 Selenium IDE 在线安装

在 Firefox 安装完成之后，就可以进行 Selenium IDE 的安装了。具体安装步骤如下。

- (1) 使用 Firefox 打开 Selenium IDE 的插件下载页面 (<https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/>)。
- (2) 单击 + Add to Firefox 按钮，如图 4-1 所示。

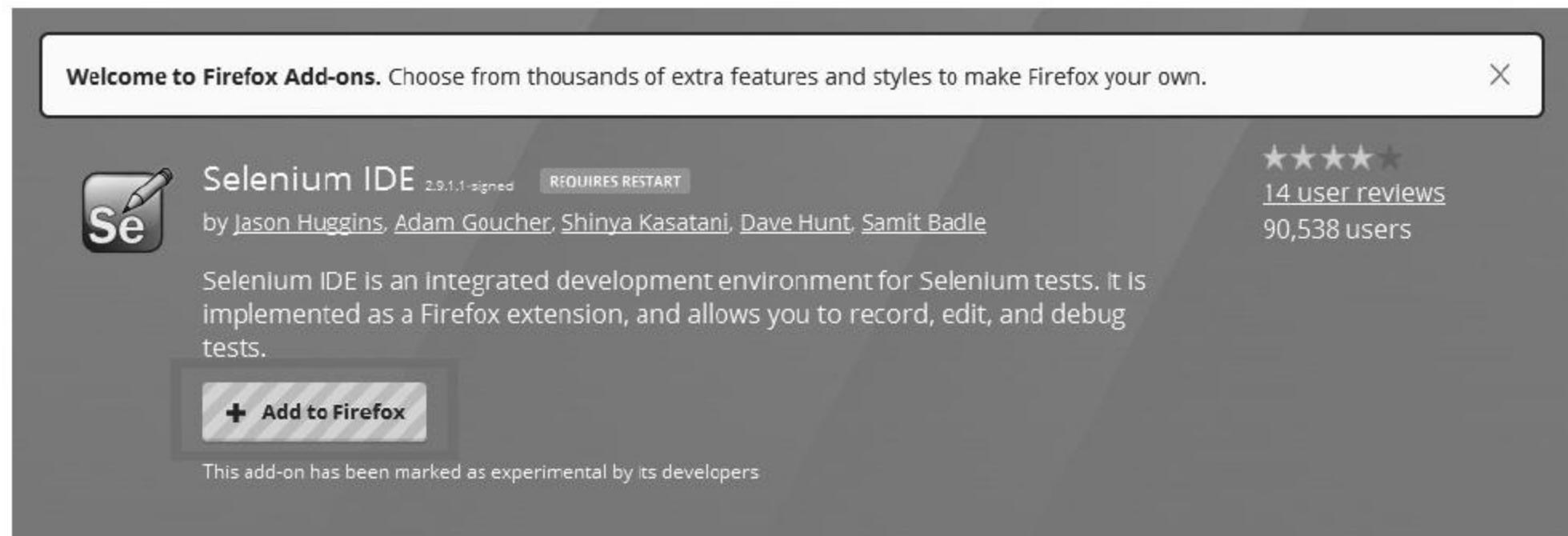


图 4-1 Selenium IDE 插件页面

- (3) 单击 Install 按钮进行安装，如图 4-2 所示。

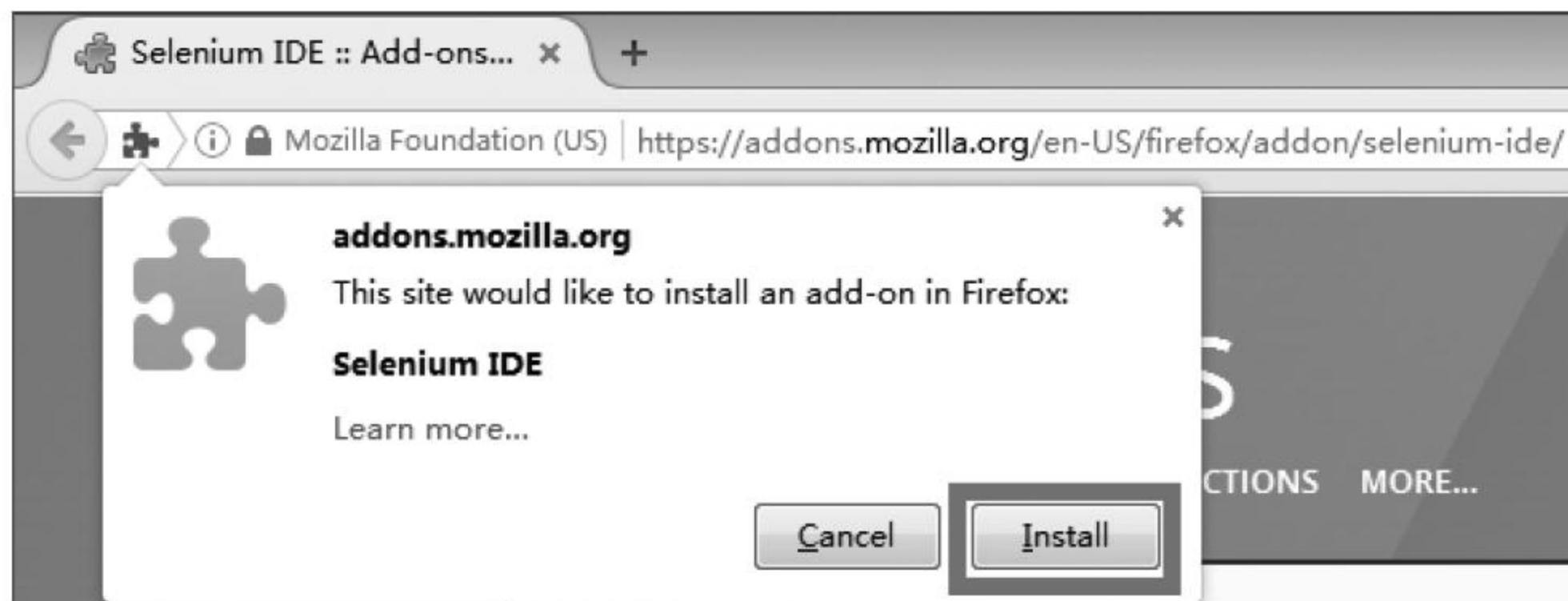


图 4-2 Selenium IDE 插件安装

- (4) 安装完成后重启 Firefox。
- (5) 按一下 Alt 键 → 单击 Tools 菜单栏 → 单击 Selenium IDE 打开 IDE，如图 4-3 所示。
- (6) 成功弹出 Selenium IDE 窗口则安装成功，如图 4-4 所示。



图 4-3 打开 Selenium IDE

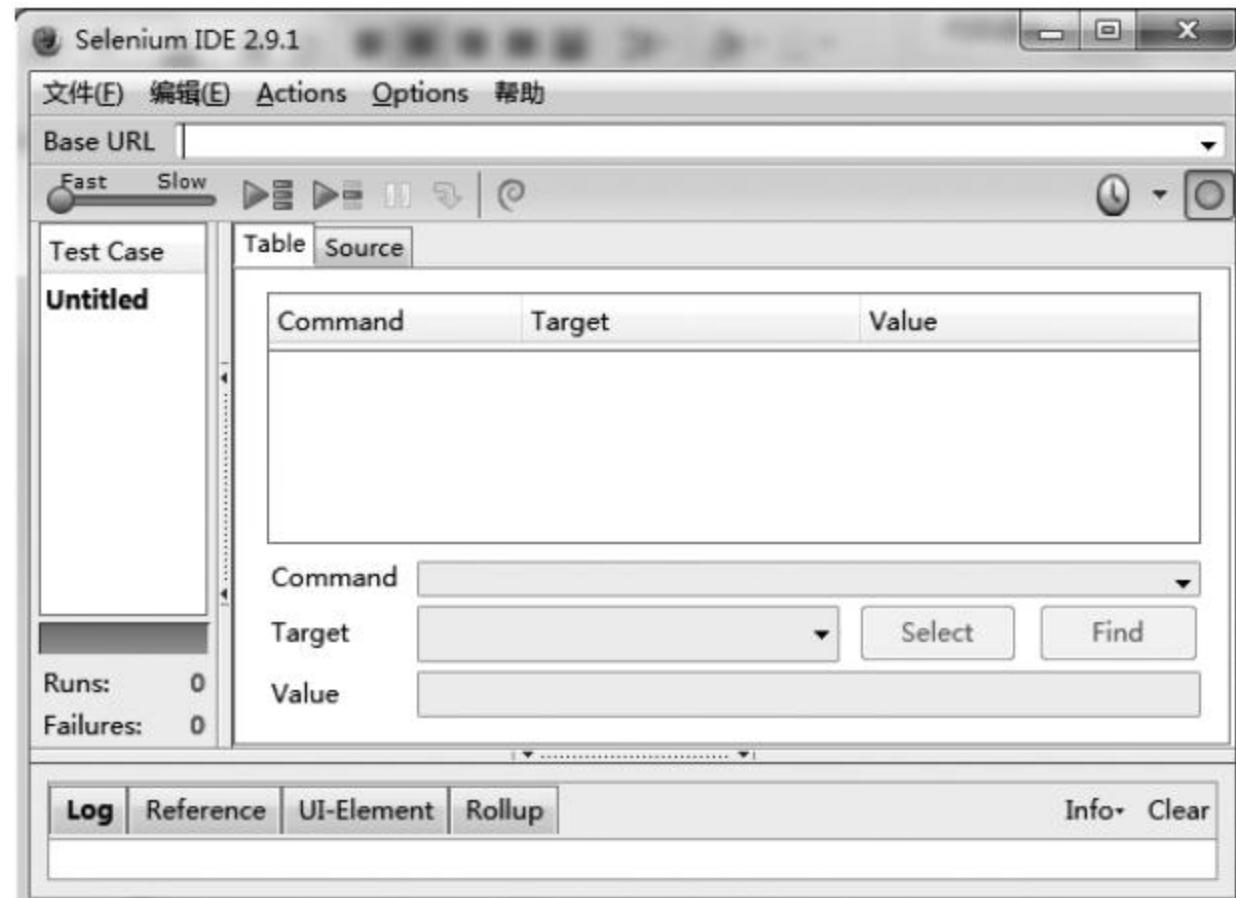


图 4-4 Selenium IDE 界面

4.1.3 Selenium IDE 本地安装

对于无法访问 Firefox 插件官网的读者，可以到 <http://www.testqa.cn/download> 下载最新火狐插件，并进行本地安装。具体本地的安装步骤如下。

- (1) 打开 Firefox 浏览器。
- (2) 按一下 Alt 键→单击 Tools 菜单栏→选择“附加组件”选项，如图 4-5 所示。



图 4-5 Firefox 附加组件

(3) 单击“配置”按钮，选择“从文件安装附加组件…”，如图 4-6 所示。



图 4-6 Firefox 本地安装组件

(4) 浏览下载到本地的 Selenium IDE 安装包并选择，如图 4-7 所示。



图 4-7 选择 Selenium IDE 插件包

(5) 在 Firefox 提示框中单击“安装”，如图 4-8 所示。

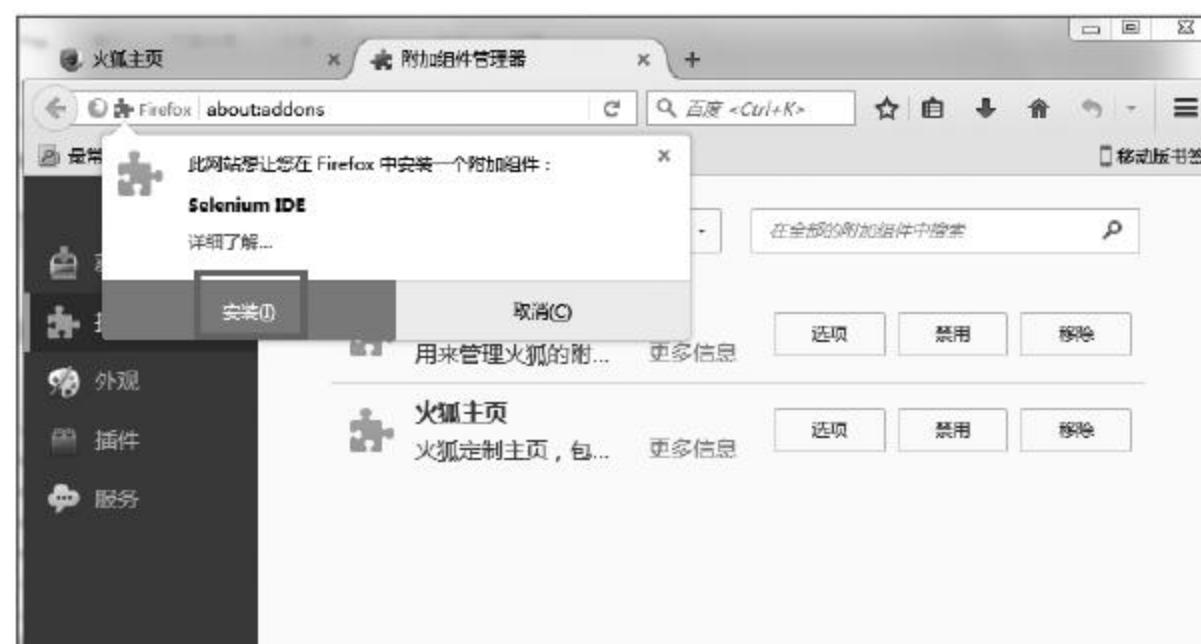


图 4-8 Selenium IDE 本地安装

(6) 重启 Firefox 浏览器使用插件安装生效，如图 4-9 所示。



图 4-9 Selenium IDE 安装确认

(7) 按下 Alt 键→单击 Tools 菜单栏→单击 Selenium IDE 打开 IDE，如图 4-10 所示。



图 4-10 打开 Selenium IDE

(8) 正常弹出 Selenium IDE 窗口则安装成功，如图 4-11 所示。

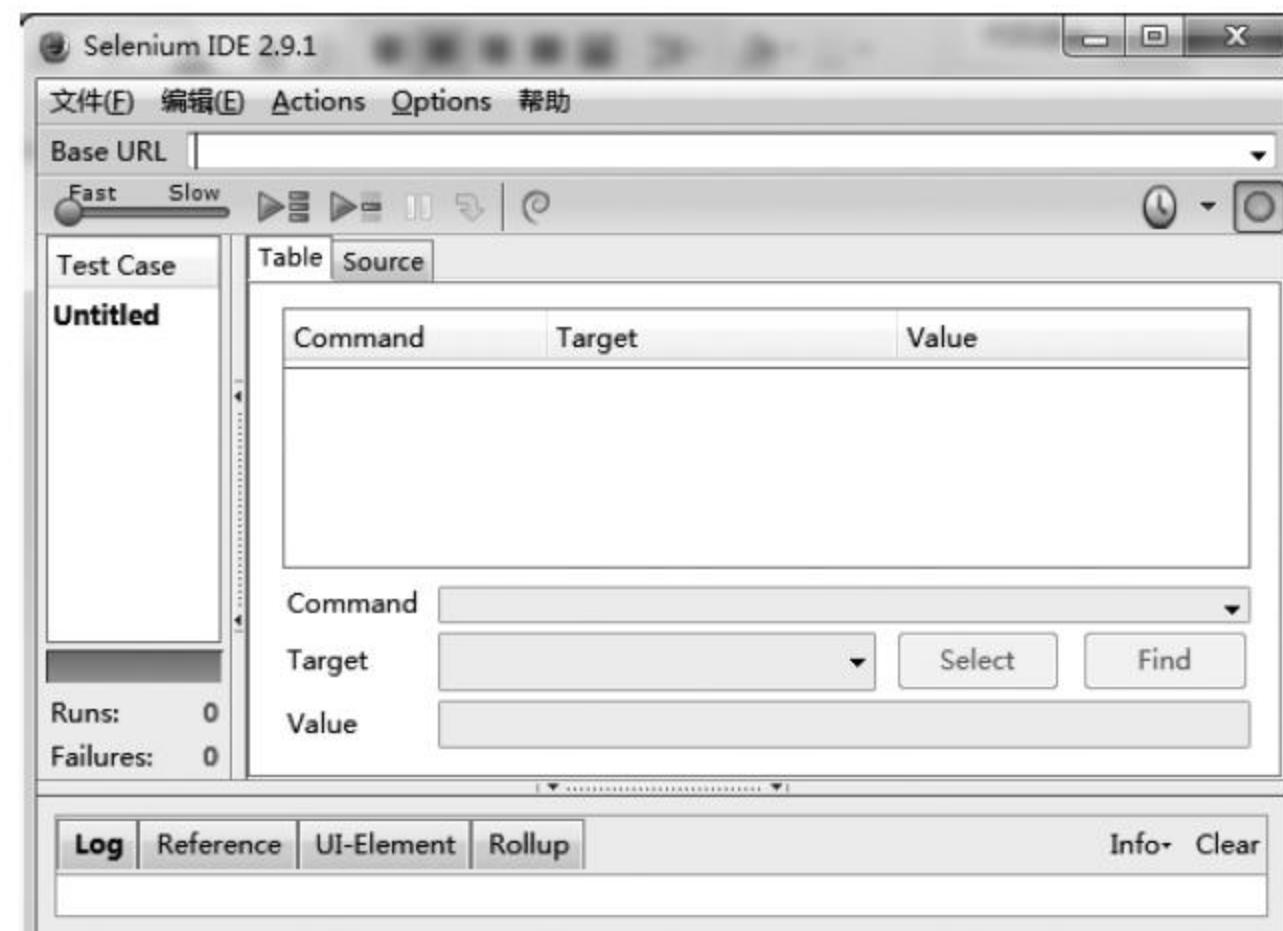


图 4-11 Selenium IDE 界面

4.2 Selenium IDE 功能介绍

Selenium IDE 安装完成之后，就可以使用 Selenium IDE 来帮助我们录制测试场景，然后转换为目标语言脚本，这里为 Python 测试脚本。通过这种方式开发测试脚本有以下三个好处。

- 初学者在代码基础不足的情况下仍可以开发自动化测试脚本。
- 可以帮助初学者了解并学习 Python 的 Selenium 脚本样例。
- 可以提到测试脚本的开发效率。

那么，接下来将会开始对 Selenium IDE 的基本功能进行一一介绍。

4.2.1 Selenium IDE 窗口

在具体进行脚本录制之前，看一下 Selenium IDE 的基本功能和使用方法。按照 4.1 节的步骤打开 Selenium IDE 窗口，就可以看到如图 4-12 所示的界面。

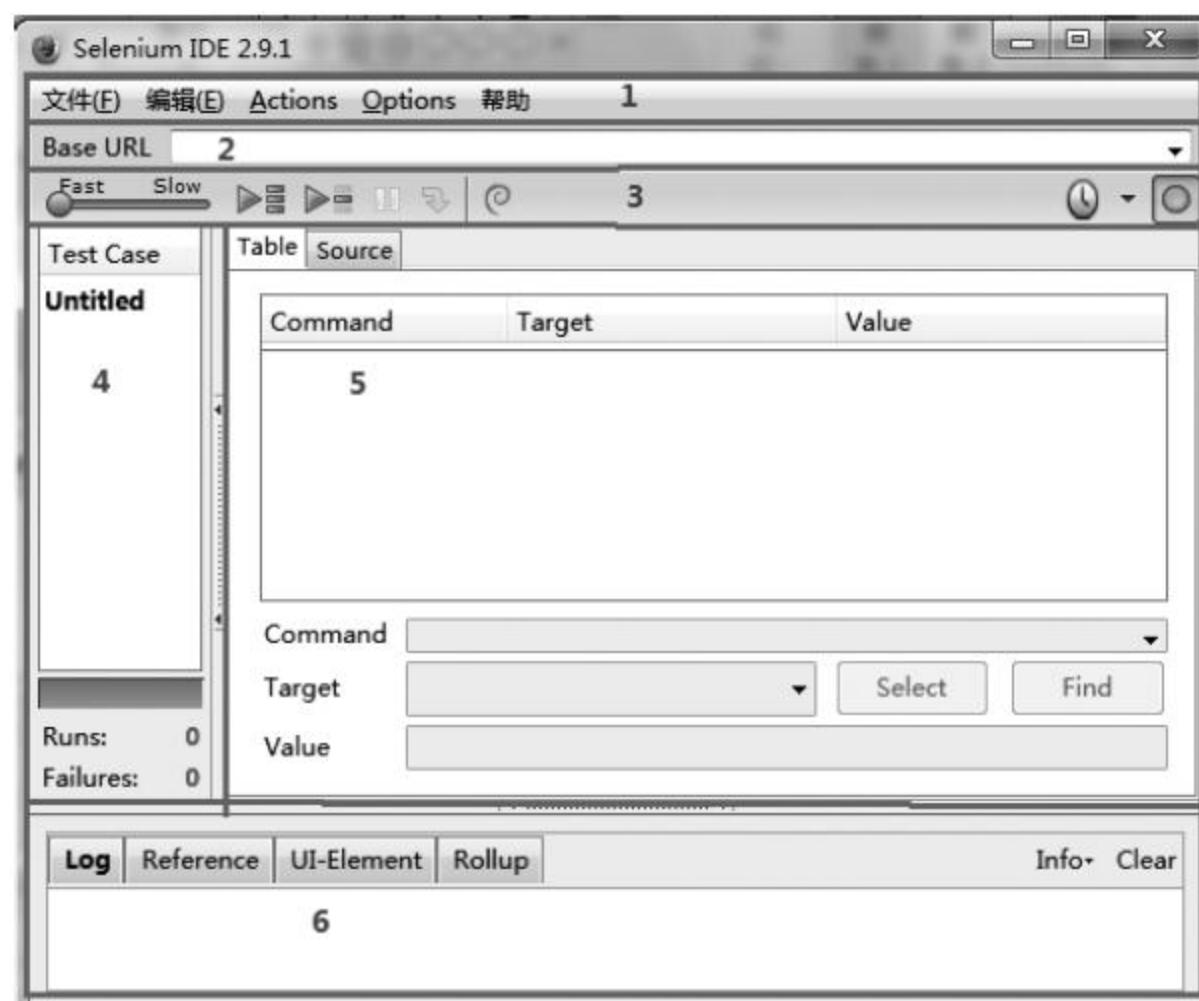


图 4-12 Selenium IDE 主面板区域

从图 4-12 中可以看出，Selenium IDE 窗口主要由以下几个区域组成。

- 菜单栏。
- 地址栏。
- 工具栏。
- 用例管理区。
- 用例脚本开发区。
- 信息输出区。