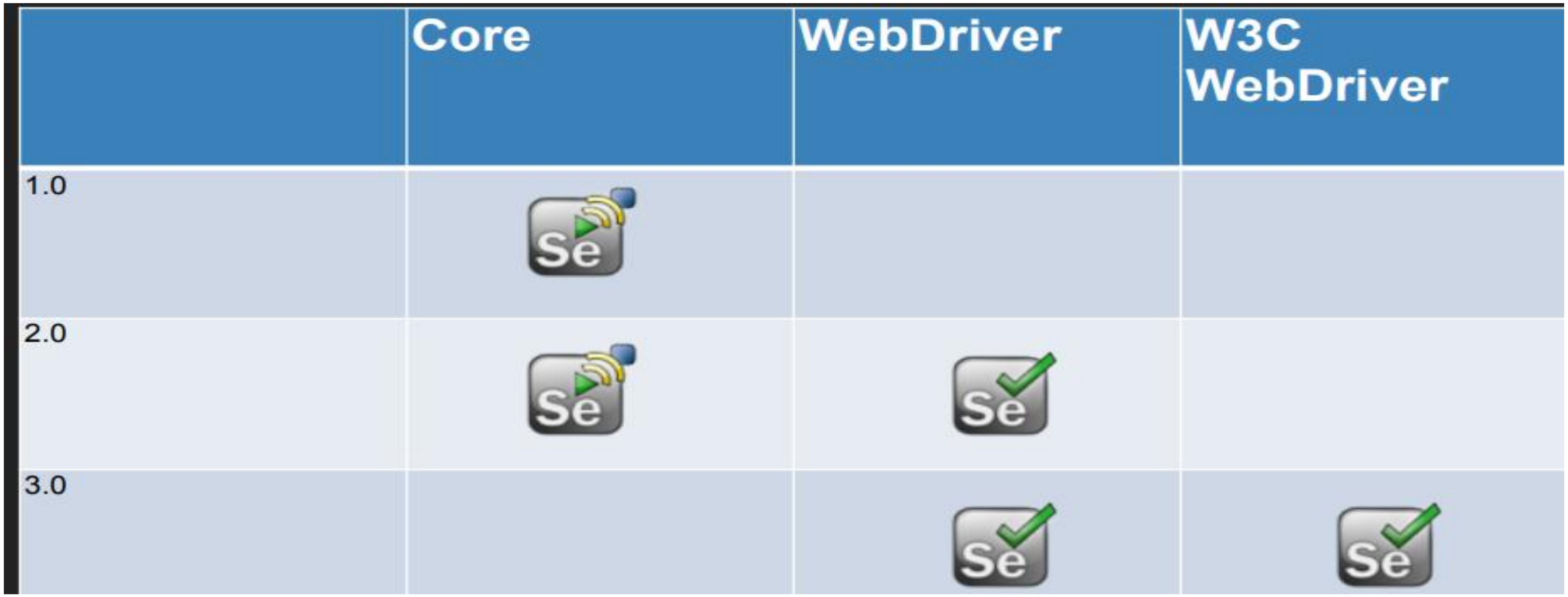
infogain

Selenium WebDriver

Introducing WebDriver and WebElements

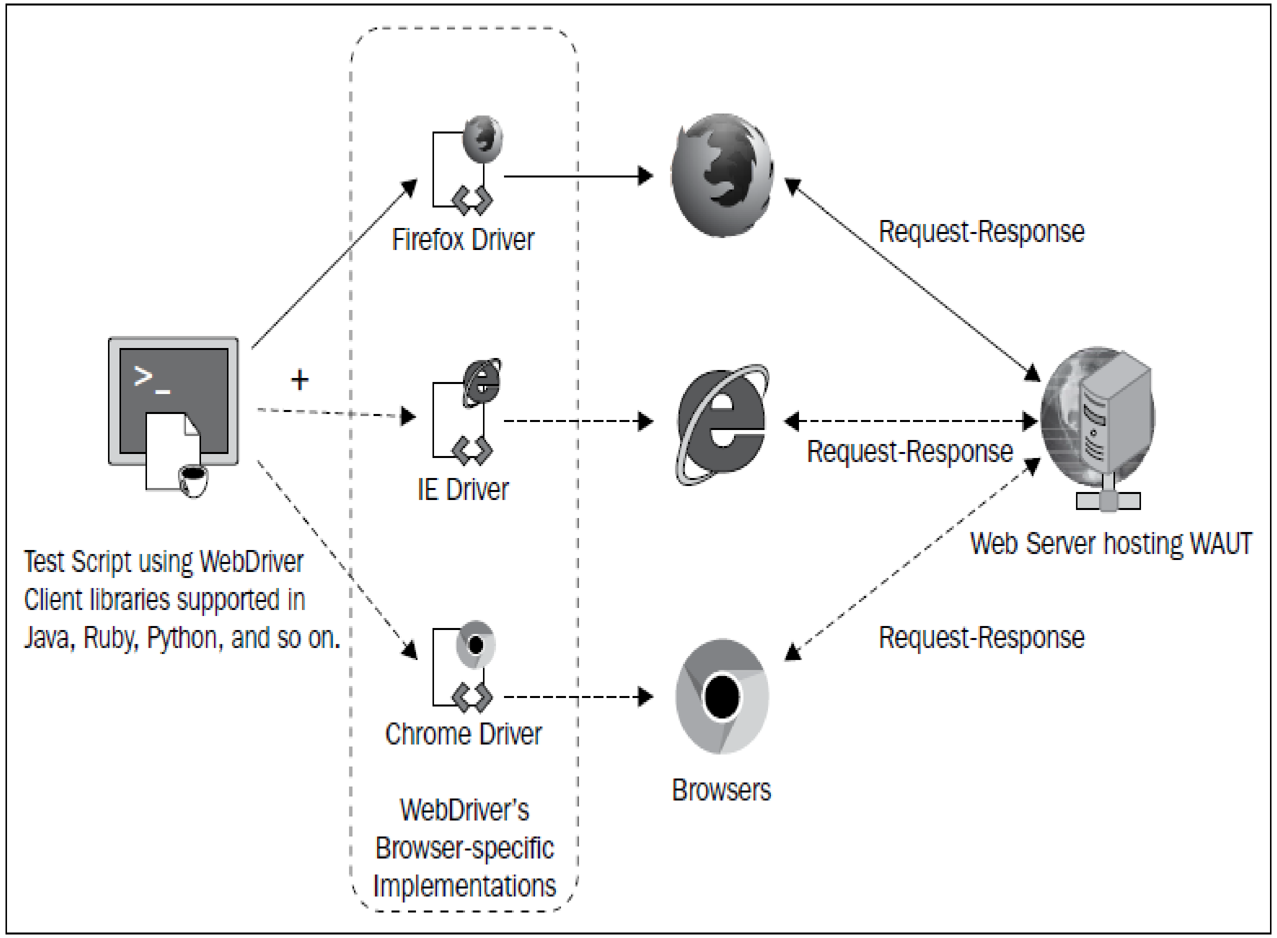
- What is Selenium ?
- Difference between Selenium 1 and Selenium 2
- Setting up an Eclipse project to execute the example code
- Locating WebElements on a web page
- Actions that can be taken on the WebElements

What is Selenium 3.0



	Core	WebDriver	W3C WebDriver
1.0	✓		
2.0	✓	✓	
3.0		✓	✓

What is Selenium 3.0

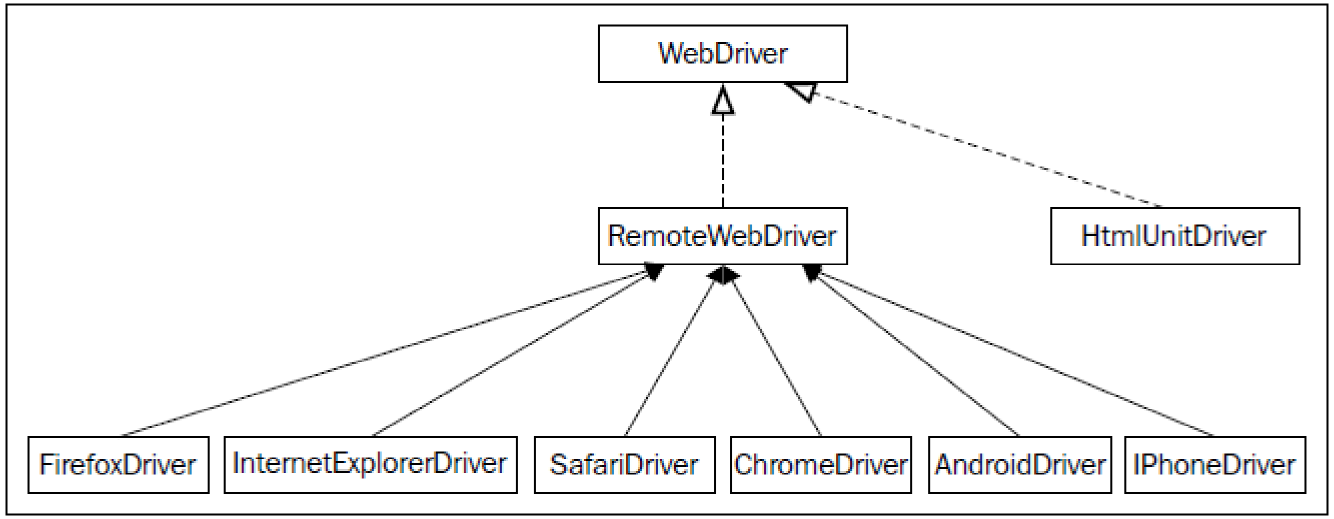

Firefox Driver
IE Driver
Chrome Driver
+
Test Script using WebDriver Client libraries supported in Java, Ruby, Python, and so on.
WebDriver's Browser-specific Implementations
Browsers
Request-Response
Request-Response
Request-Response
Web Server hosting WAUT

- ❑ Having better APIs
- ❑ Testing mobile apps
- ❑ Having developer support and advanced functionalities

WebDriver Interface

infogain
www.infogain.com

Selenium Webdriver makes direct calls to the browser using each browser’s native support for automation.



```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class NavigateToAUrl {
public static void main(String[] args){
WebDriver driver = new FirefoxDriver();
driver.get("http://www.google.com");
} }
```

WebElements and Locating WebElements

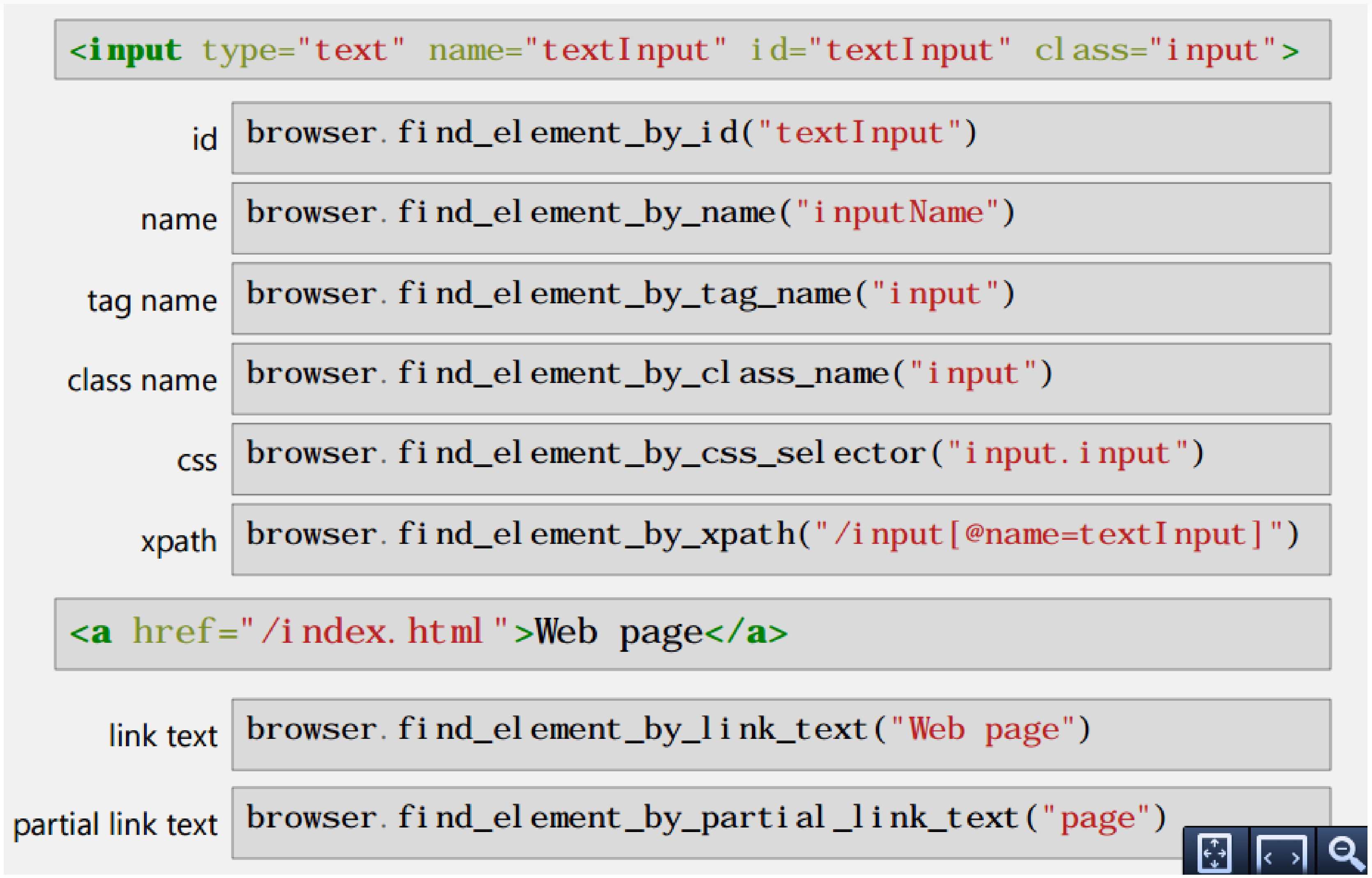
www.infogain.com

- A web page is comprised of many different HTML elements, such as buttons, links, a body, labels, forms, and so on, that are named WebElements in the context of WebDriver.

Webdriver location strategies:

- Id
- Name
- Tag name
- Link text
- Partial link text
- Class name
- Css selector
- Xpath

WebElements and Locating WebElements



```
<input type="text" name="textInput" id="textInput" class="input">
```

id	`browser.find_element_by_id("textInput")`
name	`browser.find_element_by_name("inputName")`
tag name	`browser.find_element_by_tag_name("input")`
class name	`browser.find_element_by_class_name("input")`
css	`browser.find_element_by_css_selector("input.input")`
xpath	`browser.find_element_by_xpath("/input[@name=textInput]")`

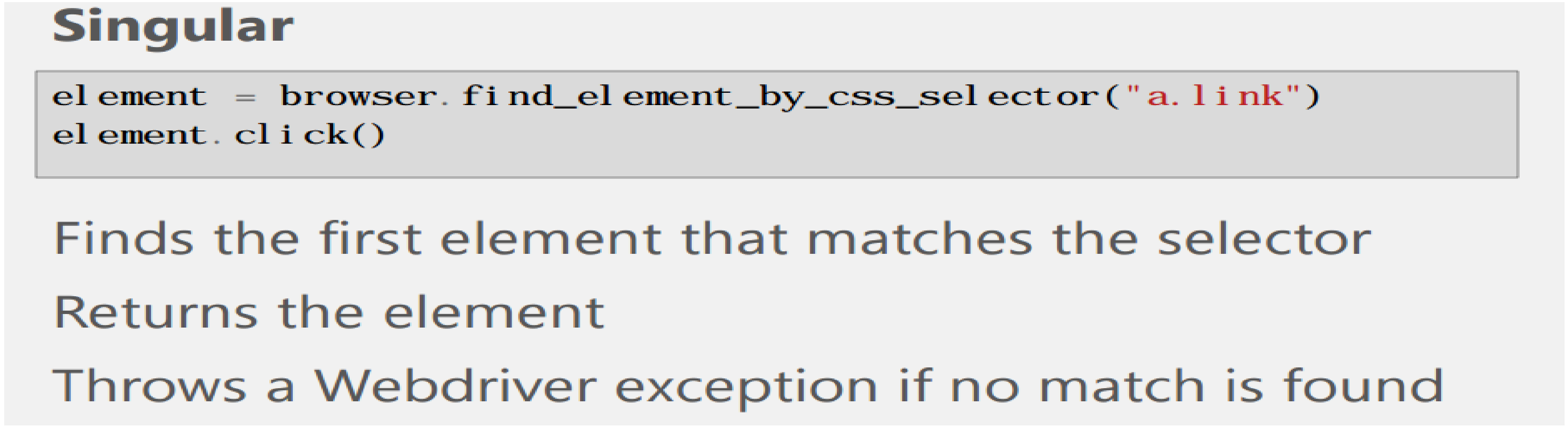
```
<a href="/index.html">Web page</a>
```

link text	`browser.find_element_by_link_text("Web page")`
partial link text	`browser.find_element_by_partial_link_text("page")`

WebElements and Locating WebElements

www.infogain.com

- Singular and Plural



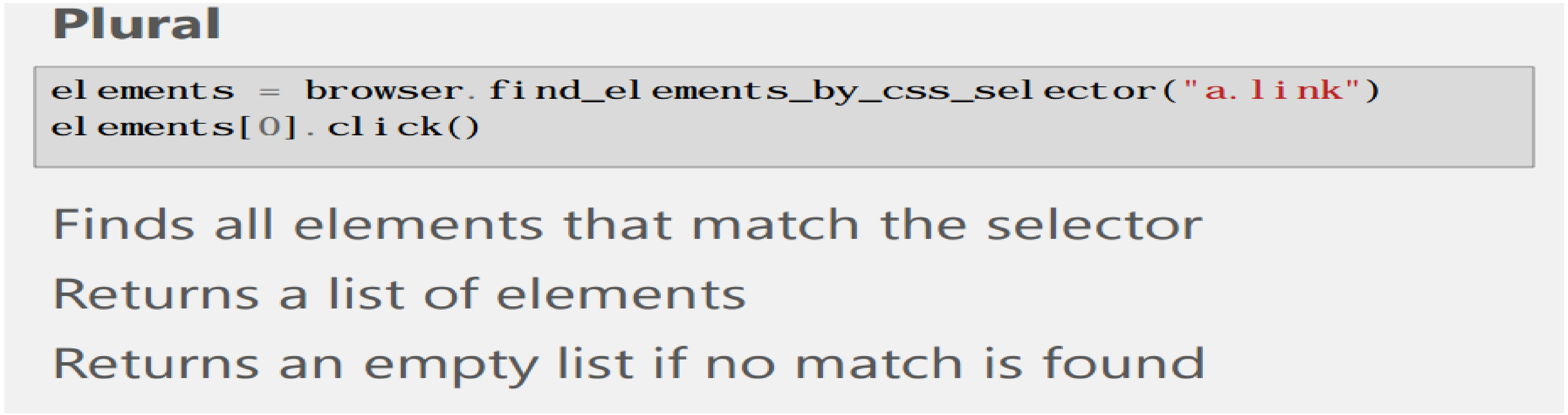
Singular

```
element = browser.find_element_by_css_selector("a.link")
element.click()
```

Finds the first element that matches the selector

Returns the element

Throws a Webdriver exception if no match is found



Plural

```
elements = browser.find_elements_by_css_selector("a.link")
elements[0].click()
```

Finds all elements that match the selector

Returns a list of elements

Returns an empty list if no match is found

WebElements and Locating WebElements

www.infogain.com

- **Choosing a location strategy**
  - Use id or name whenever possible
  - CSS attribute selectors and pseudo classes can be extremely powerful
  - Use xpath as a last resort

WebElements : Actions

- java.lang.String getAttribute(java.lang.String name)
- void sendKeys(java.lang.CharSequence...keysToSend)
- searchBox.sendKeys(Keys.chord(Keys.SHIFT,“Selenium"));
- searchBox.clear();
- searchBox.submit();
- searchButton.getLocation()
- searchButton.getText()
- searchButton.getTagName()
- searchButton.isDisplayed()
- searchButton.isEnabled()
- boolean isSelected()

Advanced Interactions of WebDriver

infogain
www.infogain.com

Understanding actions, build, and perform

- Suppose there is a scenarios where we have to perform multiple actions at the same time .

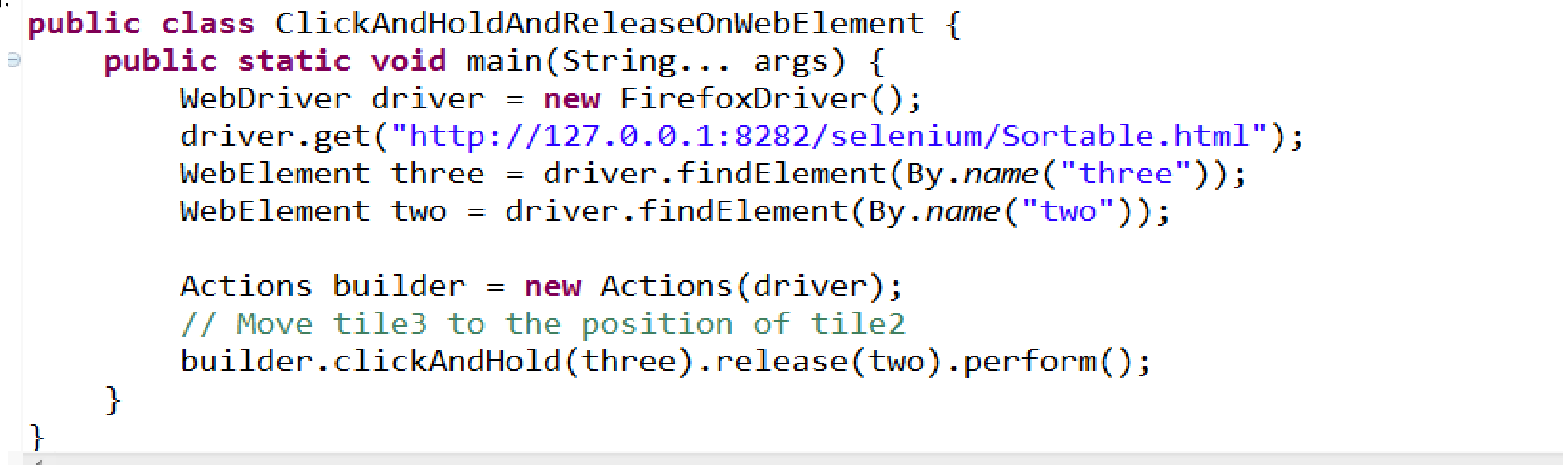
```
Actions builder = new Actions(driver);
builder.keyDown(Keys.CONTROL)
.click(one)
.click(three)
.click(five)
.keyUp(Keys.CONTROL);
// Generate the composite action.
Action compositeAction = builder.build();
// Perform the composite action.
compositeAction.perform(); }
Case 2: // Perform the action.
builder.perform();
Case 3:
```

Advanced Interactions of WebDriver: Mouse Event

infogain
www.infogain.com

- The API syntax for the click() method is as follows:

public Actions click()



```java
public class ClickAndHoldAndReleaseOnWebElement {
    public static void main(String... args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://127.0.0.1:8282/selenium/Sortable.html");
        WebElement three = driver.findElement(By.name("three"));
        WebElement two = driver.findElement(By.name("two"));

        Actions builder = new Actions(driver);
        // Move tile3 to the position of tile2
        builder.clickAndHold(three).release(two).perform();
    }
}
```

The moveToElement action

- ❑ The moveToElement() method is another method of WebDriver that helps us to move the mouse cursor to a WebElement on the web page.
- ❑ The API syntax for the moveToElement() method is as follows:

public Actions moveToElement(WebElement toElement)

The contextClick at current location action

- The contextClick() method, also known as right-click, is quite common on many web pages these days.
- The context is nothing but a menu; a list of items is associated to a WebElement based on the current state of the web page .

public Actions contextClick(WebElement onElement)

```
builder.contextClick(contextMenu)
    .click(driver.findElement(By.name("Item 4")))
         .perform();
```

❑ The contextClick at current location action

public Actions contextClick()

```
builder.moveToElement(contextMenu)
        .contextClick()
         .click(driver.findElement(By.name("Item 4")))
        .perform();
}
```

The sendKeys() method
This is used to type in alphanumeric and special character keys into WebElements such as textbox, textarea, and so on .

Exploring the Features of WebDriver

- Setting the desired capabilities for a browser
- Taking screenshots
- Locating target windows and iFrames
- Exploring Navigate
- Waiting for WebElements to load
- Handling cookies

Setting the desired capabilities for a browser

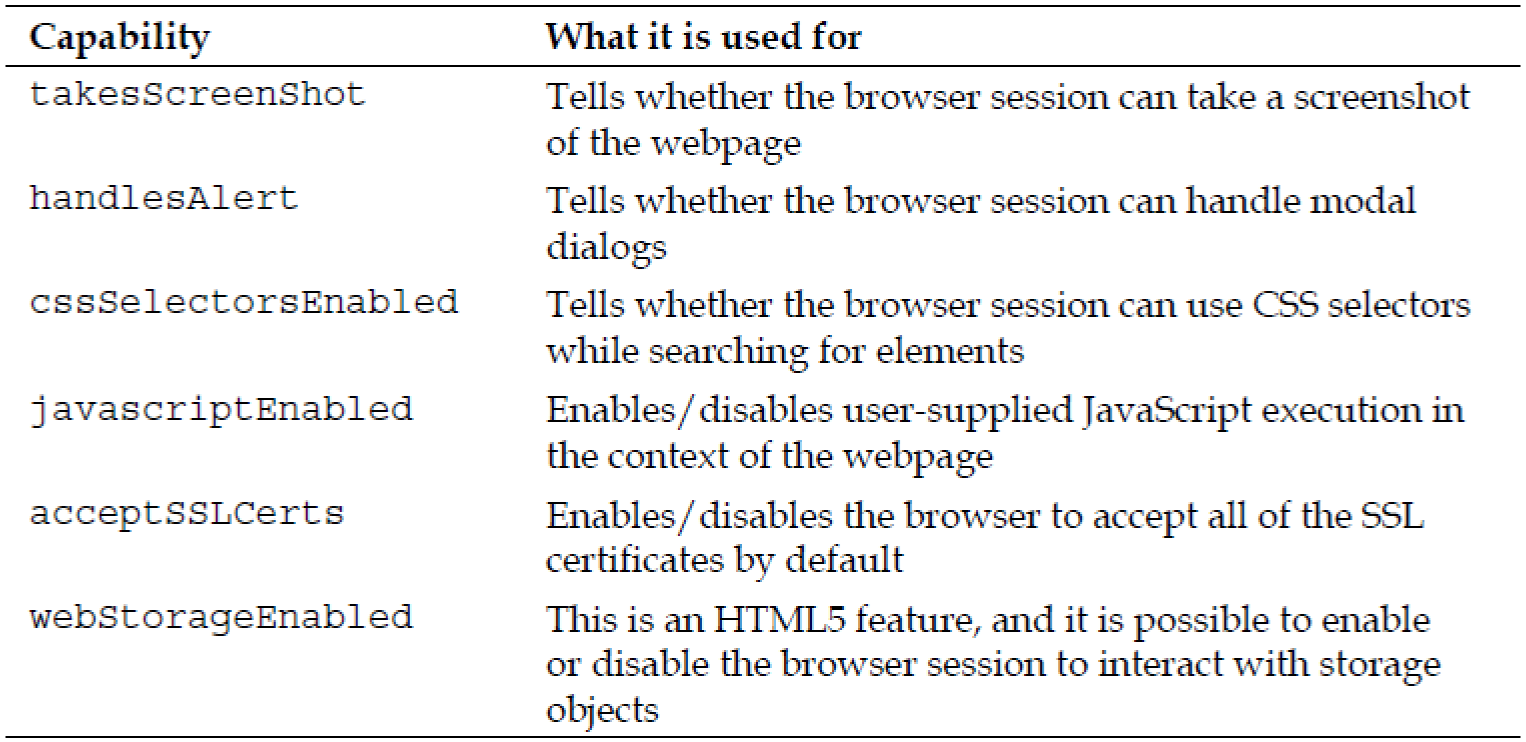
www.infogain.com

- Examples of browser capabilities include enabling a browser session
- Providing support for taking screenshots of the webpage
- Executing custom JavaScript on the webpage,
- Enabling the browser session to interact with window alerts and so on .

Capabilities is an interface in the WebDriver library whose direct implementation is the DesiredCapabilities class.

  - Identify all of the capabilities that you want to arm your browser with.
  - Create a DesiredCapabilities class instance and set all of the capabilities to it.
  - Now, create an instance of WebDriver with all of the above capabilities passed to it.

Capabilities



Capability	What it is used for
`takesScreenShot`	Tells whether the browser session can take a screenshot of the webpage
`handlesAlert`	Tells whether the browser session can handle modal dialogs
`cssSelectorsEnabled`	Tells whether the browser session can use CSS selectors while searching for elements
`javascriptEnabled`	Enables/disables user-supplied JavaScript execution in the context of the webpage
`acceptSSLCerts`	Enables/disables the browser to accept all of the SSL certificates by default
`webStorageEnabled`	This is an HTML5 feature, and it is possible to enable or disable the browser session to interact with storage objects

Locating target windows and iFrames

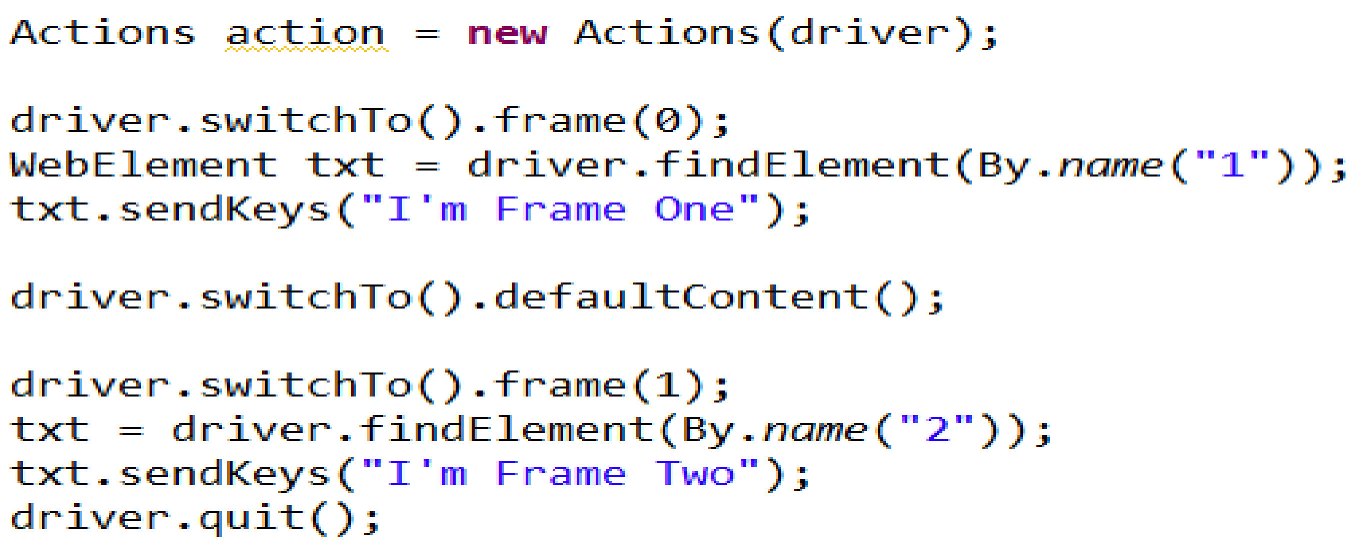
www.infogain.com

- The WebDriver.TargetLocator interface is used to locate a given frame or window .

Switching among windows :

WebDriver.TargetLocator switchTo()

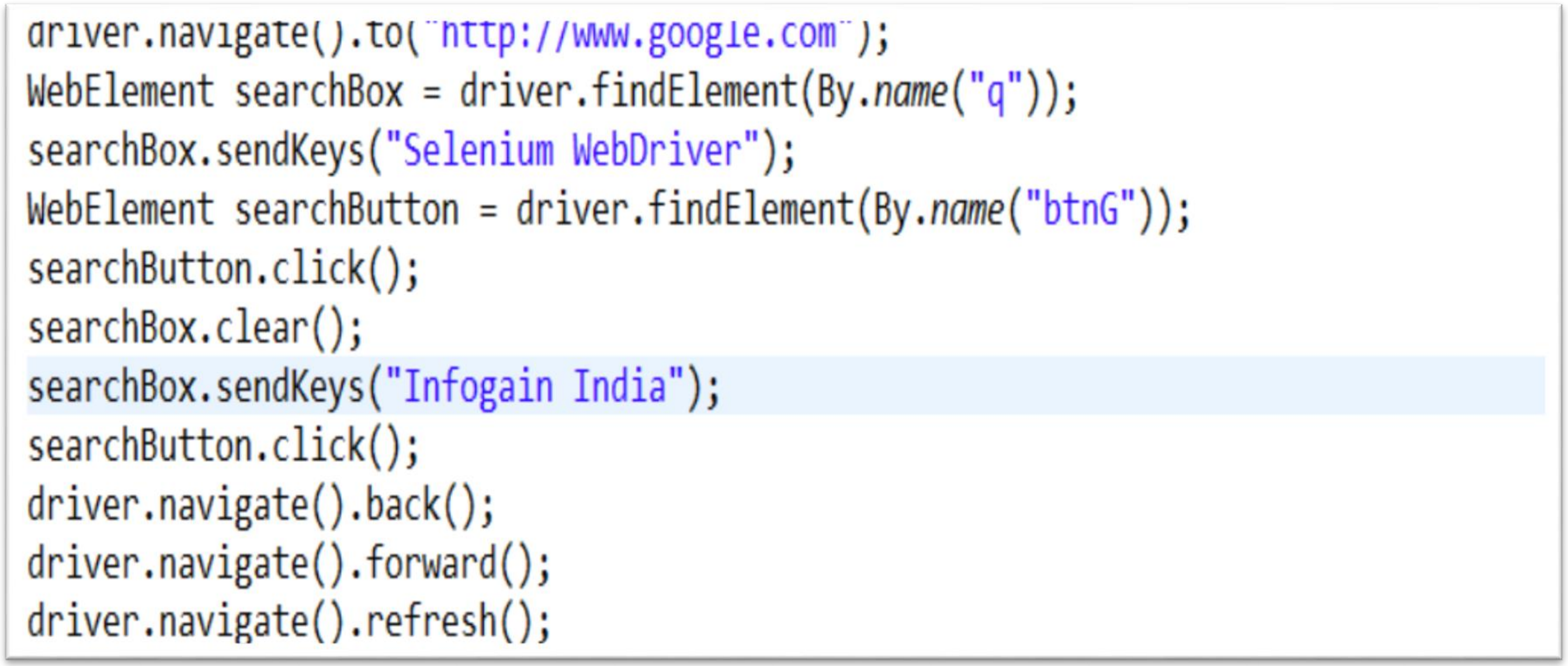
Switching among frames :



```
Actions action = new Actions(driver);

driver.switchTo().frame(0);
WebElement txt = driver.findElement(By.name("1"));
txt.sendKeys("I'm Frame One");

driver.switchTo().defaultContent();

driver.switchTo().frame(1);
txt = driver.findElement(By.name("2"));
txt.sendKeys("I'm Frame Two");
driver.quit();
```

Exploring Navigate

- Navigate is one such feature of WebDriver that allows the test script developer to work with the browser's Back, Forward, and Refresh controls.



```
driver.navigate().to("http://www.google.com");
WebElement searchBox = driver.findElement(By.name("q"));
searchBox.sendKeys("Selenium WebDriver");
WebElement searchButton = driver.findElement(By.name("btnG"));
searchButton.click();
searchBox.clear();
searchBox.sendKeys("Infogain India");
searchButton.click();
driver.navigate().back();
driver.navigate().forward();
driver.navigate().refresh();
```

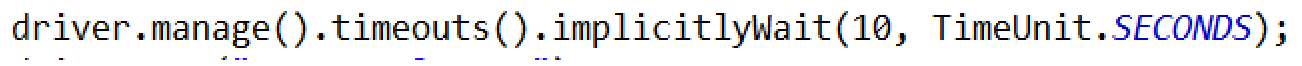
Waiting for WebElements to load

- **WebDriver** provides the test script developers a very handy feature to manage wait time.
- **Wait** time is the time your driver will wait for the WebElement to load before it gives up and throws **NoSuchElementException** .
- There are two ways by which you can make WebDriver wait for WebElement. They are **implicit wait time** and **Explicit wait time**.
- Implicit timeouts are common to all the WebElements and has a global timeout period associated to it.
- The explicit timeouts can be configured to individual WebElements

Waiting for WebElements to load

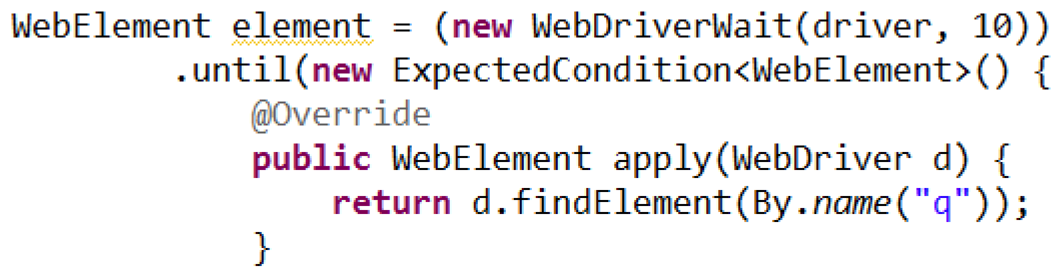
www.infogain.com

- Implicit time out :



```
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Explicit Time Out



```
WebElement element = (new WebDriverWait(driver, 10))
        .until(new ExpectedCondition<WebElement>() {
            @Override
            public WebElement apply(WebDriver d) {
                return d.findElement(By.name("q"));
            }
```

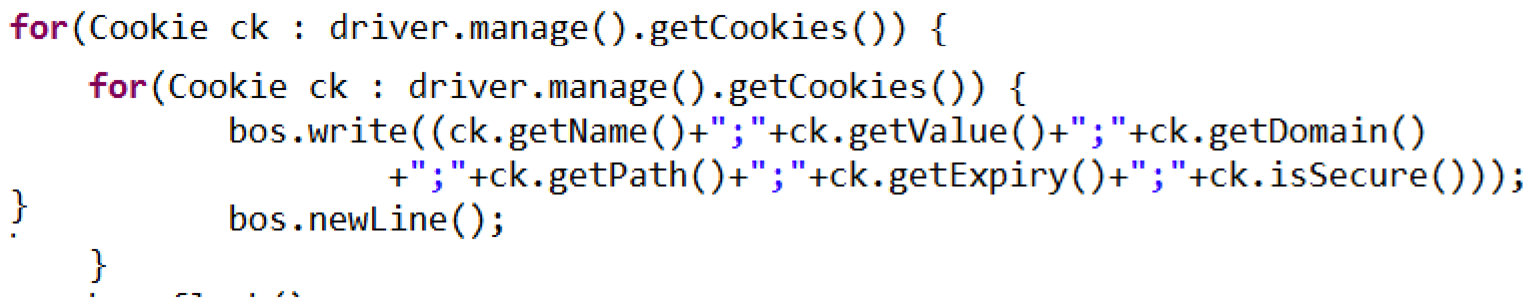
We have created a conditional wait for a particular **WebElement**.
 The **ExpectedCondition** interface can be used to apply the conditional wait on a **WebElement.**

Handling cookies

- To fetch all of the cookies that are loaded for a webpage, WebDriver provides the following method:

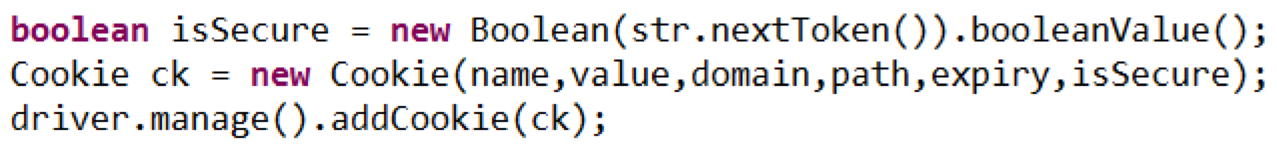
  driver.manage().getCookies()

  Store cookies :



```
for(Cookie ck : driver.manage().getCookies()) {

    for(Cookie ck : driver.manage().getCookies()) {
            bos.write((ck.getName()+";"+ck.getValue()+";"+ck.getDomain()
                    +";"+ck.getPath()+";"+ck.getExpiry()+";"+ck.isSecure()));
}           bos.newLine();
    }
```

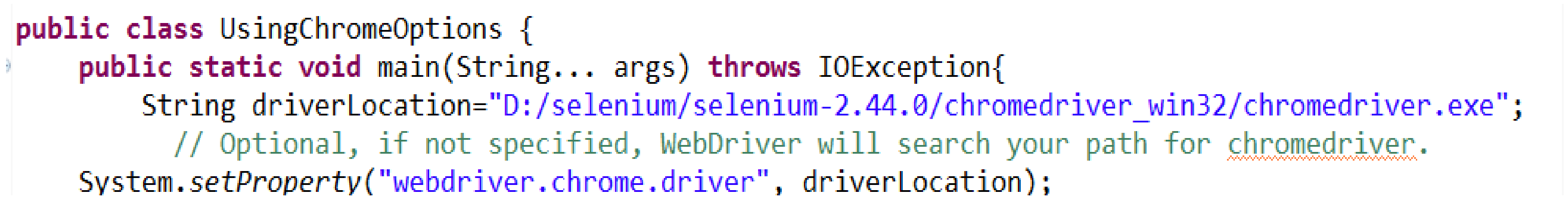
Load Cookies:



```
boolean isSecure = new Boolean(str.nextToken()).booleanValue();
Cookie ck = new Cookie(name,value,domain,path,expiry,isSecure);
driver.manage().addCookie(ck);
```

Different Available WebDrivers

Installing ChromeDriver :


```
public class UsingChromeOptions {
    public static void main(String... args) throws IOException{
        String driverLocation="D:/selenium/selenium-2.44.0/chromedriver_win32/chromedriver.exe";
          // Optional, if not specified, WebDriver will search your path for chromedriver.
    System.setProperty("webdriver.chrome.driver", driverLocation);
```

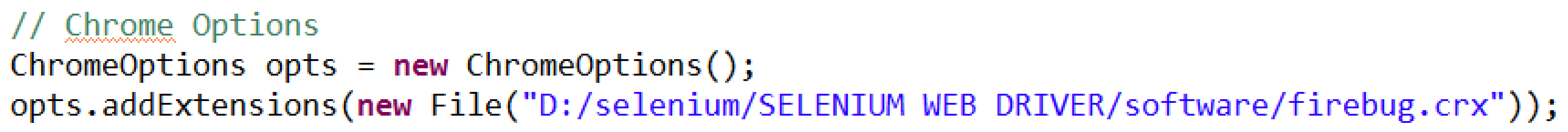
If you execute the above code several times, you will see that the port being assigned to the server changes randomly .

```
ChromeDriverService.Builder builder =  new
  ChromeDriverService.Builder();
ChromeDriverService srvc = builder.usingDriverExecutable(new
  File("C:\\chromedriver_win32_2.2\\chromedriver.exe"))
.usingPort(65423).build();
try {
  srvc.start();
```

Using ChromeOptions

infogain
www.infogain.com

- You can add extensions to your Chrome browser.
- Specify the binary location of the Chrome browser if you have multiple versions of Chrome browsers installed on your machine, and so on .



```java
// Chrome Options
ChromeOptions opts = new ChromeOptions();
opts.addExtensions(new File("D:/selenium/SELENIUM WEB DRIVER/software/firebug.crx"));
```

infogain

Thank You