



QA & Testing

Junior Academy

160 horas

Módulo 2

Introducción a la Automatización de Pruebas

80 horas aproximadamente

2.2 Introducción automatizar pruebas para aplicaciones web

Jonatan Villén / Rubén García

Introducción a Selenium

- Arquitectura y comandos Selenium WebDriver
- Introducción a XPath y CSS Selector
- Comandos WebDriver
- Switches Alerts and Windows
- Action Class
- JUnit en Selenium.

Buenas Prácticas

- Patrón Page Object Model
- Reusabilidad



Buenas Prácticas

Patrones de diseño

¿Qué es un patrón?

- Un patrón es una técnica para resolver problemas comunes en el desarrollo del software.
- Solución a un problema de diseño, ¿Beneficios?

Buenas Prácticas

Patrones de diseño

¿Qué es un patrón?

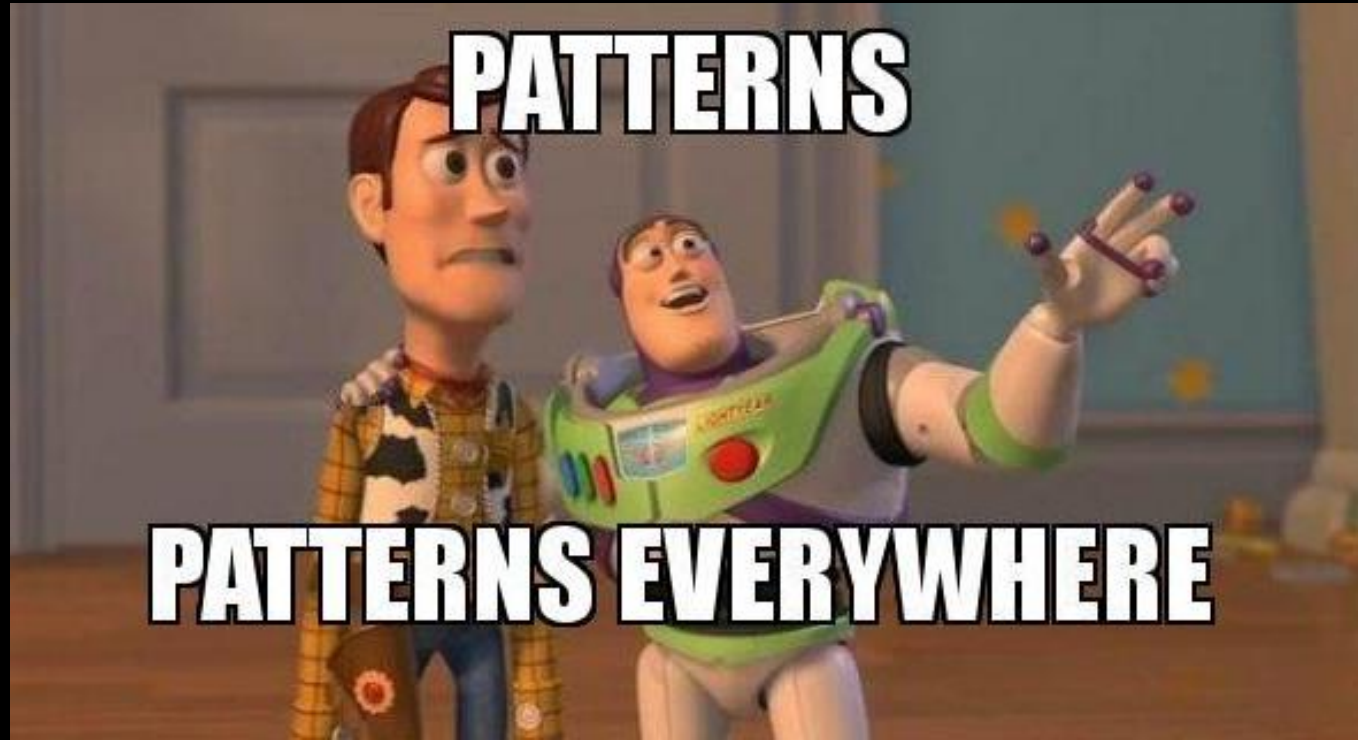
- Un patrón es una técnica para resolver problemas comunes en el desarrollo del software.
- Solución a un problema de diseño, ¿Beneficios?
 - Solución común
 - Problemas resueltos anteriormente
 - Reutilizable
 - Formalizar un “vocabulario común”
 - Facilitar aprendizaje
 - Estandarizar
- Asimismo, no pretenden:
 - Imponer ciertas alternativas de diseño frente a otras.
 - Eliminar la creatividad inherente al proceso de diseño.

Buenas Prácticas

Patrones de diseño

Ejemplos:

- Patrones estructurales:
 - Adapter o Wrapper
 - Bridge
 - Composite
 - Facade
 - ...
- Patrones de comportamiento
 - Observer
 - State
 - Strategy
 - ...



Buenas Prácticas

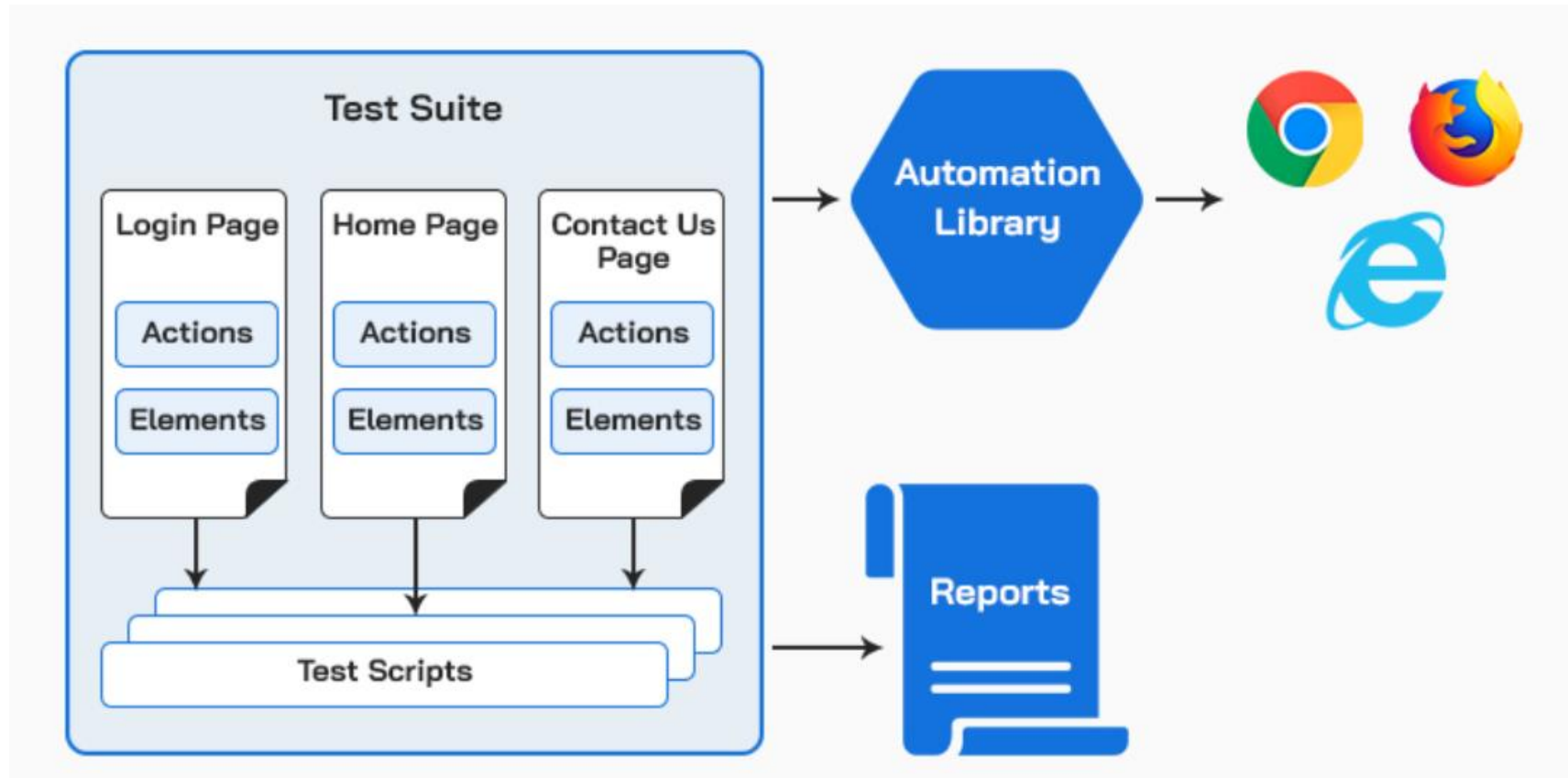
Patrón Page Object Model

Patrón Page Object Model (POM)

- Patrón de diseño usado en la automatización de pruebas
- Genera un repositorio de objetos con los elementos y acciones de cada página web
- Reduce la duplicación de código (código reusable)
- Mejora el mantenimiento de las pruebas
- Robustez en las pruebas

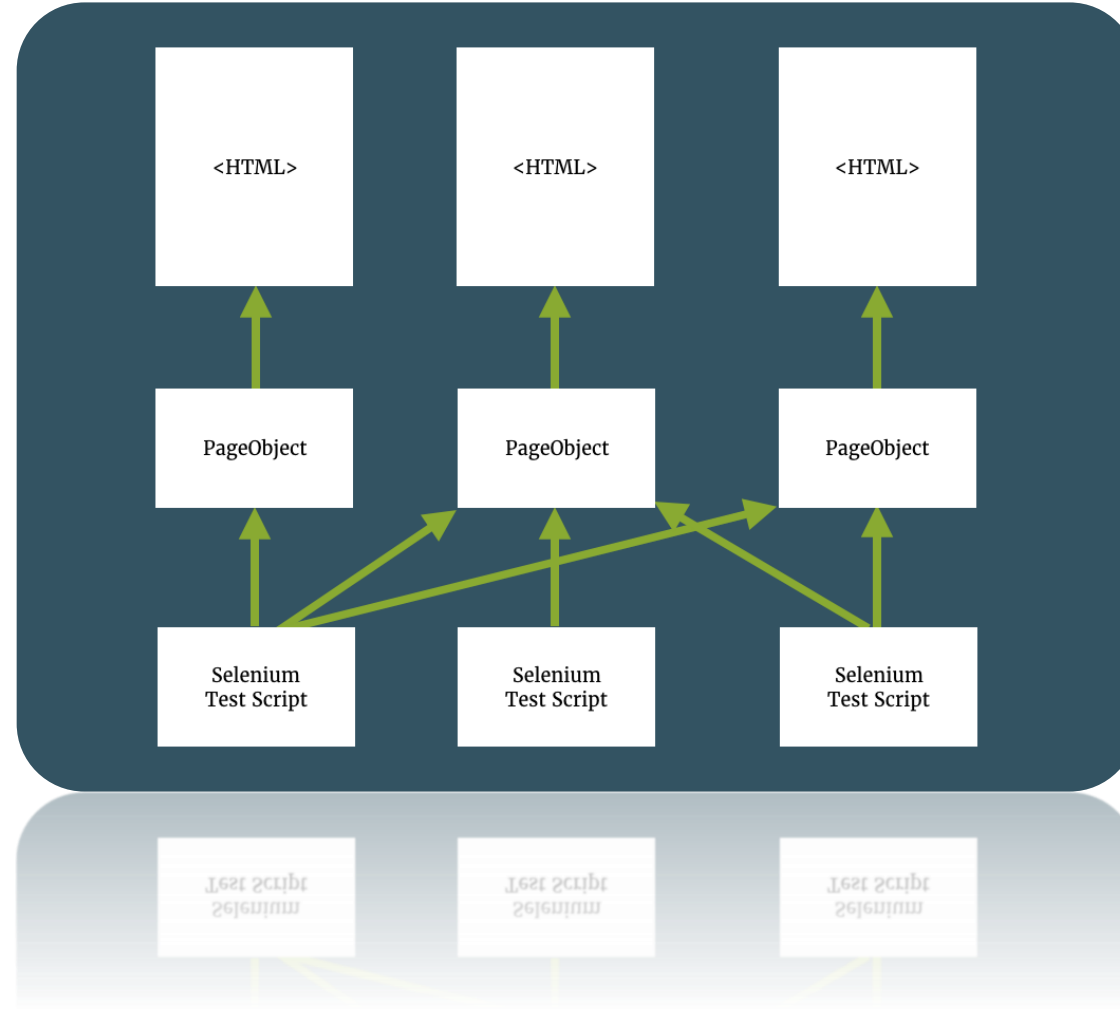
Buenas Prácticas

Patrón Page Object Model



Buenas Prácticas

Patrón Page Object Model



Buenas Prácticas

Patrón Page Object Model

Patrón Page Factory

- Se utiliza con el patrón Page Object Model
- Mayor mantenibilidad de código
- Código uniforme y estandarizado
- Gestión del repositorio de objetos
- Permite gestionar todas la páginas del patrón page object model
 - Inicializar las páginas y sus elementos de forma única
 - Instanciar las páginas
 - Acceder a componentes
 - Utilizar métodos de las páginas
 - Uso de la anotación **@FindBy** (lo veremos más adelante)

¿Comenzamos?

<https://demoqa.com/books>

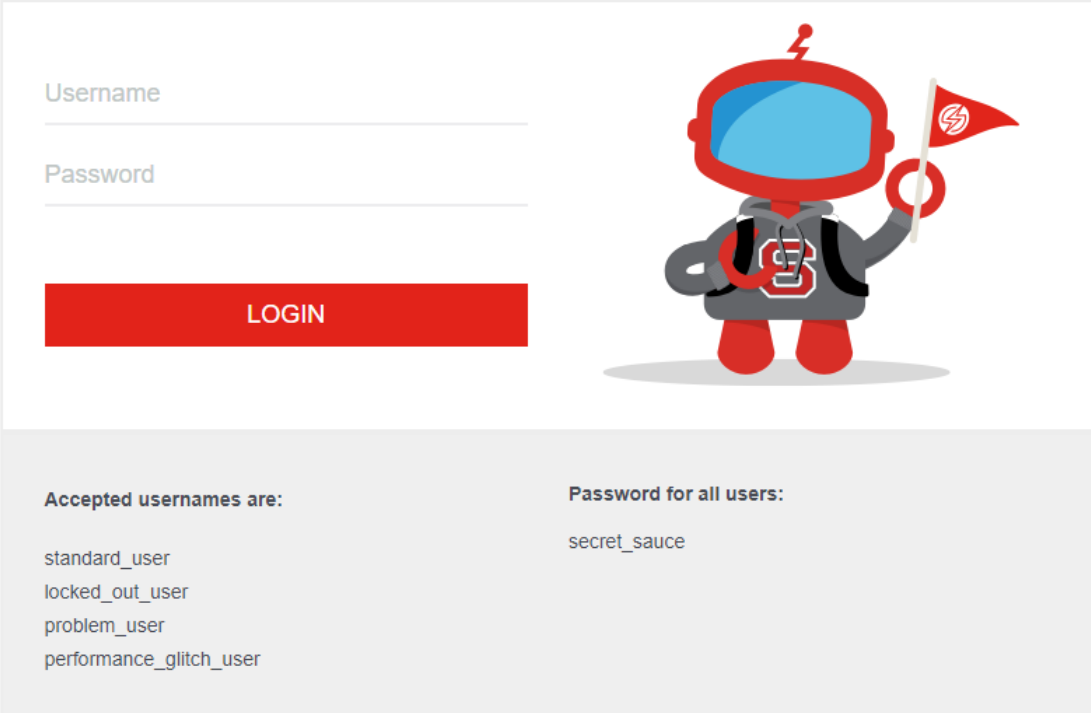
<https://www.saucedemo.com/>

Buenas Prácticas

Patrón Page Object Model – Ejercicio 1

Ejemplo “Login”

- Elementos
 -
 -
 -
- Acciones
 -



A login form for SWAGLABS. It features a red robot character on the right holding a flag. The form has two input fields: "Username" and "Password". Below the fields is a red "LOGIN" button. At the bottom, there is a grey box containing the accepted usernames and the password for all users.

Username

Password

LOGIN

Accepted usernames are:

- standard_user
- locked_out_user
- problem_user
- performance_glitch_user

Password for all users:

secret_sauce

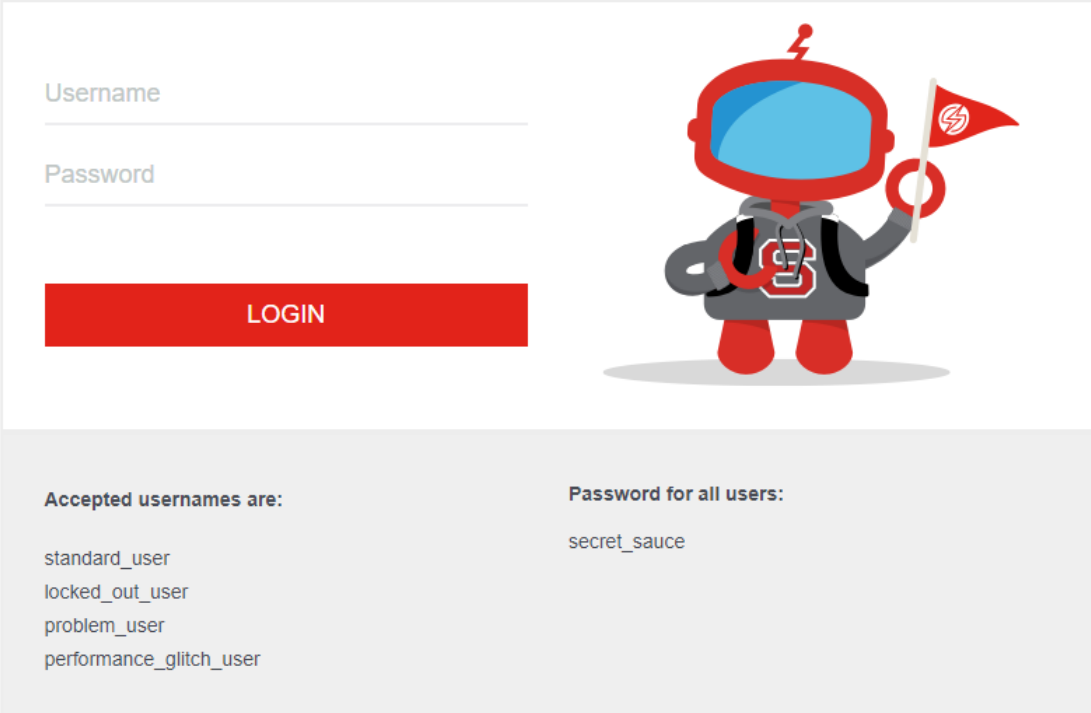
Buenas Prácticas

Patrón Page Object Model – Ejercicio 1



Ejemplo “Login”

- Elementos
 - Introducir usuario: **Username**
 - Introducir contraseña: **Password**
 - Botón de acceso: “**Login**”
- Acciones
 - Hacer login



A login form for SWAGLABS. It features a red robot character on the right holding a flag. The form has two input fields: "Username" and "Password". Below the fields is a red "LOGIN" button. At the bottom, there is a section with accepted usernames and a password for all users.

Username

Password

LOGIN

Accepted usernames are:

- standard_user
- locked_out_user
- problem_user
- performance_glitch_user

Password for all users:

secret_sauce

Buenas Prácticas

Patrón Page Object Model

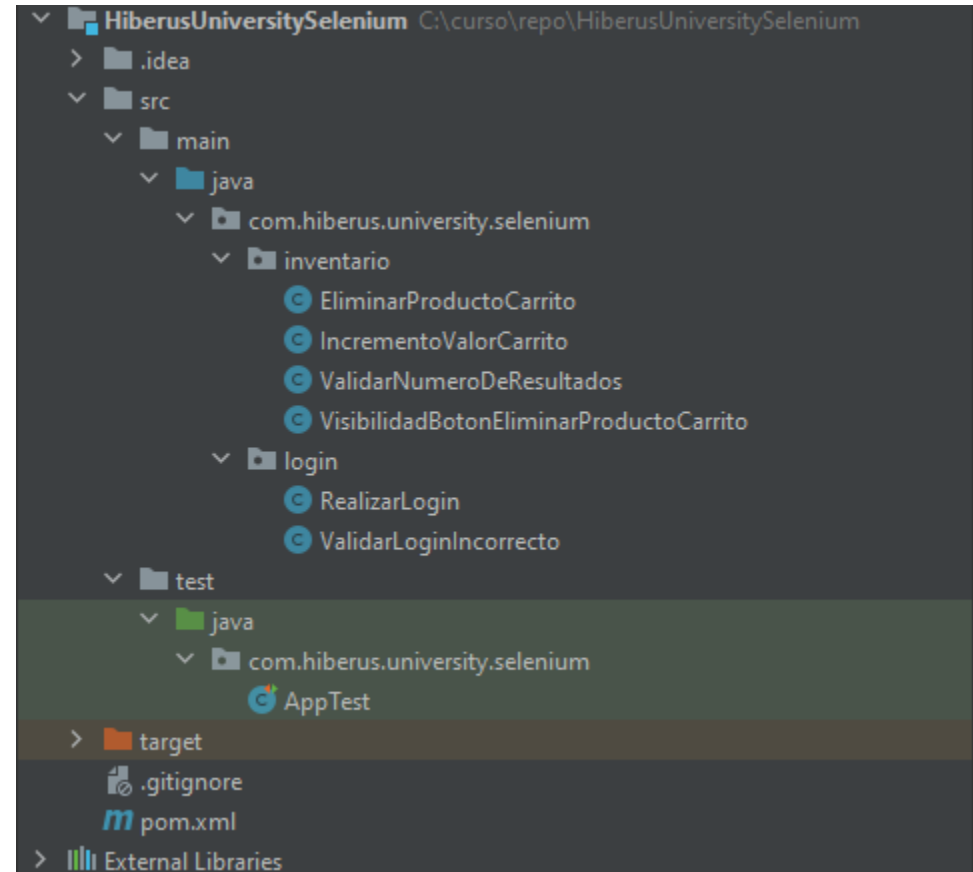
Del Proyecto actual

- **Paso 1: Definir las páginas y sus elementos**
- Paso 2: Implementar una estructura común
- Paso 3: Implementar cada página con sus elementos y métodos

Ejercicio 1 (1 hora)

Definir todas las páginas de
<https://www.saucedemo.com/>

con sus elementos y acciones posibles



Patrón Page Object Model – Ejercicio 1

- **CartPage**

- Elementos:
 - checkoutButton
 - removeButton
 - continueShoppingButton
 - openMenu
 - itemsList
- Métodos
 - clickCheckout()
 - getItemCount()
 - clickContinueShopping()

- **CheckOutStepOnePage**

- Elementos:
 - firstNameInput
 - lastNameInput
 - postalCodeInput
 - ContinueButton
- Métodos
 - enterFirstName()
 - enterLastName()
 - enterPostalCode()
 - clickContinue()

- **CheckOutStepSecondPage**

- Elementos:
 - itemTotalElement
 - taxElement
 - totalElement
- Métodos
 - getItemTotal()
 - getItemTotal()
 - getTax()
 - getTotal()

Buenas Prácticas

Patrón Page Object Model – Ejercicio 1

- **InventoryPage**

- Elementos:
 - openMenu
 - shoppingCartElement
 - inventoryContainerElement
 - productSortContainerSelect
- Métodos
 - addItemToCartByName()
 - removeItemByName()
 - clickOnShoppingCart()
 - sortInventoryByNameAsc()
 - sortInventoryByNameDesc()
 - sortInventoryByPriceAsc()
 - sortInventoryByPriceDesc()

- **InventoryItemPage**

- Elementos:
 - openMenu
 - itemName
 - itemDescription
 - addToCartButton
 - removeButton
 - itemPrice
 - shoppingCartElement
 - backToProductsButton
- Métodos:
 - addToCart()
 - removeToCart()
 - checkImage()

- **LoginPage**

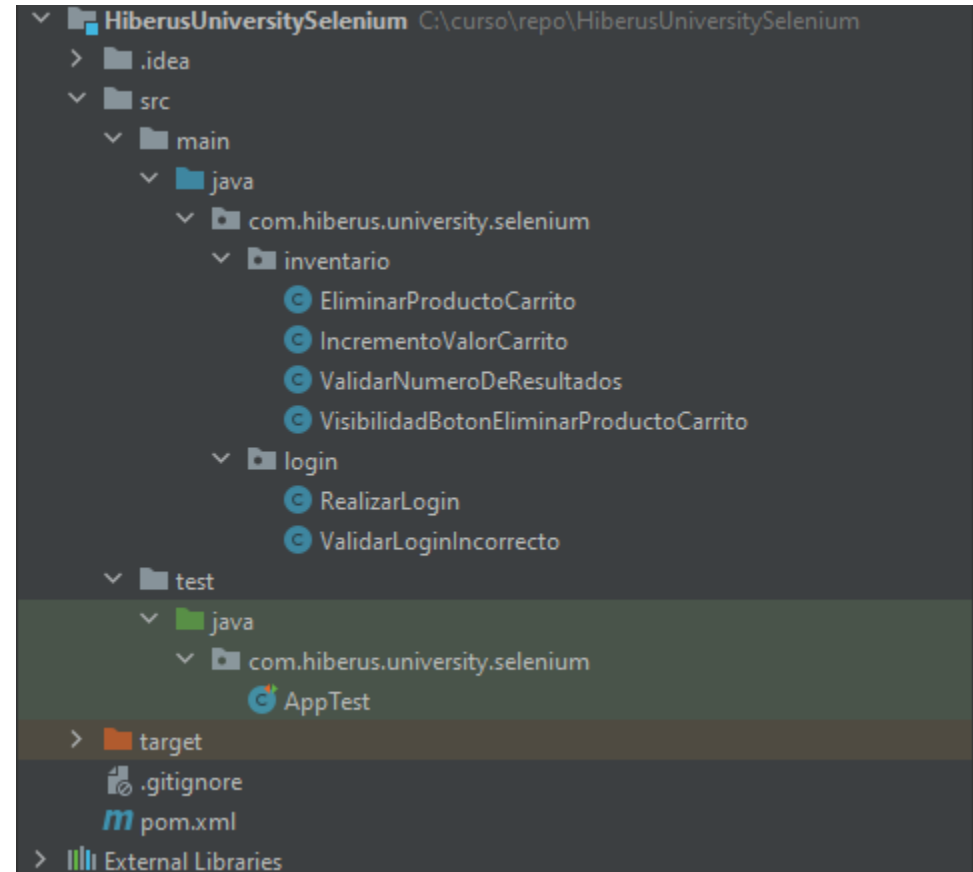
- Elementos:
 - usernameInput
 - passwordInput
 - loginButton
- Métodos:
 - clickLogin()
 - enterUsername()
 - enterPassword()
 - hasLockedOutError()
 - hasErrorLogin()

Buenas Prácticas

Patrón Page Object Model

Del Proyecto actual

- Paso 1: Definir las páginas y sus elementos
- **Paso 2: Implementar una estructura común**
- Paso 3: Implementar cada página con sus elementos y métodos

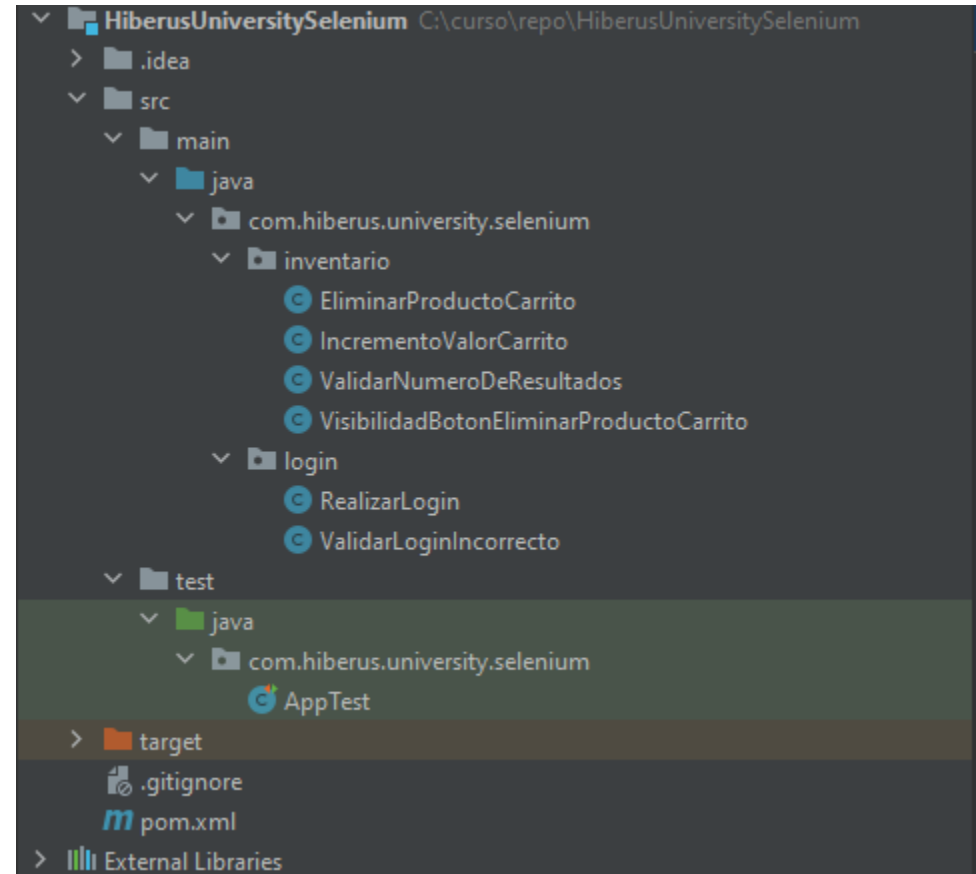


Buenas Prácticas

Patrón Page Object Model – Paso 2

Paso 2:

- Refactorizar proyecto actual
- Crear nuevos paquetes
- Crear nuevas clases



Buenas Prácticas

Patrón Page Object Model – Ejercicio 2

- Creación de los paquetes necesarios:
 - model
 - pages
 - stepdefs
 - utils
- Creación clase InventoryItem:
 - Nombre del item
 - Descripción del item
 - Precio del item

The image shows a screenshot of an IDE with two windows. The left window displays the project structure of 'HiberusUniversitySelenium'. The 'src/main/java' directory is expanded, showing the 'com.hiberus.university.selenium' package. Inside this package, the 'model' directory is highlighted, and the 'InventoryItem' class is selected. The right window shows the code for 'InventoryItem.java'. The code is organized into three sections, each highlighted with a red box and a label: 'Atributos' (Attributes) for the private fields, 'Constructores' (Constructors) for the two public constructors, and 'Getters y Setters' (Getters and Setters) for the public methods that access the fields.

```
package com.hiberus.university.selenium.model;

public class InventoryItem {

    private String name;
    private String description;
    private String price;

    public InventoryItem(String name, String description, String price) {
        this.name = name;
        this.description = description;
        this.price = price;
    }

    public InventoryItem(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

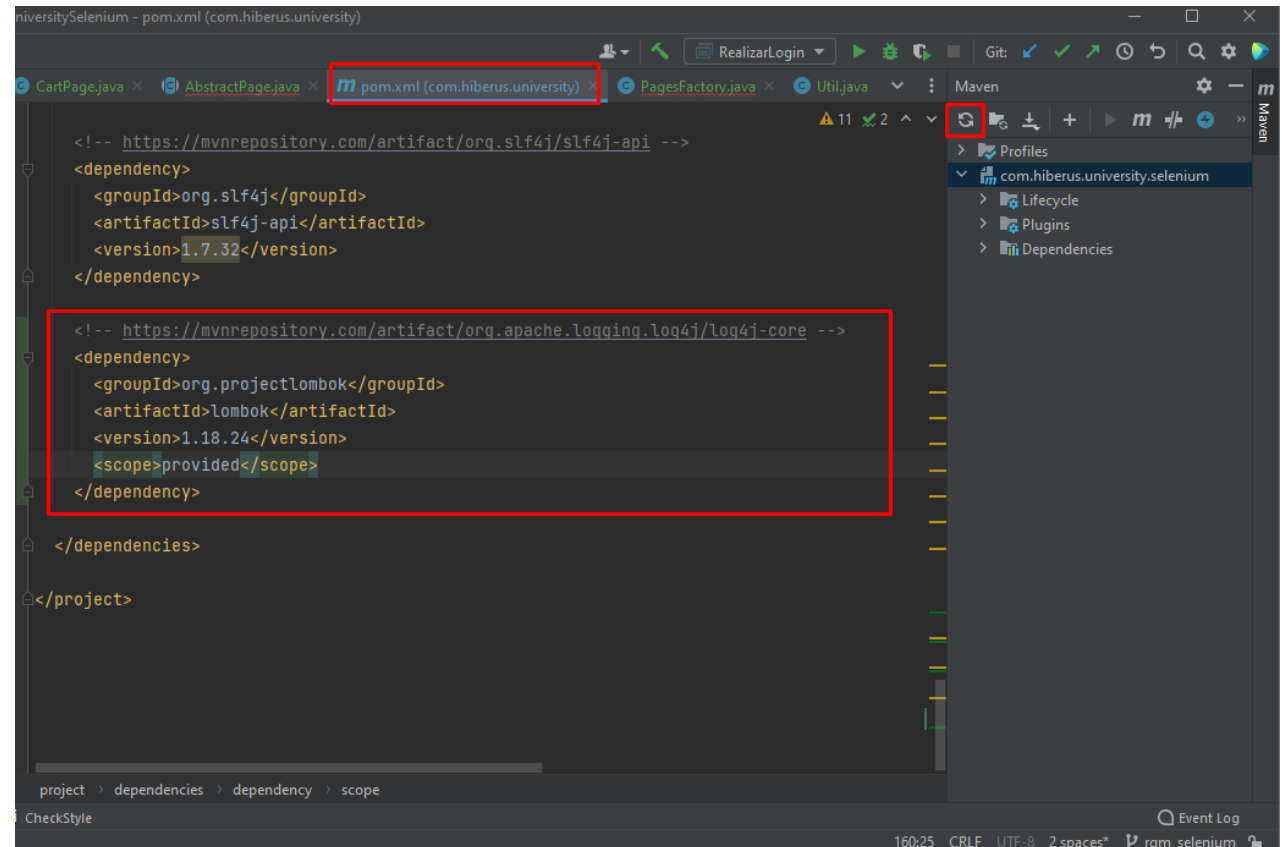
    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }
}
```

Buenas Prácticas

Patrón Page Object Model – Ejercicio 2

- Añadir dependencia Lombok:
 - <https://projectlombok.org/setup/maven>
 - Facilita el código por anotación
 - Getters
 - Setters
 - Constructores
 - Logs
 - ...
 - Copiar la dependencia en el pom.xml del proyecto
 - Actualizar dependencias



Buenas Prácticas

Patrón Page Object Model – Ejercicio 2

- Obtenemos la clase **MyFluentWait.java**:
 - Dentro del paquete util
 - Implements de `Wait<T>`
 - Añadir atributos:
 - timeout
 - interval
 - Crear constructores
 - Métodos:
 - `withTimeout()`
 - `pollingEvery()`
 - `until()`

<https://www.codepile.net/>

```
1 package com.hiberus.university.selenium.util;
2
3 // Licensed to the Software Freedom Conservancy (SFC) under one
4 // or more contributor license agreements. See the NOTICE file
5 // distributed with this work for additional information
6 // regarding copyright ownership. The SFC licenses this file
7 // to you under the Apache License, Version 2.0 (the
8 // "License"); you may not use this file except in compliance
9 // with the License. You may obtain a copy of the License at
10 //
11 // http://www.apache.org/licenses/LICENSE-2.0
12 //
13 // Unless required by applicable law or agreed to in writing,
14 // software distributed under the License is distributed on an
15 // "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
16 // KIND, either express or implied. See the License for the
17 // specific language governing permissions and limitations
18 // under the License.
19
20 import static com.google.common.base.Preconditions.checkNotNull;
21
22 import com.google.common.base.Throwables;
23 import com.google.common.collect.ImmutableList;
24 import com.google.common.collect.Lists;
25 import java.time.Clock;
26 import java.time.Duration;
27 import java.time.temporal.ChronoUnit;
28 import java.time.temporal.TemporalUnit;
29 import java.util.Collection;
30 import java.util.List;
31 import java.util.function.Function;
32 import java.util.function.Supplier;
33 import org.openqa.selenium.TimeoutException;
34 import org.openqa.selenium.WebDriverException;
35 import org.openqa.selenium.support.ui.Sleeper;
36 import org.openqa.selenium.support.ui.Wait;
37
38 /** An implementation of the {@link Wait} interface that may have its timeout and polling interval ... */
39 public class MyFluentWait<T> implements Wait<T>
40 {
41
42     public static final Duration FIVE_HUNDRED_MILLIS = Duration.of( amount: 500, ChronoUnit.MILLIS);
43
44     private final T input;
45     private final Clock clock;
46     private final Sleeper sleeper;
47
48     private Duration timeout = FIVE_HUNDRED_MILLIS;
49     private Duration interval = FIVE_HUNDRED_MILLIS;
50     private Supplier<String> messageSupplier = () -> null;
51
52     private List<Class<? extends Throwable>> ignoredExceptions = Lists.newLinkedList();
53 }
```

Buenas Prácticas

Patrón Page Object Model – Ejercicio 2

- Creación clase **AbstractPage.java**:
 - Dentro del paquete pages
 - Crear clase abstracta
 - Añadir atributos:
 - WebDriver
 - Wait<WebDriver
 - Crear constructores
 - Añadir anotación @Slf4j
 - Métodos:
 - moveTo()
 - isPageLoaded()
 - navigateTo()

```
@Slf4j
abstract class AbstractPage {
    private WebDriver driver;
    protected Wait<WebDriver> wait;

    AbstractPage(WebDriver driver) {
        this.driver = driver;
    }

    public abstract WebElement getPageLoadedTestElement();

    protected WebDriver getDriver() {
        return driver;
    }

    protected Wait<WebDriver> getWait() { return wait; }

    protected void setWait(Wait<WebDriver> wait) { this.wait = wait; }

    public void waitForPageLoad() {
        WebElement testElement = getPageLoadedTestElement();
        wait.until(ExpectedConditions.visibilityOf(testElement));
    }

    protected void moveTo(WebElement elem) {
        if (((RemoteWebDriver) driver).getCapabilities().getBrowserName().equals("firefox")) {
            ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", elem);
        } else {
            Actions actions = new Actions(driver);
            actions.moveToElement(elem).build().perform();
        }
    }

    protected boolean isPageLoaded(WebElement elem) {
        boolean isLoading = false;

        try {
            isLoading = elem.isDisplayed();
        } catch (org.openqa.selenium.NoSuchElementException e) {
            e.printStackTrace();
        }

        return isLoading;
    }

    public void navigateTo(String url) {
        WebDriver driver = getDriver();

        try {
            driver.navigate().to(url);
        } catch (java.lang.Exception e) {
            if (e instanceof TimeoutException) {
                log.info("Timeout loading home page");
            } else if (e instanceof ScriptTimeoutException) {
                log.info("Script Timeout loading home page");
            } else {
                log.error(e.getMessage());
            }
        }
    }
}
```


Buenas Prácticas

Patrón Page Object Model – Ejercicio 2

- Creación clase **PagesFactory.java**:
 - Dentro del paquete pages
 - Crear clase pública
 - Añadir atributos:
 - WebDriver
 - List<PagesFactory>
 - Crear constructores
 - Añadir anotación @Slf4j
 - Métodos:
 - start()
 - getInstance()
 - getDriver()

```
package com.hiberus.university.selenium.pages;

import org.openqa.selenium.WebDriver;

public class PagesFactory {

    private static PagesFactory pagesFactories;
    private final WebDriver driver;

    public PagesFactory(WebDriver driver) {
        this.driver = driver;
    }

    public static void start(WebDriver driver) {
        pagesFactories = new PagesFactory(driver);
    }

    public static PagesFactory getInstance() {
        return pagesFactories;
    }

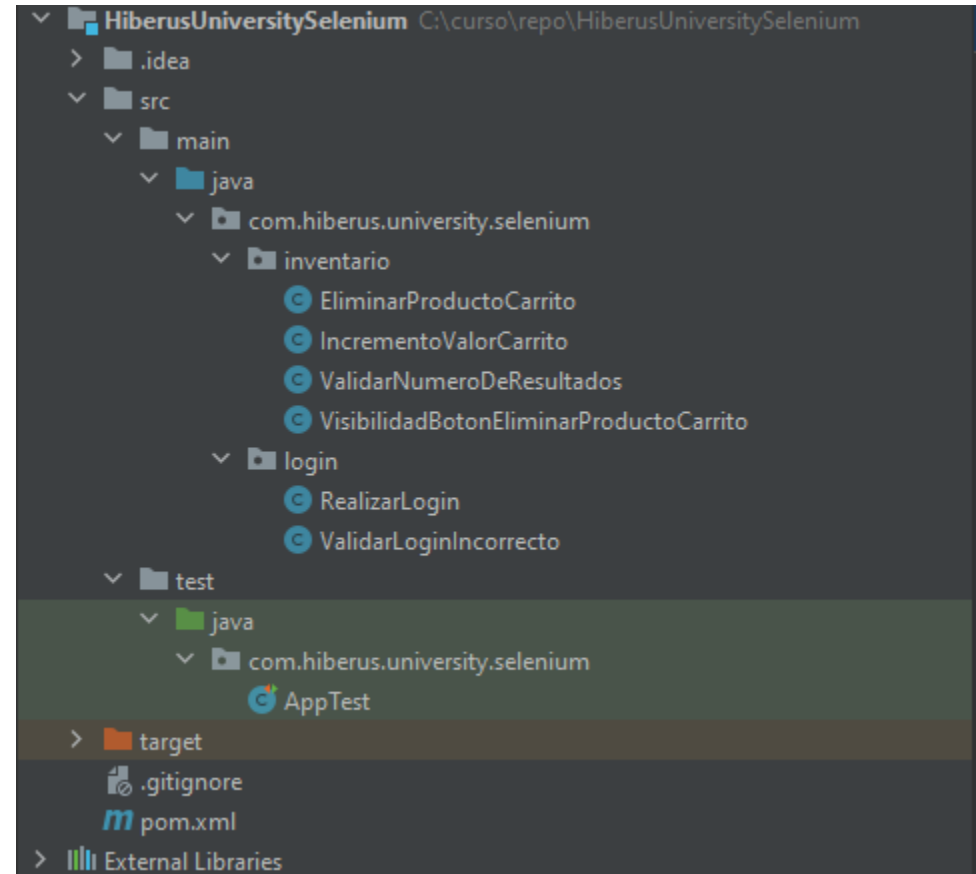
    public WebDriver getDriver() {
        return driver;
    }
}
```

Buenas Prácticas

Patrón Page Object Model

Del Proyecto actual

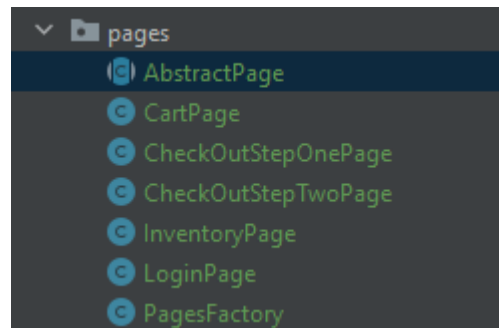
- Paso 1: Definir las páginas y sus elementos
- Paso 2: Implementar una estructura común
- **Paso 3: Implementar cada página con sus elementos y métodos**



Buenas Prácticas

Patrón Page Object Model – Ejercicio 2

- Ejemplo LoginPage.java
 - Extiende de AbstractPage.java
 - Constante con la URL de la página
 - Elementos definidos anteriormente
 - Métodos definidos anteriormente



```
@Slf4j
public class LoginPage extends AbstractPage {
    public static final String PAGE_URL = "https://www.saucedemo.com";

    @FindBy(xpath = "//input[@data-test='username']")
    private WebElement usernameElem;

    @FindBy(xpath = "//input[@data-test='password']")
    private WebElement passwordElem;

    @FindBy(xpath = "//input[@value='LOGIN']")
    private WebElement loginElem;

    public LoginPage(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, page: this);
    }

    @Override
    public WebElement getPageLoadedTestElement() { return loginElem; }

    public void login() {
        log.info("Logging in...");
        try {
            loginElem.click();
        } catch (org.openqa.selenium.TimeoutException e) {
            log.info("Timeout clicking login: " + e.getClass().getSimpleName());
        } catch (Exception e) {
            log.info("Clicking login, caught exception, type=" + e.getClass().getSimpleName());
        }
    }

    public void clickLogin() { loginElem.click(); }

    public void enterPassword(String password) {
        passwordElem.click();
        passwordElem.sendKeys(password);
    }

    public void enterUsername(String username) {
        usernameElem.click();
        usernameElem.sendKeys(username);
    }

    public boolean hasLockedOutError() {
        WebElement elem = getDriver().findElement(By.xpath("//button[@class='error-button']"));
        return elem.isDisplayed();
    }

    public boolean hasUsernamePasswordError() {
        WebElement elem = getDriver().findElement(By.xpath("//button[@class='error-button']"));
        return elem.isDisplayed();
    }
}
```

Buenas Prácticas

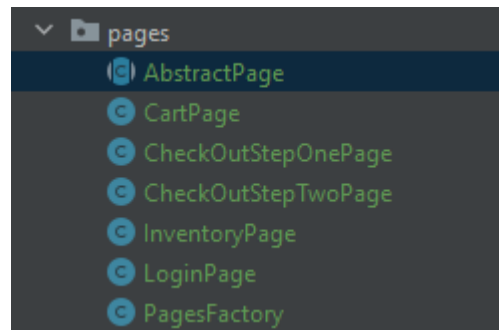
Patrón Page Object Model – Ejercicio 2

- Ejemplo LoginPage.java
 - Extiende de AbstractPage.java
 - Constante con la URL de la página
 - Elementos definidos anteriormente
 - Métodos definidos anteriormente

Ejercicio 2 (2 horas)

Creación del resto de clases *Page.java de la página sauceDemo con los elementos y métodos definidos en el ejercicio anterior:

- CartPage
- CheckoutStepOnePage
- CheckoutStepSecondPage
- InventoryPage
- LoginPage



```
@Slf4j
public class LoginPage extends AbstractPage {
    public static final String PAGE_URL = "https://www.saucedemo.com";

    @FindBy(xpath = "//input[@data-test='username']")
    private WebElement usernameElem;

    @FindBy(xpath = "//input[@data-test='password']")
    private WebElement passwordElem;

    @FindBy(xpath = "//input[@value='LOGIN']")
    private WebElement loginElem;

    public LoginPage(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, page: this);
    }

    @Override
    public WebElement getPageLoadedTestElement() { return loginElem; }

    public void login() {
        log.info("Logging in...");
        try {
            loginElem.click();
        } catch (org.openqa.selenium.TimeoutException e) {
            log.info("Timeout clicking login: " + e.getClass().getSimpleName());
        } catch (Exception e) {
            log.info("Clicking login, caught exception, type=" + e.getClass().getSimpleName());
        }
    }

    public void clickLogin() { loginElem.click(); }

    public void enterPassword(String password) {
        passwordElem.click();
        passwordElem.sendKeys(password);
    }

    public void enterUsername(String username) {
        usernameElem.click();
        usernameElem.sendKeys(username);
    }

    public boolean hasLockedOutError() {
        WebElement elem = getDriver().findElement(By.xpath("//button[@class='error-button']"));
        return elem.isDisplayed();
    }

    public boolean hasUsernamePasswordError() {
        WebElement elem = getDriver().findElement(By.xpath("//button[@class='error-button']"));
        return elem.isDisplayed();
    }
}
```