How To Get The HTTP Status Code In Selenium WebDriver

By Roger Keays, 16 November 2012

One of the first things you'll notice when you start testing your web application with **Selenium WebDriver** is that there is no API to get the **HTTP status code** for a page. If you want to know why, you can go and read WebDriver issue #141. Personally **I don't care why** - I just want to test my HTTP reponse codes (especially 403 Permission Denied).

There is a fairly simple workaround you can use for WebDriver, but firstly lets look at how to do it **without WebDriver**. Nobody said you HAVE to use Selenium right?

Apache HTTPClient

This is a Java library use for general HTTP network communication and includes supports sessions via cookies. Their fluent API makes it relatively simple to get the response code for a URL.

```
public int getResponseCode(String url) {
    try {
        return Request.Get(url).execute().returnResponse().getStatusLine()
                .getStatusCode();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

You can easily add the HTTPClient fluent api to your Java project via maven:

```
<dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>fluent-hc</artifactId>
   <version>4.2.1</version>
   <scope>test</scope>
 </dependency>
```

The disadvantage of HTTPClient is it is doesn't include a browser like Selenium, although you can create sessions which might be useful for testing more complex use cases.

HTMLUnit

HTMLUnit is a headless Java browser and is actually one of the browsers supported by WebDriver. It's very easy to get the response code with HTMLUnit:

```
public int getResponseCode(String url) {
    try {
        WebClient client = new WebClient();
        client.setThrowExceptionOnFailingStatusCode(false);
        return client.getPage(url).getWebResponse().getStatusCode();
    } catch (IOException ioe) {
        throw new RuntimeException(ioe);
    }
}
```

This library is also available from Maven.

```
<dependency>
   <groupId>net.sourceforge.htmlunit</groupId>
   <artifactId>htmlunit</artifactId>
   <version>2.10</version>
   <scope>test</scope>
 </dependency>
```

Selenium WebDriver

Finally we get to WebDriver. If you've written a few tests with WebDriver already you're probably used to this API, only there is no native support for inspecting response headers. The simple workaround is to **write the response code to your response output**.

For example, on my 403 Permission Denied page, I have:

```
<h1 id="web_403">403 Access Denied</h1>
```

which can be easily checked via the WebDriver API:

```
public boolean is403(WebDriver driver) {
    try {
        driver.findElement(By.id("web_403"));
        return true;
    } catch (NoSuchElementException e) {
```

```
            return false;
        }
    }
```

You can generalise the idea for most HTTP responses that don't cause a redirect by adding a meta field to your <head> section. In JSF / Servlets 3.0 you do it like this:

```
<metaid="web_response" name="response"
      content="${facesContext.externalContext.response.status}"/>
```

This can be tested easily with WebDriver:

```
/**
 * Selenium doesn't give you access to the response headers, so we parse
 * the content. This relies on the web_response meta tag in the head
 * section of the output.
 */
public int getResponseCode(WebDriver driver) {
    return Integer.parseInt(driver.findElement(By.id("web_response"))
            .getAttribute("content"));
}
```

Note for Servlets before version 3.0 you will need to[implement your own response.getStatus()]method and I'm told that in PHP you use$_SERVER['REDIRECT_STATUS']for versions < 5.4 andhttp_response_code()for newer versions. Let me know how it is done in your language in the comments below.

This solution works pretty well and is a lot more productive than bashing on the Selenium developers for the gaping hole in their API.

So get testing slacker!

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. Since he left Australia in 2009, he has been living as a digital nomad in over 40 different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, and finding good food.