Selenium-WebDriver-Practical-Guide_Packt_2014

Lewis Cawthorne

May 19, 2015

Contents

Necessary imports

import org.openqa.selenium.WebDriver; import org.openqa.selenium.firefox.FirefoxDriver;

Fetch page: get(String url)

driver.get("http://www.url.com/"); // fetch website

```
package com.packt.webdriver.chapter1;

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.packtpub.com");
    }
}
```

Finding web elements

WebElement findElement(By by)

WebElement searchBox = driver.findElement(By.name("q")); // find by name NoSuchElement Exception if there is no match on current page. Use

findElements instead if you need zero or more matches, instead of only the first match.

By.name(String elementName)

```
 public class GoogleSearchButtonByName {
  public static void main(String[] args){
          WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement searchBox = driver.findElement(By.name("btnK"));
    searchBox.submit();
  }
}
```

By.id(String elementId)

```
public class GoogleSearchButtonById {
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement searchBox = driver.findElement(By.id("gbqfba"));
    searchBox.submit();
  }
}
```

By.tagName(String elementsTag)

More commonly used with findElements, as it is less likely to uniquely match

```
 public class GoogleSearchPageByTagName {
    public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    List<WebElement> buttons =   driver.findElements(By.tagName("button"));
    System.out.println(buttons.size());
  }
}
```

By.className(String elementClass)

More commonly used with CSS styling, class can be unique or match a set of items

```
public class GoogleSearchByClassName{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchBox = driver.findElement(By.className("gbqfif"));
   searchBox.sendKeys("Packt Publishing");
 }
}
```

By.linkText(String linkText)

This is the text of the link as it appears on the webpage, not a property of the <a> tag Example: "About Google"

```
 public class GoogleSearchByLinkText{
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement aboutLink = driver.findElement(By.linkText("About Google"));
    aboutLink.click();
  }
}
```

By.partialLinkText(String link Text)

This matches part of the display link text, Ex: "About"

```
 public class GoogleSearchByPartialLinkText{
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement aboutLink = driver.findElement(By.partialLinkText("About"));
    aboutLink.click();
  }
}
```

By.xpath(String xpath)

XPath is a short name for XML path. Firebug with Firepath makes it easy to find these.

```
public class GoogleSearchByXPath{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton =  driver.findElement(By.xpath("//*[@id='gbqfba']"));
   System.out.println(searchButton.getText());
 }
}
```

- XPath can be very slow, compared to other methods, and requires much scanning!
- The root element is //
- All div elemnts are //div
- Link tags within div elements are //div/a
- All the elemtns with a tag are matched by . **So *//div*** is everything in a div
- All the div elements three levels down are //*/*/div
- //*/div/a[@id='attrValue'] is the anchor element with id attrValue three levels down within a div

By.cssSelector(String elementSelector)

CSS Selector is faster than XPath, but similar.

- To identify an element using the div element with id #flrs, By.cssSelector("#flrs")
- To identify the child anchor element, By.cssSelector("#flrs > a")
- To identify the anchor element with its attribute, By.cssSelector("#flrs>a[a[href='/intl/en/about.h

```
public class GoogleSearchByCSSSelector{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton = driver.findElement(By.cssSelector("#flrs>a[href='/int
   System.out.println(searchButton.getText());
 }
}
```

List<WebElement> findElements(By by)

Same By type clause as findElement, but it doesn't throw an exception on zero matches

Use WebDriver's findElements() method if you think you need all the WebElements that satisfy a locating mechanism condition.

Using Firebug to help determine elts for Selenium

Inspecting a webpage element

Click on Firebug, in the Firebug panel there is a white box with a cursor over it. This is the "Inspect Element" button. Click it, then the element you want to inspect. If we do this for the Google site search box, we get:

<input type="text" name="q" value="" id="searchText" maxlength="256" dir="auto" placeholder="Yahoo" autocomplete="off" aria-autocomplete="true" aria-controls="searchSuggestionTable" aria-expanded="false"/>

So, we could easily find this element by name or id. name, class, or id are normally useful. The developer of the test script needs to determine teh best way to uniquely determine an element or elements.

Finding XPath for By.xpath(String xpath) method

Just click the FirePath tab and click inspect element. The XPath is in the XPath box. Ex: Google search button is XPath: ".//*[@id='searchSubmit']"

Basic Actions on WebElement 'elt'

String elt.getAttribute(String attName)

Web Elements let you fetch attributes with getAttribute(attName)

Some popular attributes include name, class, id, href, aria-label

```
public class GetAttributes{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton = driver.findElement(By.name("btnK"));
   System.out.println("Name of the button is: "
                        +searchButton.getAttribute("name"));
   System.out.println("Id of the button is: "                          +searchButton.ge
   System.out.println("Class of the button is: "
```

```
                        +searchButton.getAttribute("class"));
   System.out.println("Label of the button is: "
                        +searchButton.getAttribute("aria-  label"));
 }
}
```

void elt.sendKeys(CharSequence keysToSend)

Only appropriate for TextBoxes and other character accepting elements.

```
 public class sendKeys{
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement searchBox = driver.findElement(By.name("q"));
    searchButton.sendKeys("Packt Publishing");
  }
}
```

WebDriver contains a Keys enum for special keys

elt.sendKeys(Keys.ENTER);

elt.sendKeys(Keys.chords(Keys.SHIFT, "packt publishing"));

```
public class SendKeys{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchBox = driver.findElement(By.name("q"));
   searchBox.sendKeys(Keys.chord(Keys.SHIFT,"packt publishing"));
 }
}
```

void elt.clear() // erase text in a WebElement; easier than sending multiple Keys.BACK_SPACE

```
public class Clear{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchBox = driver.findElement(By.name("q"));
```

```
   searchBox.sendKeys(Keys.chord(Keys.SHIFT,"packt publishing"));
   searchBox.clear();
 }
}
```

void elt.submit() // submits a form

throws NoSuchElementException if executed on an element not within a form

```
 public class Submit{
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement searchBox = driver.findElement(By.name("q"));
    searchBox.sendKeys(Keys.chord(Keys.SHIFT,"packt publishing"));
    searchBox.submit();
  }
}
```

String elt.getCssValue(String propertyName)

Point elt.getLocation();

```
 public class GetLocation{
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement searchButton = driver.findElement(By.name("btnK"));
    System.out.println(searchButton.getLocation());
  }
 }
// Snippet prints out 372, 356 currently in Google.  The (x,y) location of Google Sear
```

Dimension elt.getSize()

The width and height of the element

```
public class GetSize{
public static void main(String[] args){
  WebDriver driver = new FirefoxDriver();
  driver.get("http://www.google.com");
```

```
  WebElement searchButton = driver.findElement(By.name("btnK"));
  System.out.println(searchButton.getSize());
 }
}
```

String elt.getText()

Returns the visible innerText string of the WebElement or empty string.

```
public class GetText{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton = driver.findElement(By.name("btnK"));
   System.out.println(searchButton.getText());
 }
}
```

String elt.getTagName()

tag name of the web element. The following prints out 'button'

```
 public class GetTagName{
  public static void main(String[] args){
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement searchButton = driver.findElement(By.name("btnK"));
    System.out.println(searchButton.getTagName());
  }
}
```

boolean elt.isDisplayed()

verifies an element is displayed on the page

```
public class isDisplayed{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton = driver.findElement(By.name("btnK"));
   System.out.println(searchButton.isDisplayed());
 }
}
```

boolean elt.isEnabled()

verifies an element is enabled on the page

```
public class isDisplayed{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton = driver.findElement(By.name("btnK"));
   System.out.println(searchButton.isEnabled());
 }
}
```

boolean elt.isSelected()

Verifies a WebElement is selected on the page It only returns true for a radio button, options in select, or checkbox that is selected The following always returns false, since a search box is not one of these.

```
public class isSelected{
 public static void main(String[] args){
   WebDriver driver = new FirefoxDriver();
   driver.get("http://www.google.com");
   WebElement searchButton = driver.findElement(By.name("btnK"));
   System.out.println(searchButton.isSelected());
 }
}
```

Complex Actions

Composite/Multi-step Actions

You can use an Actions class to accomplish this. Let's say you want to hold control while clicking 1, 3, and 5. The following would accomplish this:

```
public class ActionBuildPerform {
    public static void main(String... args) {
      WebDriver driver = new FirefoxDriver();
      driver.get("file://C:/selectable.html");
      WebElement one = driver.findElement(By.name("one"));
      WebElement three = driver.findElement(By.name("three"));
     WebElement five = driver.findElement(By.name("five"));
```

```
      // Add all the actions into the Actions builder.
     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();
      }
   }
```

Actually, the perform action will automatically call build, so the following is equivalent:

```
public class ActionBuildPerform {
    public static void main(String... args) {
      WebDriver driver = new FirefoxDriver();
      driver.get("file://C:/selectable.html");
      WebElement one = driver.findElement(By.name("one"));
      WebElement three = driver.findElement(By.name("three"));
      WebElement five = driver.findElement(By.name("five"));
      // Add all the actions into the Actions builder.
      Actions builder = new Actions(driver);
      builder.keyDown(Keys.CONTROL)
              .click(one)
              .click(three)
              .click(five)
              .keyUp(Keys.CONTROL);
       // Perform the action.
       builder.perform();
  }
}
```

Mouse-based Interactions

This is the web, so you often want to interact using the mouse. There are eight different mouse actions that can be performed using the Actions class.

public Actions moveByOffset(int xOffset, int yOffSet)

The mouse is considered to be at (0, 0) initially, unless some other focus is declared. Offset values are measured in pixels. xOffset moves left for positive, right for negative values. yOffset moves down for positive, up for negative. This throws MoveTargetOutOfBoundsException if you move the cursor out of the document.

In this example, we add 1 to the X and Y because the the element has a 1 pixel border in the example page.

```
public class MoveByOffSet{
 public static void main(String... args) {
   WebDriver driver = new FirefoxDriver();
   driver.get("file://C:/Selectable.html");
   WebElement three = driver.findElement(By.name("three"));
   System.out.println("X coordinate: "+three.getLocation().getX()
                      +"  Y coordinate: "+three.getLocation().getY());
   Actions builder = new Actions(driver);
   builder.moveByOffset(three.getLocation().getX()+1, three.getLocation().getY()+1);
   builder.perform();
 }
}
```

public Actions click()

```
public class MoveByOffsetAndClick{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Selectable.html");
    WebElement seven = driver.findElement(By.name("seven"));
    System.out.println("X coordinate: "+seven.getLocation().getX()
                       +" Y coordinate: "+seven.getLocation().getY());
    Actions builder = new Actions(driver);
    builder.moveByOffset(seven.getLocation().getX()+1, seven.getLocation().getY()+1).c
    builder.perform();
  }
}
```

A more complex usage (which we see a much cleaner version of in click(onElement) desc) follows:

```
public class MoveByOffsetAndClick{
```

```
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Selectable.html");
    WebElement one = driver.findElement(By.name("one"));
    WebElement eleven = driver.findElement(By.name("eleven"));
    WebElement five = driver.findElement(By.name("five"));
    int border = 1;
    int tileWidth = 100;
    int tileHeight = 80;
    Actions builder = new Actions(driver);
    //Click on One
    builder.moveByOffset(one.getLocation().getX()+border, one.getLocation().getY()+bor
    builder.build().perform();
    // Click on Eleven
    builder.moveByOffset(2*tileWidth+4*border, 2*tileHeight+4*border).click();
    builder.build().perform();
    //Click on Five
    builder.moveByOffset(-2*tileWidth-4*border,-tileHeight-2*border).click();
    builder.build().perform();
   }
 }
```

public Actions click(WebElement onElement)

This overloaded version directly clicks an element without having to work with offsets.

```
public class ClickOnWebElement{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Selectable.html");
    WebElement one = driver.findElement(By.name("one"));
    WebElement eleven = driver.findElement(By.name("eleven"));
    WebElement five = driver.findElement(By.name("five"));
    Actions builder = new Actions(driver);
    //Click on One
    builder.click(one);
    builder.build().perform();
    // Click on Eleven
    builder.click(eleven);
```

```
    builder.build().perform();
    //Click on Five
    builder.click(five)
    builder.build().perform();
  }
}
```

This is a lot cleaner than the example in the click() method, and is a great reason to set identifiers for WebElements. We can simplify this example yet again by building all the actions before performing them as follows:

```
public class ClickOnWebElement{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Selectable.html");
    WebElement one = driver.findElement(By.name("one"));
    WebElement eleven = driver.findElement(By.name("eleven"));
    WebElement five = driver.findElement(By.name("five"));
    Actions builder = new Actions(driver);
    //Click on One, Eleven and Five
    builder.click(one).click(eleven).click(five);
    builder.build().perform();
  }
}
```

public Actions clickAndHold()

This is equivalent to clicking somewhere and holding down the mouse button. You generally want to move somewhere else afterwards to highlight a region. You should use release at some point (likely very soon) after using this.

```
public class ClickAndHold{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Sortable.html");
    Actions builder = new Actions(driver);
    //Move tile3 to the position of tile2
    builder.moveByOffset(200, 20)
           .clickAndHold()
           .moveByOffset(120, 0)
           .perform();
```

```
  }
}
```

public Actions clickAndHold(WebElement onElement)

```
public class ClickAndHold{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Sortable.html");
    WebElement three = driver.findElement(By.name("three"));
    Actions builder = new Actions(driver);
    //Move tile3 to the position of tile2
    builder.clickAndHold(three)
           .moveByOffset(120, 0)
           .perform();
  }
}
```

public Actions release()

Note: using either of these release methods without calling clickAndHold() first results in undefined behavior.

```
public class ClickAndHoldAndRelease{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Sortable.html");
    WebElement three = driver.findElement(By.name("three"));
    Actions builder = new Actions(driver);
    //Move tile3 to the position of tile2
    builder.clickAndHold(three)
           .moveByOffset(120, 0)
           .release()
           .perform();
  }
}
```

public Actions release(WebElement onElement)

Note: using either of these release methods without calling clickAndHold() first results in undefined behavior.

```
public class ClickAndHoldAndReleaseOnWebElement{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/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();
  }
}
```

public Actions moveToElement(WebElement toElement)

Moves the mouse cursor to the specified element on the webpage.

```
public class ClickAndHold{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/Sortable.html");
    WebElement three = driver.findElement(By.name("three"));
    Actions builder = new Actions(driver);
    //Move tile3 to the position of tile2
    builder.moveToElement(three)
           .clickAndHold()
           .moveByOffset(120, 0)
           .release()
           .perform();
  }
}
```

public Actions dragAndDropBy(WebElement source, int xOffset, int yOffset)

This drags and drops the source element by x,y pixels.

Ex: Drag and drop elt where id=draggable by 300px horizontally, 200px vertically.

```
public class DragMe {
```

```
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/DragMe.html");
    WebElement dragMe = driver.findElement(By.id("draggable"));
    Actions builder = new Actions(driver);
    builder.dragAndDropBy(dragMe, 300, 200).perform();
  }
}
```

public Actions dragAndDrop(WebElement source, WebElement target)

This drags and drops source element onto target element.

```
public class DragAndDrop {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/DragAndDrop.html");
    WebElement src = driver.findElement(By.id("draggable"));
    WebElement trgt = driver.findElement(By.id("droppable"));
    Actions builder = new Actions(driver);
    builder.dragAndDrop(src, trgt).perform();
  }
}
```

public Actions doubleClick()

Double click whatever the mouse is over.

```
public class DoubleClick {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/DoubleClick.html");
    WebElement dblClick= driver.findElement(By.name("dblClick"));
    Actions builder = new Actions(driver);
    builder.moveToElement(dblClick).doubleClick().perform();
  }
}
```

public Actions doubleClick(WebElement onElement)

Often we want to doubleClick on some element.

```
public class DoubleClick {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/DoubleClick.html");
    WebElement dblClick = driver.findElement(By.name("dblClick"));
    Actions builder = new Actions(driver);
    builder.doubleClick(dblClick).perform();
  }
}
```

public Actions contextClick(WebElement onElement)

```
public class ContextClick {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/ContextClick.html");
    WebElement contextMenu = driver.findElement(By.id("div-context"));
    Actions builder = new Actions(driver);
    builder.contextClick(contextMenu)
       .click(driver.findElement(By.name("Item 4")))
           .perform();
  }
}
```

public Actions contextClick()

This moves to div-context, context clicks it (right click), then selects Item 4 from the list by name.

```
public class ContextClick {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("file://C:/ContextClick.html");
    WebElement contextMenu = driver.findElement(By.id("div-context"));
    Actions builder = new Actions(driver);
    builder.moveToElement(contextMenu)
           .contextClick()
            .click(driver.findElement(By.name("Item 4")))
           .perform();
  }
}
```

Keyboard-based interactions

public Actions keyDown(Keys theKey) throws IllegalArgumentException

The only legal arguments are one of Shift, Ctrl, and Alt

public Actions keyUp(Keys theKey)

Causes unexpected results if we have not already held the key down

public Actions sendKeys(CharSequence keysToSend)

Waiting for WebElements to load

Implicit Wait Time

Used to configure wait time for all get/find operations of application under test. The driver throws NoSuchElementException when time is exceeded.

```
public class ImplicitWaitTime {
   public static void main(String... args) {
      WebDriver driver = new FirefoxDriver();
      driver.manage().timeouts().implicitlyWait(10,   TimeUnit.SECONDS);
      driver.get("www.google.com");
   }
}
```

Explicit Wait Time

Sometimes you have a single element you want to wait longer for, so you set an explicit wait time.

```
public class ExplicitWaitTime {
   public static void main(String... args) {
      WebDriver driver = new FirefoxDriver();
      driver.get("http://www.google.com");
      WebElement element = (new WebDriverWait(driver, 20)).until(new ExpectedCondition
         @Override
         public WebElement apply(WebDriver d) {
            return d.findElement(By.name("q"));
         }
```

```
        });
   }
}
```

Windows and iFrames

Multiple Windows

String window1 = driver.getWindowHandle(); */ store a handle for the current window driver.switchTo().window(window1); /* switch back to the window driver.getWindowHandles(); // returns all window handles opened by driver so far Ex:

```
public class WindowHandling {
   public static void main(String... args){
      WebDriver driver = new FirefoxDriver();
      driver.get("file://C:/Window.html");

      String window1 = driver.getWindowHandle();
      System.out.println("First Window Handle is: "+window1);

      WebElement link = driver.findElement(By.linkText("Google Search"));
      link.click();

      String window2 = driver.getWindowHandle();
      System.out.println("Second Window Handle is: "+window2);
      System.out.println("Number of Window Handles so for: "
            +driver.getWindowHandles().size());

      driver.switchTo().window(window1);

   }
}
```

Multiple Frames

driver.switchTo().frame(int index); driver.switchTo().frame(Strign frameNameOrFrameId); driver.switchTo().frame(WebElement frameElement); driver.switchTo().defaultConte
// you need to use before switching to another frame!

```
public class SwitchBetweenFrames {
```

```
   public static void main(String... args) {
      WebDriver driver = new FirefoxDriver();
      driver.get("file://C:/Frames.html");

      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");
   }
}
```

Handling Alerts is covered under special capabilities.

Navigation

Navigate allows WebDriver to use native browse Back/Forward/Refresh, among other things. WebDriver.Navigation navigate() allows us to get an object to use these features.

```
public class WebDriverNavigate{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    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("Packt Publishing");
    searchButton.click();

    driver.navigate().back();
```

```
    driver.navigate().forward();
    driver.navigate().refresh();
  }
}
```

Navigation object methods

void driver.navigate().to(String url); void driver.navigate().back(); void driver.navigate().forward(); void driver.navigate().refresh();

Cookies

Fetching all loaded cookies for a page: driver.manage().getCookies();

Full program to store all current session cookie information to a 'browser.data' file

```
package com.packt.webdriver.chapter3;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;

import org.openqa.selenium.By;
import org.openqa.selenium.Cookie;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class StoreCookieInfo {

  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.facebook.com");
    driver.findElement(By.name("email")).sendKeys("<<ur mailID>>");
    driver.findElement(By.name("pass")).sendKeys("<<ur password>>");
    driver.findElement(By.name("persistent")).click();
    driver.findElement(By.name("pass")).submit();

    File f = new File("browser.data");
    try{
```

```
         f.delete();
         f.createNewFile();
         FileWriter fos = new FileWriter(f);
         BufferedWriter bos = new BufferedWriter(fos);

         for(Cookie ck : driver.manage().getCookies()) {
                      bos.write((ck.getName()+";"+ck.getValue()+";"+ck.getDomain()
                     +";"+ck.getPath()+";"+ck.getExpiry()+";"+ck.isSecure()));
                    bos.newLine();
         }
         bos.flush();
         bos.close();
         fos.close();
     }catch(Exception ex){
         ex.printStackTrace();
     }

   }
}
```

Load a stored cookied: driver.manage().addCookie(ck);

```
package com.packt.webdriver.chapter3;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.Date;
import java.util.StringTokenizer;

import org.openqa.selenium.Cookie;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class LoadCookieInfo {

   public static void main(String... args){
      WebDriver driver = new FirefoxDriver();
      driver.get("http://www.facebook.com");
      try{
```

```
            File f = new File("browser.data");
            FileReader fr = new FileReader(f2);
            BufferedReader br = new BufferedReader(fr);
            String line;
            while((line=br.readLine())!=null){
                StringTokenizer str = new StringTokenizer(line,";");
                while(str.hasMoreTokens()){
                    String name = str.nextToken();
                    String value = str.nextToken();
                    String domain = str.nextToken();
                    String path = str.nextToken();
                    Date expiry = null;
                    String dt;
                    if(!(dt=str.nextToken()).equals("null")){
                        expiry = new Date(dt);
                    }
                    boolean isSecure = new Boolean(str.nextToken()).booleanValue();
                    Cookie ck = new Cookie(name,value,domain,path,expiry,isSecure);
                    driver.manage().addCookie(ck);
                }
            }
        }catch(Exception ex){
            ex.printStackTrace();
        }
        driver.get("http://www.facebook.com");
    }
}
```

The two driver.get calls above are necessary, since you need to go to the site, then load the cookies for the site, then reload the site.

Special capabilities

Adding Capabilities to the WebDriver

Capabilities is an interface implemented in the DesiredCapabilities class

```
public class BrowserCapabilities {
  public static void main(String... args) {
    Map capabilitiesMap = new HashMap();
```

```
    capabilitiesMap.put("takesScreenShot", true);
    DesiredCapabilities capabilities
              = new DesiredCapabilities(capabilitiesMap);
    WebDriver driver = new FirefoxDriver(capabilities);
    driver.get("http://www.google.com");
  }
}
```

Common capabilities

takesScreenShot

handlesAlert

cssSelectorsEnabled

javascriptEnabled

acceptSSLCerts

webStorageEnabled

Using special capabilities

Screenshots

- Enabling

```
public class BrowserCapabilities {
  public static void main(String... args) {
    Map capabilitiesMap = new HashMap();
    capabilitiesMap.put("takesScreenShot", true);
    DesiredCapabilities capabilities
              = new DesiredCapabilities(capabilitiesMap);
    WebDriver driver = new FirefoxDriver(capabilities);
    driver.get("http://www.google.com");
  }
}
```

- API public <X> X getScreenshotAs(OutputType<X> target)

  Output type is one of: OutputType.BASE64, OutputType.BYTES (raw data), OutputType.FILE

- Using

```
public class TakesScreenShotExample{
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.packtpub.com/");
    File scrFile = ((TakesScreenShot)driver).getScreenshotAs(OutputType.FILE);
    System.out.println(scrFile.getAbsolutePath());
    try {
      FileHandler.copy(scrFile, new File("C:\\Dest\\"));
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

- Warning The file returned for OutputType.FILE is temporarily and is deleted once the JVM exits, so you will need to copy the file somewhere before the test completes.
- Browser dependency The method tries to take as large a screenshot as possible, but what it can actually do is dependent upon the browser. In Firefox, the screenshot will be of the entire page. In Chrome, it will take a picture of the visibile area.

Handling alerts

- Enabling

```
public class BrowserCapabilities {
  public static void main(String... args) {
    Map capabilitiesMap = new HashMap();
    capabilitiesMap.put("handlesAlert", true);
    DesiredCapabilities capabilities
              = new DesiredCapabilities(capabilitiesMap);
    WebDriver driver = new FirefoxDriver(capabilities);
    driver.get("http://www.google.com");
  }
}
```

- API Alert alert(): switch to currently active modal dialog on the page or throw NoAlertPresentException

void accept(): equivalent to OK on dialog void dismiss(): equivalent to CANCEL String getText(): return text in dialog void sendKeys(String keysToSend): type text into alert

Firefox Specific Features

Using a specific Firefox Profile

The driver creates a new anonymous profile each time it launches a browser. Instead, we can use a specific profile with a FirefoxProfile object: public FirefoxProfile(java.io.File profileDir)

```
public class FirefoxCustomProfile {

  public static void main(String... args){
    FirefoxProfile profile = new FirefoxProfile();

    FirefoxDriver driver = new FirefoxDriver(profile);
    driver.get("http://www.google.com");
  }
}
```

Installing a browser extension in the WebDriver profile

Profile objects have an addExtension method. It will throw IOException if it cannot find the file.

```
public class AddExtensionToProfile {
  public static void main(String... args){
    FirefoxProfile profile = new FirefoxProfile();

    try {
      profile.addExtension(new File("C:\\firebug-1.12.0-fx.xpi"));
    } catch (IOException e) {
      e.printStackTrace();
    }

    FirefoxDriver driver = new FirefoxDriver(profile);
  }
}
```

Setting Profile Preferences

```
public class SettingPreferences {

  public static void main(String... args){
    FirefoxProfile profile = new FirefoxProfile();

    profile.setPreference("general.useragent.override", "Mozilla/5.0 (iPhone; U; CPU i

    FirefoxDriver driver = new FirefoxDriver(profile);
    driver.get("http://www.google.com");
  }
}
```

Testing with Events

Flow of Setting up to Capture Events during Test Execution

1. Create an EventListener Class
2. Create a WebDriver instance
3. Create an instance of EventFiringWebDriver to wrap the WebDriver
4. Create an instance of EventListener class created above
5. Register the EventListener with the EventFiringWebDriver
6. Execute the events with the EventFiringWebDriver
7. Verify Listener got informed of events occurence

Creating an instance of EventListener (done one of two ways)

You can do this either way, and experience will help you choose which might be easiest for a situation

Implementing WebDriverEventListener (more verbose)

Requires you to implement the 15 methods declared in the WebDriverEventListener interface

```
public class IAmTheEventListener implements WebDriverEventListener{

////////// NAVIGATION RELATED METHODS ////////////////
   @Override
   public void beforeNavigateTo(String url, WebDriver driver) {
      System.out.println("Before Navigate To "+url);
   }

   @Override
   public void afterNavigateTo(String url, WebDriver driver) {
      // TODO Auto-generated method stub
   }

   @Override
   public void beforeNavigateBack(WebDriver driver) {
      System.out.println("Before Navigate Back. Right now I'm at "+driver.getCurrentUr
   }

   @Override
   public void afterNavigateBack(WebDriver driver) {
      // TODO Auto-generated method stub
   }

   @Override
   public void beforeNavigateForward(WebDriver driver) {
      // TODO Auto-generated method stub
   }

   @Override
   public void afterNavigateForward(WebDriver driver) {
      // TODO Auto-generated method stub
   }

////////////////// FINDBY RELATED METHODS ///////////////
   @Override
   public void beforeFindBy(By by, WebElement element, WebDriver driver) {
      // TODO Auto-generated method stub
   }

   @Override
```

```
    public void afterFindBy(By by, WebElement element, WebDriver driver) {
        // TODO Auto-generated method stub
    }

////////////////////// CLICKON RELATED METHODS ///////////////
    @Override
    public void beforeClickOn(WebElement element, WebDriver driver) {
        // TODO Auto-generated method stub
    }

    @Override
    public void afterClickOn(WebElement element, WebDriver driver) {
        // TODO Auto-generated method stub
    }

////////////////// CHANGE OF VALUE RELATED METHODS //////////////
    @Override
    public void beforeChangeValueOf(WebElement element, WebDriver driver) {
        // TODO Auto-generated method stub
    }

    @Override
    public void afterChangeValueOf(WebElement element, WebDriver driver) {
        // TODO Auto-generated method stub
    }

/////////////// SCRIPT EXECUTION RELATED METHODS ///////////////
    @Override
    public void beforeScript(String script, WebDriver driver) {
        // TODO Auto-generated method stub
    }

    @Override
    public void afterScript(String script, WebDriver driver) {
        // TODO Auto-generated method stub
    }

/////////////// EXCEPTION RELATED METHODS ///////////////////////
    @Override
    public void onException(Throwable throwable, WebDriver driver) {
```

```
      // TODO Auto-generated method stub
   }
}
```

Extending the AbstractWebDriverEventListener

Dummy implementations of all 15 methods in the above interface are provided, so you only have to override the methods you are interested in.

```
public class IAmTheEventListener2 extends AbstractWebDriverEventListener{

   @Override
   public void beforeNavigateTo(String url, WebDriver driver) {
      System.out.println("Before Navigate To "+url);

   }

   @Override
   public void beforeNavigateBack(WebDriver driver) {
      System.out.println("Before Navigate Back. Right now I'm at "+driver.getCurrentUr

   }
}
```

Example with Custom Listener and Registered Driver

// CustomEventListener.java

```
package com.packt.webdriver;

import org.openqa.selenium.support.events.AbstractWebDriverEventListener;
import org.openqa.selenium.WebDriver;

public class CustomEventListener extends AbstractWebDriverEventListener{

  @Override
  public void beforeNavigateTo(String url, WebDriver driver) {
     System.out.println("Before Navigate To "+url);

  }
```

```
  @Override
  public void beforeNavigateBack(WebDriver driver) {
     System.out.println("Before Navigate Back. Right now I'm at "+driver.getCurrentUrl

  }

}
```

// CustomEventFiringDriver.java

```
package com.packt.webdriver;

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.events.EventFiringWebDriver;

public class CustomEventFiringDriver {
   public static void main(String... args){
      WebDriver driver = new FirefoxDriver();

      EventFiringWebDriver eventFiringDriver = new EventFiringWebDriver(driver);
      CustomEventListener eventListener = new CustomEventListener();
      eventFiringDriver.register(eventListener);

      eventFiringDriver.get("http://www.google.com");
      eventFiringDriver.get("http://www.facebook.com");
      eventFiringDriver.navigate().back();

   }
}
```

Skipped Chapters: Dealing with I/O (FileHandler, isZipped, etc), RemoteWebDriver, Selenium Grid

Understanding the Page Object Pattern

WordPress blog Sample Test Cases - Fine Tests that are manual to maintain.

Each of the following involves logging in to WordPress and then performing some actions. If an ID element on the login page changes, all three java

classes would need modification. They are excellent examples of successfully using the Selenium API in a hard to maintain fashion.

Adding a post to WordPress site `http://pageobjectpattern.wordpress.com/`

1. Log in to the WordPress Admin portal.
2. Go to the All Posts page.
3. Click on the Add New post button.
4. Add a new post by providing the title and description.
5. Publish the post.

```
public class TestAddNewPost {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    // Login to Admin portal
    driver.get("http://pageobjectpattern.wordpress.com/wp-admin");
    WebElement email = driver.findElement(By.id("user_login"));
    WebElement pwd = driver.findElement(By.id("user_pass"));
    WebElement submit = driver.findElement(By.id("wp-submit"));
    email.sendKeys("pageobjectpattern@gmail.com");
    pwd.sendKeys("webdriver123");
    submit.click();
    // Go to AllPosts page
    driver.get("http://pageobjectpattern.wordpress.com/wp-admin/edit.php");
    // Add New Post
    WebElement addNewPost = driver.findElement(By.linkText("Add New"));
    addNewPost.click();
    // Add New Post's Content
    driver.switchTo().frame("content_ifr");
    WebElement postBody = driver.findElement(By.id("tinymce"));
    postBody.sendKeys("This is description");
    driver.switchTo().defaultContent();
    WebElement title = driver.findElement(By.id("title"));
    title.click();
    title.sendKeys("My First Post");
    // Publish the Post
    WebElement publish = driver.findElement(By.id("publish"));
    publish.click();
```

```
  }
}
```

Deleting a post

1. Log in to the WordPress Admin portal.
2. Go to the All Posts page.
3. Click on the post to be deleted.
4. Delete the post.

```
public class TestDeletePost {
  public static void main(String... args) {
    WebDriver driver = new FirefoxDriver();
    // Login to Admin portal
    driver.get("http://pageobjectpattern.wordpress.com/wp-admin");
    WebElement email = driver.findElement(By.id("user_login"));
    WebElement pwd = driver.findElement(By.id("user_pass"));
    WebElement submit = driver.findElement(By.id("wp-submit"));
    email.sendKeys("pageobjectpattern@gmail.com");
    pwd.sendKeys("webdriver123");
    submit.click();
    // Go to a All Posts page
    driver.get("http://pageobjectpattern.wordpress.com/wp-admin/edit.php");
    // Click on the post to be deleted
    WebElement post = driver.findElement(By.linkText("My First Post"));
    post.click();
    // Delete Post
    WebElement publish = driver.findElement(By.linkText("Move to Trash"));
    publish.click();
  }
}
```

Counting the number of posts

1. Log in to the Admin portal.
2. Go to the All Posts page.
3. Count the number of posts available.

#+begin_src java public class TestPostsCount { public static void main(String... args){ WebDriver driver = new FirefoxDriver(); // Login to Admin portal driver.get("http://pageobjectpattern.wordpress.com/wp-admin"); WebElement email = driver.findElement(By.id("user_login")); WebElement pwd = driver.findElement(By.id("user_pass")); WebElement submit = driver.findElement(By.id("wp-submit")); email.sendKeys("pageobjectpattern@gmail.com"); pwd.sendKeys("webdriver123"); submit.click(); // Count the number of posts. driver.get("http://pageobjectpattern.wordpress.com/wp-admin/edit.php"); WebElement postsContainer = driver.findElement(By.id("the-list")); List postsList = postsContainer.findElements(By.tagName("tr")); System.out.println(postsList.size()); } } #+end_sr

WordPress blog using PageObject pattern

AdminLoginPage

```
public class AdminLoginPage {
    WebDriver driver;
    WebElement email;
   WebElement password;
    WebElement submit;
    public AdminLoginPage(WebDriver driver){
        this.driver = driver;
        driver.get("http://pageobjectpattern.wordpress.com/wp-admin");
email = driver.findElement(By.id("user_login"));
password = driver.findElement(By.id("user_pass"));
submit = driver.findElement(By.id("wp-submit"));
    }
    public void login(){
        email.sendKeys("pageobjectpattern@gmail.com");
        password.sendKeys("webdriver123");
        submit.click();
    }
}
```

TestAddNewPostUsingPageObject

```
public class TestAddNewPostUsingPageObject {
   public static void main(String... args) {
       WebDriver driver = new FirefoxDriver();
       // Login to Admin portal - factored into other object only be two lines here
```

```
        AdminLoginPage admLoginPage = new AdminLoginPage(driver);
        admLoginPage.login();
        // Go to New Posts page
        driver.get("http://pageobjectpattern.wordpress.com/wp-admin/edit.php");
        WebElement addNewPost = driver.findElement(By.linkText("Add New"));
        addNewPost.click();
        // Add New Post
        driver.switchTo().frame("content_ifr");
        WebElement postBody = driver.findElement(By.id("tinymce"));
        postBody.sendKeys("This is description");
        driver.switchTo().defaultContent();
        WebElement title = driver.findElement(By.id("title"));
        title.click();
        title.sendKeys("My First Post");
        WebElement publish = driver.findElement(By.id("publish"));
        publish.click();
    }
}
```

Using the @FindBy annotation to locate elements on a page

There are two ways of using the @FindBy annotation: Usage 1 is shown as follows: @FindBy(id="user_login") WebElement userId; Usage 2 is shown as follows: @FindBy(how=How.ID, using="user_login") WebElement userId;

Different ways to @FindBy supported by the How enumeration

- CLASS_NAME
- CSS
- ID
- ID_OR_NAME
- LINK_TEXT
- NAME
- PARTIAL_LINK_TEXT
- TAG_NAME

- XPATH

Wordpress AdminLoginPage using How and @FindBy

```
public class AdminLoginPage {
    WebDriver driver;
    @FindBy(how=How.ID, id="user_login")
    WebElement email;
    @FindBy(how=How.ID, id="user_pass")
    WebElement password;
    @FindBy(how=How.ID, id="wp-submit")
    WebElement submit;
    public AdminLoginPage(WebDriver driver){
        this.driver = driver;
        driver.get("http://pageobjectpattern.wordpress.com/wp-admin");
    }
    public void login(){
        email.sendKeys("pageobjectpattern@gmail.com");
        password.sendKeys("webdriver123");
        submit.click();
    }
}
```

Using the PageFactory class

A PageFactory instantiates a Page object that uses the @FindBy annotations and gives it a driver

initElements(WebDriver driver, java.lang.Class PageObjectClass)

Example Use

```
public class TestAddNewPostUsingPageObjects {
    public static void main(String... args){
        WebDriver driver = new FirefoxDriver();
        AdminLoginPage loginPage= PageFactory.initElements(driver, AdminLoginPage.class
        loginPage.login();
    }
}
```

Good practices for using PageObjects

Use PageObjects within PageObjects

- Example **AllPostsPage**

```
public class AllPostsPage {
    WebDriver driver;
    @FindBy(how=How.ID, using="the-list")
    WebElement postsContainer;
    @FindBy(how=How.ID, using="post-search-input")
    WebElement searchPosts;
    @FindBy(how=How.ID, using="cat")
    WebElement viewByCategories;
    public AllPostsPage(WebDriver driver){
        this.driver = driver;
            driver.get("http://pageobjectpattern.wordpress.com/wp-admin/edit.php")
    }
    public void createANewPost(String title, String description) {
        addNewPost.click();
        // use an AddNewPost object here
        AddNewPost newPost = PageFactory.initElements(driver, AddNewPost.class);
        newPost.addNewPost(title, description);
    }
    public void editAPost(String title){
    }
    public void deleteAPost(String postTitle) {
    }
    public void filterPostsByCategory(String category){
    }
    public void searchInPosts(String searchText){
    }
    public int getAllPostsCount(){
    }
  }
/* In AddNewPost.java */
public class AddNewPost {
    WebDriver driver;
    @FindBy(how=How.ID, using="content_ifr")
    WebElement newPostContentFrame;
    @FindBy(how=How.ID, using="tinymce")
```

```
        WebElement newPostContentBody;
        @FindBy(how=How.ID, using="title")
        WebElement newPostTitle;
        @FindBy(how=How.ID, using="publish")
        WebElement newPostPublish;
        public AddNewPost(WebDriver driver){
            this.driver = driver;
            System.out.println(driver.getCurrentUrl());
        }
        public void addNewPost(String title, String descContent){
             driver.switchTo().frame(newPostContentFrame);
             newPostContentBody.sendKeys(descContent);
             driver.switchTo().defaultContent();
             newPostTitle.click();
             newPostTitle.sendKeys(title);
             newPostPublish.click();
        }
    }
```

Consider methods in PageObjects as services and not as User Actions

At a high level, a webpage is a collection of services (such as 'add a post', 'edit a post', etc). So a PageObject should provide these services to a test case. Be sure to include "implied services" that are provided simply by reading the page (like "count posts" if there is a post listing).

Using a service can (and often will) involve multiple user actions. You should abstract this layer. Example: Adding a post requires "type text in title", "type text in content", and "click publish" actions.

Initialize some elements as required; especially lists (source example)

```
public class AllPostsPage {
    WebDriver driver;
    @FindBy(how=How.ID, using="the-list")
    WebElement postsContainer;
    public void editAPost(String presentTitle,String newTitle, String description){
        List<WebElement> allPosts= postsContainer.findElements(By.className("row-title
        for(WebElement ele : allPosts){
```

```
        if(ele.getText().equals(presentTitle)){
                Actions builder = new Actions(driver);
                builder.moveToElement(ele);
                builder.click(driver.findElement(By.cssSelector(".edit>a")));
                // Generate the composite action.
                Action compositeAction = builder.build();
                // Perform the composite action.
                compositeAction.perform();
                break;
            }
        }
        EditPost editPost= PageFactory.initElements(driver, EditPost.class);
        editPost.editPost(newTitle, description);
    }
}
```

Keep page specific details of the test script (main goal!)

Checking load()/isLoad() with LoadableComponent class

load specifies a url that should be loaded

isLoaded checks that the page specified in load() came up

the get method attempts the load() and does the isLoaded() check

Full example of use

```
package com.packt.webdriver.chapter9.pageObjects;
import org.junit.Assert;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.CacheLookup;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
import org.openqa.selenium.support.PageFactory;
import org.openqa.selenium.support.ui.LoadableComponent;
public class AdminLoginPageUsingLoadableComponent extends LoadableComponent<AdminLogin
    WebDriver driver;
    @FindBy(how=How.ID, using="user_login")
    WebElement email;
    @FindBy(how=How.ID, using="user_pass")
```

```
    WebElement password;
    @FindBy(how=How.ID, using="wp-submit")
    WebElement submit;
    public AdminLoginPageUsingLoadableComponent(WebDriver driver){
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }
    public AllPostsPage login(){
        email.sendKeys("pageobjectpattern@gmail.com");
        password.sendKeys("webdriver123");
        submit.click();
        return PageFactory.initElements(driver,
                AllPostsPage.class);
    }
    @Override
    protected void load() {
        driver.get("http://pageobjectpattern.wordpress.com/wp-admin");
    }
    @Override
    protected void isLoaded() throws Error {
        Assert.assertTrue(driver.getCurrentUrl().contains("wp-admin"));
    }
}
// using it in a test
AdminLoginPageUsingLoadableComponent loginPage = new AdminLoginPageUsingLoadableCompon
// it will throw the exception if load doesn't work
```

Full Wordpress PageObject example with TestCases

PageObject's

- AdminLoginPage.java

```
package com.packt.webdriver.chapter9.pageObjects;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.CacheLookup;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
import org.openqa.selenium.support.PageFactory;
public class AdminLoginPage {
```

```
    WebDriver driver;
    @FindBy(how=How.ID, using="user_login")
    WebElement email;
    @FindBy(how=How.ID, using="user_pass")
    WebElement password;
    @FindBy(how=How.ID, using="wp-submit")
    WebElement submit;
    public AdminLoginPage(WebDriver driver){
        this.driver = driver;
        driver.get("http://pageobjectpattern.wordpress.com/wp-admin");
    }
    public AllPostsPage login(){
        email.sendKeys("pageobjectpattern@gmail.com");
        password.sendKeys("webdriver123");
        submit.click();
        return PageFactory.initElements(driver, AllPostsPage.class);
    }
}
```

- AllPostsPage.java

```
package com.packt.webdriver.chapter9.pageObjects;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Action;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
import org.openqa.selenium.support.PageFactory;
public class AllPostsPage {
    WebDriver driver;
    @FindBy(how=How.ID, using="the-list")
    WebElement postsContainer;
    @FindBy(how=How.ID, using="post-search-input")
    WebElement searchPosts;
    @FindBy(how=How.ID, using="cat")
    WebElement viewByCategories;
    @FindBy(how=How.LINK_TEXT, using="Add New")
```

```
WebElement addNewPost;
public AllPostsPage(WebDriver driver){
    this.driver = driver;
    driver.get("http://pageobjectpattern.wordpress.com/wp-admin/edit.php");
}
public void createANewPost(String title, String description){
    addNewPost.click();
    AddNewPostPage newPost = PageFactory.initElements(driver, AddNewPostPage.c
    newPost.addNewPost(title, description);
}
public void editAPost(String presentTitle, String newTitle, String descriptio
    goToParticularPostPage(presentTitle);
    EditPostPage editPost = PageFactory.initElements(driver, EditPostPage.clas
    editPost.editPost(newTitle, description);
}
public void deleteAPost(String title) {
    goToParticularPostPage(title);
    DeletePostPage deletePost = PageFactory.initElements(driver, DeletePostPag
    deletePost.delete();
}
public void filterPostsByCategory(String category){
}
public void searchInPosts(String searchText){
}
public int getAllPostsCount(){
  List<WebElement> postsList = postsContainer.findElements(By.tagName("tr"));
  return postsList.size();
}
private void goToParticularPostPage(String title){List<WebElement> allPosts= p
    for(WebElement ele : allPosts){
        if(ele.getText().equals(title)){
            Actions builder = new Actions(driver);
            builder.moveToElement(ele);
            builder.click(driver.findElement(
                    By.cssSelector(".edit>a")));
            // Generate the composite action.
            Action compositeAction = builder.build();
            // Perform the composite action.
            compositeAction.perform();
            break;
```

```
            }
        }
    }
}
```

- AddNewPostPage.java

```
package com.packt.webdriver.chapter9.pageObjects;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
public class EditPostPage {
WebDriver driver;
    @FindBy(how=How.ID, using="content_ifr")
    WebElement newPostContentFrame;
    @FindBy(how=How.ID, using="tinymce")
    WebElement newPostContentBody;
    @FindBy(how=How.ID, using="title")
    WebElement newPostTitle;
    @FindBy(how=How.ID, using="publish")
    WebElement newPostPublish;
    public EditPostPage(WebDriver driver){
        this.driver = driver;
        System.out.println(driver.getCurrentUrl());
    }
    public void editPost(String title, String descContent){
         driver.switchTo().frame(newPostContentFrame);
         newPostContentBody.clear();
         newPostContentBody.sendKeys(descContent);
         driver.switchTo().defaultContent();
         newPostTitle.click();
         newPostTitle.clear();
         newPostTitle.sendKeys(title);
         newPostPublish.click();
    }
}
```

- EditPostPage.java

```
package com.packt.webdriver.chapter9.pageObjects;
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
public class EditPostPage {
WebDriver driver;
    @FindBy(how=How.ID, using="content_ifr")
    WebElement newPostContentFrame;
    @FindBy(how=How.ID, using="tinymce")
    WebElement newPostContentBody;
    @FindBy(how=How.ID, using="title")
    WebElement newPostTitle;
    @FindBy(how=How.ID, using="publish")
    WebElement newPostPublish;
    public EditPostPage(WebDriver driver){
        this.driver = driver;
        System.out.println(driver.getCurrentUrl());
    }
    public void editPost(String title, String descContent){
         driver.switchTo().frame(newPostContentFrame);
         newPostContentBody.clear();
         newPostContentBody.sendKeys(descContent);
         driver.switchTo().defaultContent();
         newPostTitle.click();
         newPostTitle.clear();
         newPostTitle.sendKeys(title);
         newPostPublish.click();
    }
}
```

- DeletePostPage.java

```
package com.packt.webdriver.chapter9.pageObjects;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
public class DeletePostPage {
WebDriver driver;
    @FindBy(how=How.LINK_TEXT, using="Move to Trash")
```

```
    WebElement moveToTrash;
    public DeletePostPage(WebDriver driver){
        this.driver = driver;
        System.out.println(driver.getCurrentUrl());
    }
    public void delete(){
        moveToTrash.click();
    }
}
```

- Tests
  - Adding a new post

```
package com.packt.webdriver.chapter9;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.PageFactory;
import com.packt.webdriver.chapter9.pageObjects.AdminLoginPage;
import com.packt.webdriver.chapter9.pageObjects.AllPostsPage;
public class TestAddNewPostUsingPageObjects {
    public static void main(String... args){
        WebDriver driver = new FirefoxDriver();
        AdminLoginPage loginPage = PageFactory.initElements(driver, AdminLogi
        AllPostsPage allPostsPage = loginPage.login();
        allPostsPage.createANewPost("Creating New Post using PageObjects",
                "Its good to use PageObjects");
    }
}
```

  - Editing a post

```
package com.packt.webdriver.chapter9;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.PageFactory;
import com.packt.webdriver.chapter9.pageObjects.AdminLoginPage;
import com.packt.webdriver.chapter9.pageObjects.AllPostsPage;
public class TestEditPostUsingPageObjects {
    public static void main(String... args){
        System.setProperty("webdriver.chrome.driver", "C:\\chromedriver_win32
        WebDriver driver = new ChromeDriver();
```

```
        AdminLoginPage loginPage = PageFactory.initElements(driver,AdminLogin
        AllPostsPage allPostsPage = loginPage.login();
        allPostsPage.editAPost("Creating New Post using PageObjects",
                "Editing Post using PageObjects","Test framework low maintena
    }
}
```

– Deleting a post

```
package com.packt.webdriver.chapter9;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.PageFactory;
import com.packt.webdriver.chapter9.pageObjects.AdminLoginPage;
import com.packt.webdriver.chapter9.pageObjects.AllPostsPage;
public class TestDeleteAPostUsingPageObjects {
    public static void main(String... args){
        System.setProperty("webdriver.chrome.driver", "C:\\chromedriver_win32
        WebDriver driver = new ChromeDriver();
        AdminLoginPage loginPage = PageFactory.initElements(driver,AdminLogin
        AllPostsPage allPostsPage = loginPage.login();
        allPostsPage.deleteAPost("Creating New Post using PageObjects");
    }
}
```

– Counting posts

```
package com.packt.webdriver.chapter9;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.PageFactory;
import com.packt.webdriver.chapter9.pageObjects.AdminLoginPage;
import com.packt.webdriver.chapter9.pageObjects.AllPostsPage;
public class TestPostsCountUsingPageObjects {
    public static void main(String... args){
        System.setProperty("webdriver.chrome.driver", "C:\\chromedriver_win32
        WebDriver driver = new ChromeDriver();
        AdminLoginPage loginPage = PageFactory.initElements(driver,AdminLogin
        AllPostsPage allPostsPage = loginPage.login();
        System.out.println(allPostsPage.getAllPostsCount());
    }
}
```