We are GLOBANT

Empowering organizations **for a digital and cognitive revolution**

Selenium Webdriver

La innovación es lo que distingue al Líder de sus seguidores.

Steve Jobs

Carmelo Buelvas Comas

Test Automation Engineer

Globant

Agenda

Temas a tratar:

- Introducción
- Selenium
- Usando herramientas en los navegadores para inspeccionar elementos.
- Firebug
- Selenium WebDriver
- Arquitectura Selenium WebDriver
- Localizando elementos utilizando los métodos findElement y findElements
- Page Object Model
- Page Factory

Introducción

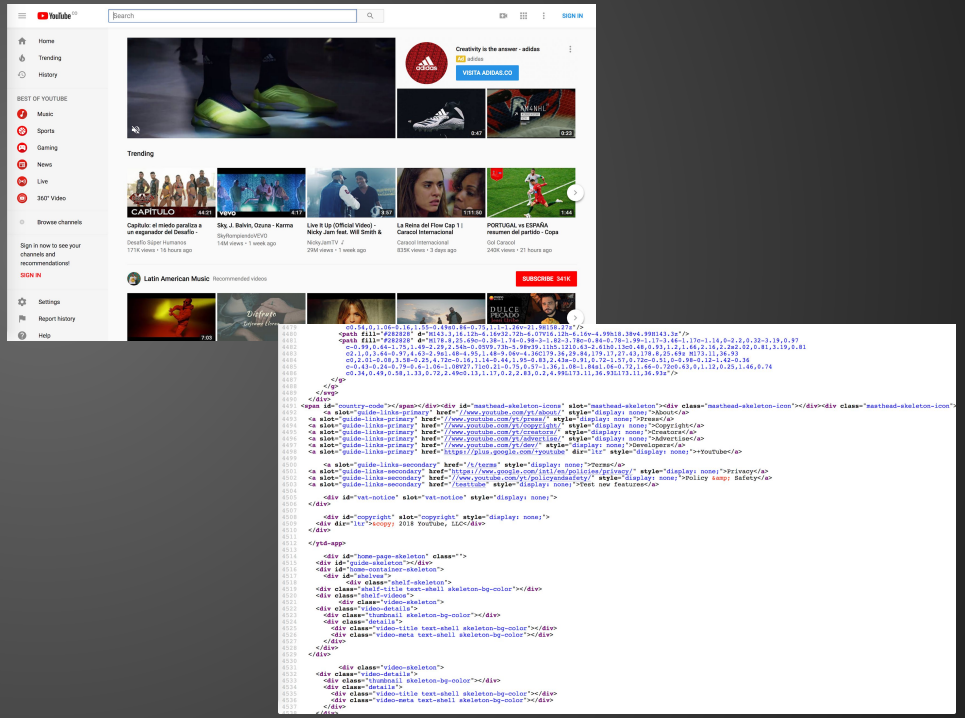
Las pruebas automáticas de aplicaciones web se realizan en la interfaz de usuario (UI - User Interface) y depende de la identificación y localización de los elementos de la UI de la app web bajo prueba, luego de la realización de operaciones y verificaciones en estos elementos para lograr éxito de la prueba. Esto se reduce a la capacidad de la herramienta de prueba para reconocer varios elementos de la GUI de manera efectiva y eficaz.

Test Automation Pyramid

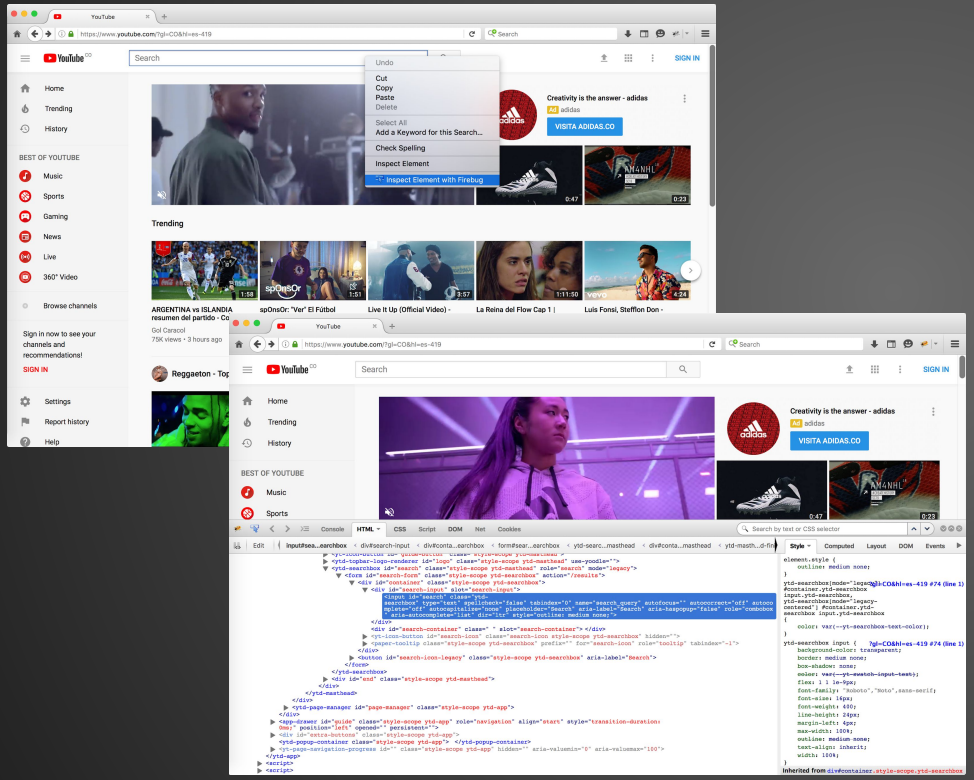
Usando herramientas en los navegadores para inspeccionar elementos.

Los navegadores web representan los elementos visuales de la aplicación a fin de que los usuarios finales los puedan apreciar, por otro lado ocultan el código HTML y otros recursos.

Al pretender automatizar la interacción con aplicaciones web a través de Selenium WebDriver, debemos analizar el código HTML de fondo. Es Necesario identificar información como los valores de los atributos y la estructura de los elementos, esto para ubicar los elementos y realizar acciones utilizando la API Selenium WebDriver.



Firebug

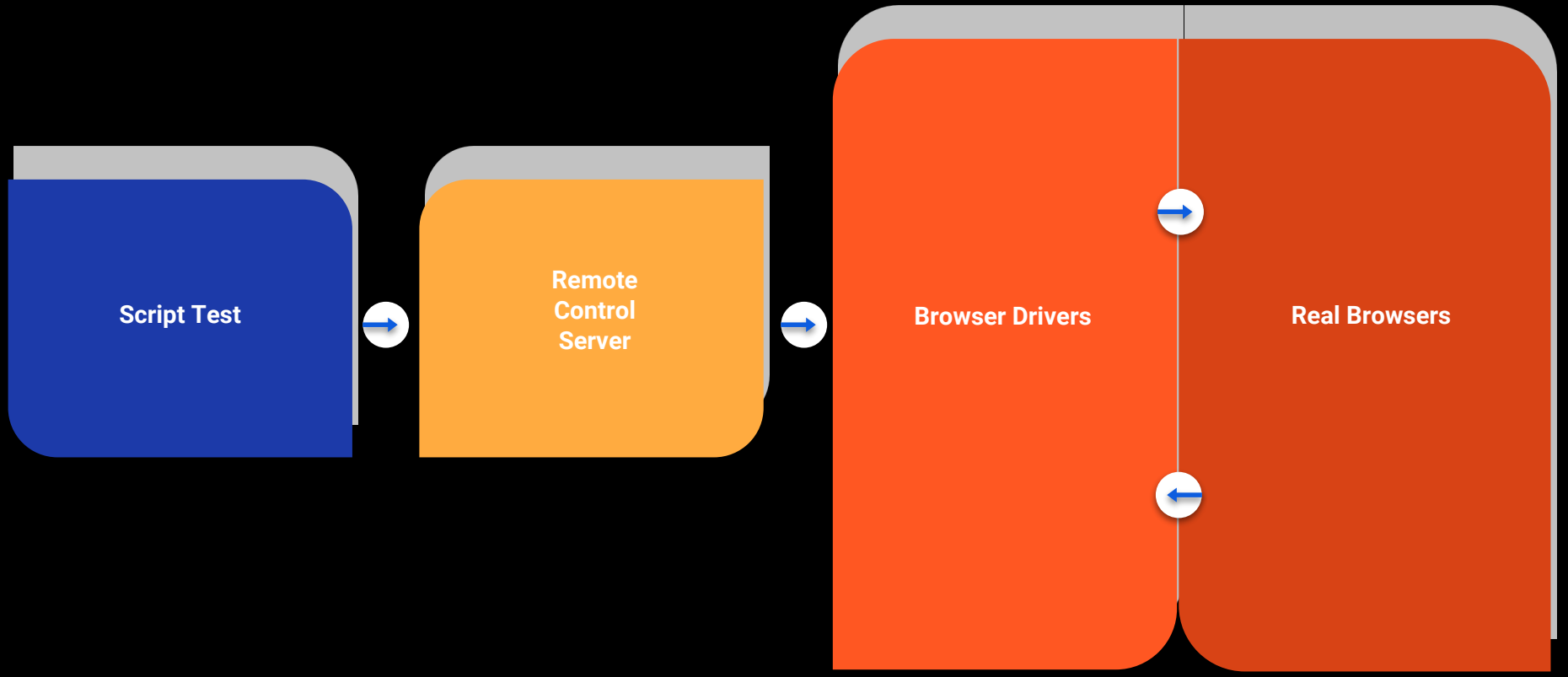


El complemento Firebug es una herramienta que se utiliza sobre Firefox, actualmente está disponible para su uso hasta la versión 46.0 de este navegador. Mediante su uso podemos localizar los elementos sobre los cuales queremos interactuar de forma más sencilla.

Selenium Webdriver

Selenium como herramienta para automatización de pruebas, actúa sobre la UI y proporciona una de las técnicas avanzadas y estables para localizar elementos en páginas web. La API de Selenium provee múltiples estrategias de localización como nombre, ID, CssSelector, XPath, entre otras estrategias. También podemos implementar estrategias personalizadas de localización para obtener elementos.

Cómo funciona y se integra Selenium Webdriver



Localizando elementos utilizando los métodos findElement y findElements

Selenium WebDriver provee varias estrategias para la localización de elementos en este apartados abordaremos los métodos ***findElement()*** y ***findElements()*** proporcionados por la API de WebDriver y la clase WebElement.

- El método ***findElement ()*** devuelve un objeto de tipo WebElement en función de un criterio de búsqueda específico, en caso de no encontrar valor alguno según el criterio de búsqueda lanza una excepción.

```
WebElement username = driver.findElement(By.id("username"));
```

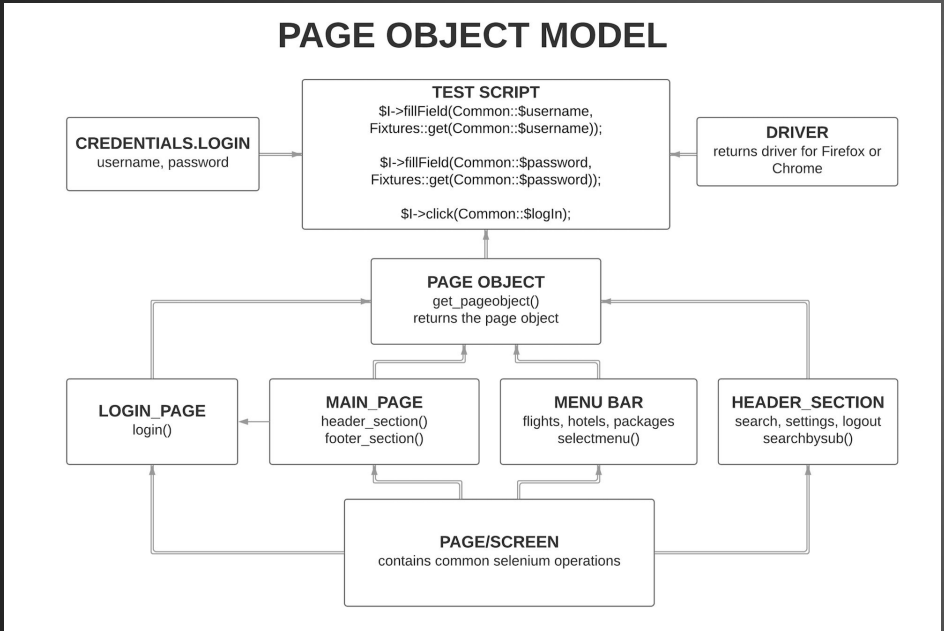
- El método ***findElements ()*** devuelve una lista de objetos tipo WebElement que coinciden con los criterios de búsqueda especificados. Si no se encuentran elementos, devuelve una lista vacía.

```
List<WebElement> links = driver.findElements(By.tagName("a"));
```

Estrategia	Sintaxis	Descripción
By Tag Name	driver.findElement(By. tagName(...))	Estrategia de localización que ubica un elemento usando el nombre de la etiqueta HTML
By link text	driver.findElement(By. linkText(...))	Estrategia de localización que ubica el enlace usando su texto
By partial link text	driver.findElement(By. partialLinkText(...))	Estrategia de localización que ubica el enlace usando su texto parcial
By CSS	driver.findElement(By. cssSelector(...))	Estrategia de localización que ubica el elemento usando el selector de CSS
By XPath	driver.findElement(By.xpath (...))	Estrategia de localización que ubica el elemento usando la consulta XPath

Selenium WebDriver

Page Object Model



Page Object Model - POM, es un patrón de diseño para la automatización de pruebas que se ha hecho popular, este nos permite mejorar el mantenimiento de las pruebas y reducir la duplicación de código. Una página se traduce a una clase - objetos que sirve para representar los elementos de la misma y definir algún tipo de comportamiento.

El patrón de diseño fue propuesto por *Martin Flower* en el año 2013.

Page Factory

PageFactory en Selenium es una extensión del patrón de diseño Page Object Model. Se utiliza para la inicialización de los elementos representados en el objeto de página.

Todos los elementos en el caso de Java anotados con @FindBy son susceptibles de ser inicializados por el Page Factory en tiempo de ejecución.

Globant
we are ready

Tecnologías que soportan Selenium

Java

TestNG

Maven™

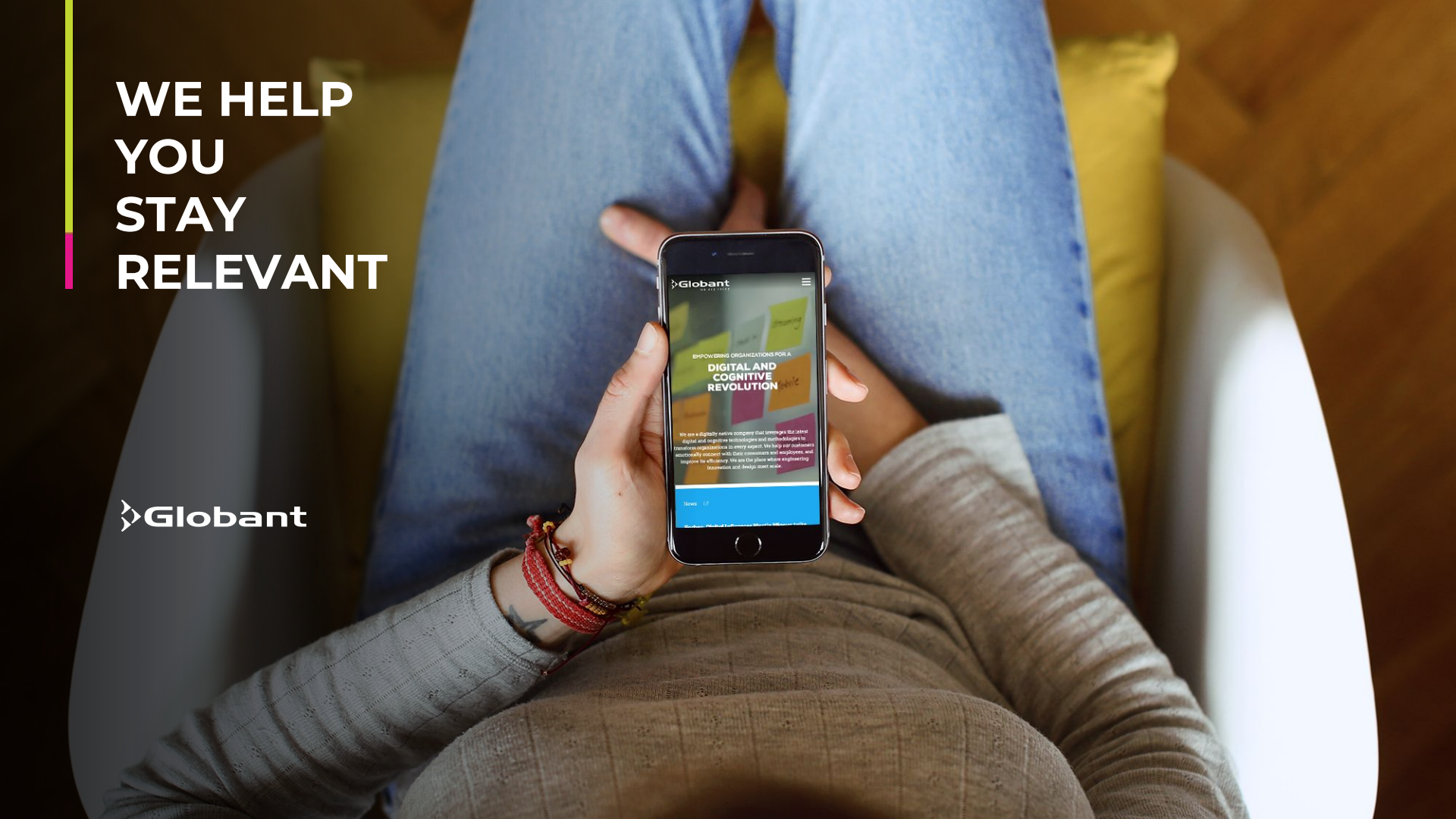
Se

AssertJ
Fluent assertions for java

Fuentes Bibliográficas

- https://medium.com/tech-tajawal/page-object-model-pom-design-pattern-f9588630800b
- https://martinfowler.com/bliki/PageObject.html
- https://www.seleniumhq.org/docs/05_selenium_rc.jsp

WE HELP YOU STAY RELEVANT



Globant

THANKS!