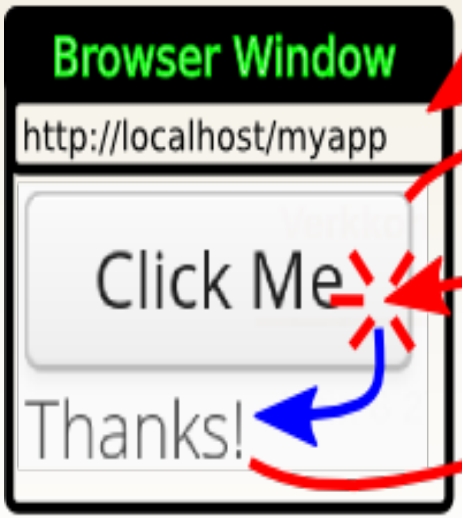
Vaadin TestBench

Vaadin TestBench

- Java kodu üzerinden **tarayıcıyı programatik olarak kontrol etme**yi sağlar
- Tarayıcı penceresi açılabilir, uygulamaya giriş yapılabilir, UI bileşenleri ile etkileşim gerçekleştirilebilir, dönen HTML elemanları incelenebilir
- Yazılım geliştirme sürecinin hemen her evresinde **test otomasyonu** amacı ile kullanılabilir

Vaadin TestBench




```
@Test
public void testButton() throws Exception {
    getDriver().get("http://localhost/myapp");

    ButtonElement button = $(ButtonElement)
      .caption("Click Me").first();
    button.click();

    LabelElement label = $(LabelElement).first();
    assertEquals("Thanks!", label.getText());
}
```

Genel Özellikleri

- Java kodundan **tarayıcı kontrolü**
- **UI state**'inin **assertion**'larla denetlenmesi
- **Screen capture** karşılaştırması yapılması
- Debug penceresinden UI assertion'ları için **selector**'ların üretilmesi
- Testlerin **dağıtık ortam**da paralel biçimde çalıştırılması
- Testlerin headless modda çalıştırılması

TestBench Mimarisi

- **Selenium** kütüphanesi üzerine kuruludur
- Özellikle **WebDriver** ön plandadır
- WebDriver ile tarayıcının **Java kodu**ndan kontrolü sağlanır
- Selenium üzerinde Vaadin'e özel **extension**'lar geliştirilmiştir
- WebDriver ile çalışabilmek için bazı **tarayıcılara özel sürücü** kurulumu gerekebilir

Lisans ve Kullanım

- **Ücretlidir**, 30 günlük deneme süresi vardır
- **Vaadin Directory**'den indirilerek kullanılabilir
- Yine aynı yerden ücretsiz **deneme lisansı** alınabilir
- Vaadin **Pro Account** satın alındığı takdirde de TestBench lisansı verilmektedir
- Alınan lisans JVM **sistem property**'si olarak set edilir
  - -Dvaadin.testbench.developer.license=<*license-key*>

JUnit ile Kullanımı

- Vaadin TestBench ve WebDriver **herhangi bir test framework** ile kullanılabilir
- Hatta testleri geliştirmek için **sıradan Java uygulaması** da yazılabilir
- Yaygın kullanım örnekleri **JUnit** üzerindendir

Bir Testin Anatomisi

UI testlerinin extend etmesi gereken ana sınıf **TestBenchTestCase**'dir

Herhangi bir test başarısız olduğunda screenshot alınmasını sağlayan bir Junit **@Rule** tanımıdır

```
public class SampleTestCase extends TestBenchTestCase {

    @Rule
    public ScreenshotOnFailureRule screenshotOnFailureRule
            = new ScreenshotOnFailureRule(this, true);


    @Before
    public void setUp() throws Exception {
        setDriver(new FirefoxDriver());
    }
```

Screenshot projede **error-screenshots** altında üretilir

Driver sınıfı tarayıcıya göre değişir. FirefoxDriver, ChromeDriver, InternetExplorerDriver, SafariDriver, yada PhantomJSDriver olabilir.

Bir Testin Anatomisi

Belirtilen URL'e erişim sağlayarak tarayıcı penceresinin açılmasını sağlar

```
@Test
public void testClickButton() throws Exception {

    getDriver().get("http://localhost/myapp");

    ButtonElement button =
        $(ButtonElement.class).caption("Click Me!").first();

    button.click();

    LabelElement label = $(LabelElement.class).first();

    Assert.assertEquals("Thanks!", label.getText());

    Assert.assertTrue(testBench().compareScreen("testClickButton"));
}
```

Sayfa içeriğinde belirtilen tipte Vaadin bileşenlerini sorgulamayı sağlar

Projenin **reference-screenshots** dizini altındaki belirtilen referans id'li imajı test sonucu oluşan ekran görüntüsü ile karşılaştırır. Fark olup olmamasına göre true/false döner

Bir Testin Anatomisi

Test sonucu sürücü'den çıkış yapılması tarayıcının kapanmasını sağlar

```
    @After
    public void tearDown() throws Exception {

        driver.quit();

    }
}
```

Eğer ScreenshotOnFailureRule tanımlı ise çağrılMAmalıdır. ScreenshotOnFailureRule kendisi implicit biçimde quit yapacaktır.

Element Query API

- TestBench Vaadin'e özel **Element Query API** sunar
- Bileşenler **tip**lerine, **hiyerarşi**deki yerlerine, id, caption değerlerine vb. göre erişilebilir
- API'de Vaadin'deki **her bileşene karşılık** gelen bir **Element** sınıfı vardır
- Tarayıcı üzerinde herhangi bir bileşen seçildiğinde **Debug penceresinde** bu bileşen için bir query otomatik olarak üretilebilir

Element Query API

$() ile belirtilen Element tipindeki bütün bileşenler sorgulanabilir
Sorgu sayfa içerisinde hiyerarşideki ilgili bütün bileşenleri döner.

```
List<ButtonElement> buttons = $(ButtonElement.class).all();

for (ButtonElement button : buttons) {
    button.click();
}

LabelElement label = $(LabelElement.class).caption("Hello
World!").first();

label.showTooltip();
label.doubleClick();

String html = $(PanelElement.class).id("myPanel")
        .$$(TextFieldElement.class).get(2).getHTML();
```

$$() ise belirtilen tipteki elemanı search context'in içerisinde onun doğrudan çocuk elemanları arasında arar. Hiyerarşide alt elemanlara devam etmez.

Element Selector

- TestBench ayrıca **Selenium WebDriver API** üzerinden çalışan Vaadin extension'larına da sahiptir
- Sayfa içerisindeki bileşenlere **XPath** ifadesi, html **element id**'si, **CSS** style class'ı gibi bilgilerle erişmeyi sağlar
- Mevcut selector'ler **com.vaadin.testbench.By** üzerinden statik metotlarla erişilebilir

Element Selector

```
findElement(By.id("myButton")).click();

findElement(By.className("v-button")).getText();

findElement(By.vaadin(
"petclinicapp::/VVerticalLayout[0]/VButton[0]")).isSelected();

findElement(By.xpath("//*[@id=\"myButton\"]")).clear();

findElement(By.tagName("button")).getLocation();

findElement(By.linkText("Click Me!")).isDisplayed();
```

Vaadin'e Özel Bekleme

- Selenium **klasik web uygulamaları** için tasarlanmıştır
- Her bir request'e ait response **senkron** olarak üretilmektedir
- AJAX tabanlı uygulamalarda ise response **asenkron** olarak elde edilmektedir
- Dolayısı ile WebDriver'in cevabın dönmesi için **bir süre bekleme**si gerekir

Vaadin'e Özel Bekleme

- Vaadin TestBench, client side engine ile birlikte çalıştığı için bekleme işini kendisi ele almaktadır
- Ancak bazı durumlarda bekleme özelliğini kapatmak ve manuel bekleme yapmak gerekebilir

```
testBench().disableWaitForVaadin();

testBench().waitForVaadin();

testBench().enableWaitForVaadin();
```

Tooltip'lerin Test Edilmesi

- Tooltip mesajların sayfada görüntülenmesi için farenin ilgili bileşen üzerine gelmesi gerekir
- TestBench ile bunu programatik yapmak için **showToolTip()** metodu mevcuttur

```
ButtonElement button = $(ButtonElement.class).caption("Click").first();

button.showTooltip();

String tooltip = findElement(By.className("v-tooltip")).getText();

Assert.assertEquals("This is tooltip message", tooltip);
```

Açılan tooltip, HTML sayfa içerisinde özel bir overlay elemanı içerisinde floating point olarak görüntülendiği için sadece classname selector ile erişilebilir

Programatik Scroll

- Table, Panel gibi Vaadin bileşenlerinde zaman zaman manuel biçimde scroll yapmak gerekebilir
- Bunun için ilgili bileşen elemanına erişip **scroll**() ve **scrollLeft**() metotları kullanılabilir

```
PanelElement panel = $(PanelElement.class).first();

panel.scroll(200);

panel.scrollLeft(300);
```

Scroll metotlarına verilen input arg pixel değeridir

Notifikasyonların Test Edilmesi

Notification'ın görüntülenmesi tetiklenir

NotificationElement ile o anda Sayfada beliren notification bileşenin elemanına erişilebilir

```
ButtonElement button = $(ButtonElement.class)
                       .caption("Show Message").first();

button.click();

NotificationElement notification = $(NotificationElement.class).first();

Assert.assertEquals("The caption", notification.getCaption());

Assert.assertEquals("The description", notification.getDescription());

Assert.assertEquals("warning", notification.getType());

notification.close();
```

Notification'a ait caption, description ve tip bilgileri erişilip sınanabilir

İşlem sonunda notification'ın kapatılması gerekmektedir

Context Menu'lerin Test Edilmesi

Öncelikle context menu'ye sahip bileşene erişilir

```
TableElement table = $$(TableElement.class).first();

WebElement cell = table.getCell(3, 0);

new Actions(getDriver()).contextClick(cell).perform();

WebElement menu = findElement(By.className("v-contextmenu"));

WebElement menuitem = menu
        .findElement(By.xpath("//*[text() = 'Add Comment']"));

menuitem.click();
```

Daha sonra Selenium'a ait **Actions** nesnesi ile ilgili bileşen üzerinde context menu açılabilir

Açılan context menu HTML sayfa içerisinde özel bir overlay elemanı altında floating point olduğu için classname selector ile erişilebilir

Page Object Pattern

- UI testlerini daha **basit**, **modüler** ve **bakımı kolay** biçimde oluşturmak için kullanılır
- Test'i yapılacak view'a ait bir **Page Object** sınıfı yaratılır
- Sayfaya özel elemanlara erişim, UI etkileşimleri, kontroller vs bu sınıf tarafından **encapsule** edilir
- Daha sonra UI test metodu içerisinde Page Object oluşturularak kullanılır

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com