WebDriver: Advanced Usage

Explicit and Implicit Waits

Waiting is having the automated task execution elapse a certain amount of time before continuing with the next step. You should choose to use Explicit Waits or Implicit Waits.

WARNING: Do not mix implicit and explicit waits. Doing so can cause unpredictable wait times. For example setting an implicit wait of 10 seconds and an explicit wait of 15 seconds, could cause a timeout to occur after 20 seconds.

Explicit Waits

An explicit wait is code you define to wait for a certain condition to occur before proceeding further in the code. The worst case of this is Thread.sleep(), which sets the condition to an exact time period to wait. There are some convenience methods provided that help you write code that will wait only as long as required. WebDriverWait in combination with ExpectedCondition is one way this can be accomplished.

java

```
WebDriver driver = new FirefoxDriver();
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = (new WebDriverWait(driver, 10))
  .until(ExpectedConditions.presenceOfElementLocated(By.id("myDynamicElement")));
```

csharp python ruby

This waits up to 10 seconds before throwing a TimeoutException or if it finds the element will return it in 0 - 10 seconds. WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully. A successful return value for the ExpectedCondition function type is a Boolean value of true, or a non-null object.

This example is also functionally equivalent to the first Implicit Waitsexample.

Expected Conditions

There are some common conditions that are frequently encountered when automating web browsers. Listed below are a few examples for the usage of such conditions. The Java, C#, and Python bindings include convienence methods so you don’t have to code an ExpectedCondition class yourself or create your own utility package for them.

- Element is Clickable - it is Displayed and Enabled.

java

```
WebDriverWait wait = new WebDriverWait(driver, 10);
WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("someid")));
```

csharp python

The ExpectedConditions package (Java) (Python) (.NET) contains a set of predefined conditions to use with WebDriverWait.

Implicit Waits

An implicit wait is to tell WebDriver to poll the DOM for a certain amount of time when trying to find an element or elements if they are not immediately available. The default setting is 0. Once set, the implicit wait is set for the life of the WebDriver object instance.

java

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = driver.findElement(By.id("myDynamicElement"));
```

csharp python ruby

RemoteWebDriver

Taking a Screenshot

java

```
import java.io.File;
import java.net.URL;

import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.Augmenter;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class Testing {

    public void myTest() throws Exception {
        WebDriver driver = new RemoteWebDriver(
                                new URL("http://localhost:4444/wd/hub"),
                                DesiredCapabilities.firefox());

        driver.get("http://www.google.com");

        // RemoteWebDriver does not implement the TakesScreenshot class
        // if the driver does have the Capabilities to take a screenshot
        // then Augmenter will add the TakesScreenshot methods to the instance
        WebDriver augmentedDriver = new Augmenter().augment(driver);
        File screenshot = ((TakesScreenshot)augmentedDriver).
                            getScreenshotAs(OutputType.FILE);
    }
}
```

csharp python ruby

Using a FirefoxProfile

java

```
FirefoxProfile fp = new FirefoxProfile();
// set something on the profile...
DesiredCapabilities dc = DesiredCapabilities.firefox();
dc.setCapability(FirefoxDriver.PROFILE, fp);
WebDriver driver = new RemoteWebDriver(dc);
```

csharp python

Using ChromeOptions

java

```
ChromeOptions options = new ChromeOptions();
// set some options
DesiredCapabilities dc = DesiredCapabilities.chrome();
dc.setCapability(ChromeOptions.CAPABILITY, options);
WebDriver driver = new RemoteWebDriver(dc);
```

csharp python

AdvancedUserInteractions

The Actions class(es) allow you to build a Chain of Actions and perform them. There are too many possible combinations to count. Below are a few of the common interactions that you may want to use. For a full list of actions please refer to the API docs Java C# Ruby Python

The Advanced User Interactions require native events to be enabled. Here's a table of the current support Matrix for native events:

platform	IE6	IE7	IE8	IE9	FF3.6	FF10+	Chrome stable	Chrome beta	Chrome dev	Opera	Android	iOS
Windows XP	Y	Y	Y	n/a	Y	Y	Y	Y	Y	?	Y [1]	n/a
Windows 7	n/a	n/a	Y	Y	Y	Y	Y	Y	Y	?	Y [1]	n/a
Linux (Ubuntu)	n/a	n/a	n/a	n/a	Y [2]	Y [2]	Y	Y	Y	?	Y [1]	n/a
Mac OSX	n/a	n/a	n/a	n/a	N	N	Y	Y	Y	?	Y [1]	N
Mobile Device	n/a	n/a	n/a	n/a	n/a	?	n/a	n/a	n/a	?	Y	N

[1] *(1, 2, 3, 4)* Using the emulator

[2] *(1, 2)* With explicitly enabling native events

Browser Startup Manipulation

Todo

Topics to be included:

- restoring cookies
- changing firefox profile
- running browsers with plugins

Using a Proxy

Internet Explorer

The easiest and recommended way is to manually set the proxy on the machine that will be running the test. If that is not possible or you want your test to run with a different configuration or proxy, then you can use the following technique that uses a Capababilities object. This temporarily changes the system's proxy settings and changes them back to the original state when done.

java

```
String PROXY = "localhost:8080";

org.openqa.selenium.Proxy proxy = new org.openqa.selenium.Proxy();
proxy.setHttpProxy(PROXY)
     .setFtpProxy(PROXY)
     .setSslProxy(PROXY);
DesiredCapabilities cap = new DesiredCapabilities();
cap.setCapability(CapabilityType.PROXY, proxy);

WebDriver driver = new InternetExplorerDriver(cap);
```

python ruby

Chrome

Is basically the same as internet explorer. It uses the same configuration on the machine as IE does (on windows). On Mac it uses the System Preference -> Network settings. On Linux it uses (on Ubuntu) System > Preferences > Network Proxy Preferences (Alternatively in "/etc/environment" set http_proxy). As of this writing it is unknown how to set the proxy programmatically.

Firefox up to version 47.0.1

Firefox maintains its proxy configuration in a profile. You can preset the proxy in a profile and use that Firefox Profile or you can set it on profile that is created on the fly as is shown in the following example - deprecated, no longer working with GeckoDriver.

java

```
String PROXY = "localhost:8080";

org.openqa.selenium.Proxy proxy = new org.openqa.selenium.Proxy();
proxy.setHttpProxy(PROXY)
     .setFtpProxy(PROXY)
     .setSslProxy(PROXY);
DesiredCapabilities cap = new DesiredCapabilities();
cap.setCapability(CapabilityType.PROXY, proxy);
WebDriver driver = new FirefoxDriver(cap);
```

python ruby

Firefox version 48 and newer - GeckoDriver

Firefox maintains its proxy configuration in a profile. You can preset the proxy in a profile and use that Firefox Profile or you can set it on profile that is created on the fly as is shown in the following example. With GeckoDriver the proxy has to be passed through the required capabilities.

java

```
String PROXY = "localhost";
int PORT = 8080;

com.google.gson.JsonObject json = new com.google.gson.JsonObject();
json.addProperty("proxyType", "MANUAL");
json.addProperty("httpProxy", PROXY);
json.addProperty("httpProxyPort", PORT);
json.addProperty("sslProxy", PROXY);
json.addProperty("sslProxyPort", PORT);

DesiredCapabilities cap = new DesiredCapabilities();
cap.setCapability("proxy", json);

GeckoDriverService service =new GeckoDriverService.Builder(firefoxBinary)
  .usingDriverExecutable(new File("path to geckodriver"))
  .usingAnyFreePort()
  .usingAnyFreePort()
  .build();
service.start();

// GeckoDriver currently needs the Proxy set in RequiredCapabilities
driver = new FirefoxDriver(service, cap, cap);
```

Opera

Todo

HTML5

Todo

Parallelizing Your Test Runs

Todo