Memoria TQS practica 2

David Feliu de la peña Vilarroig 1598106
Pol Muñoz Lorenzo 1601912

Lista:

AddShoppingCart: Añadir al carrito de compra un articulo.

AnarAOfertaFlash: Ir a la pagina de ofertas Flash desde el menu de la pagina principal

SearchBags: Buscar un articulo en el buscador de la pagina web

NewLetter: Subscribirse al NewLetter de la pagina web.

Register: Utilizar el form del registro para ver si funcionaba bien.

Escenarios:

AddShoppingCart

El escenario AddShoppingCart es parte de una suite de pruebas automatizadas diseñada para verificar la funcionalidad de una aplicación web, específicamente la funcionalidad de añadir un producto al carrito de compras en el sitio web de MediaMarkt. Utiliza Selenium WebDriver para la interacción con el navegador y Cucumber para la definición de los pasos de prueba basados en un lenguaje de alto nivel. A continuación, se detalla cada parte del código:

1. **Configuración de Paquete y Dependencias:**
    - El código está encapsulado en el paquete `steps`, lo que ayuda en la organización y modularidad del proyecto.
    - Se importan clases de `io.cucumber` para utilizar las anotaciones `@Given`, `@When`, y `@Then`, que permiten definir los pasos de la prueba en un lenguaje natural.

- Se importan clases de `org.openqa.selenium` y `org.openqa.selenium.support.ui` para interactuar con elementos web y gestionar esperas, respectivamente.
- La clase `org.testng.Assert` se utiliza para realizar aserciones, es decir, para verificar si el resultado de la prueba cumple con las expectativas.

2. **Declaración de la Clase y Variables:**

- La clase `AddShoppingCartSteps` contiene todos los pasos de la prueba.
- Se declaran dos variables miembro de la clase:
  - `WebDriver driver`: para controlar el navegador.
  - `WebDriverWait wait`: para implementar esperas explícitas, asegurando que ciertos elementos estén presentes o en un estado específico antes de proceder.

3. **Pasos de la Prueba:**

- `@Given("the user is on the index web")`: Este paso inicializa el navegador, navega al sitio web de MediaMarkt, acepta las cookies y prepara el entorno para la prueba.
- `@When("the user clicks the television button")`: Simula el clic en el botón de televisión. Esto se logra encontrando el elemento por su selector CSS y haciendo clic en él. Luego, hay una pausa (`Thread.sleep`) para permitir que la página cargue.
- `@When("the user clicks the añadir carrito button")`: Simula el clic en el botón de añadir al carrito, usando la clase del botón para localizarlo.
- `@When("the user clicks the no gracias button")`: Simula el clic en el botón de "no gracias", para rechazar cualquier oferta adicional, y espera para asegurar que la acción se procese completamente.
- `@Then("appears the text Su producto fue añadido con éxito al carrito")`: Verifica si el producto fue añadido exitosamente al carrito. Busca un elemento por su clase que contiene el mensaje de confirmación y compara el texto de este elemento con el mensaje esperado. Si no coincide, la prueba fallará en este punto.

NewsLetter

1. **Configuración de Paquete y Dependencias:**

- El código pertenece al paquete `steps`, lo que facilita su organización y reutilización.

- Se importan clases específicas de `io.cucumber` para usar las anotaciones `@Given`, `@When`, y `@Then`, que permiten escribir pruebas en un lenguaje cercano al natural.
- Se importan clases de `org.openqa.selenium` y `org.openqa.selenium.support.ui` para la interacción con elementos web y la implementación de esperas explícitas.
- La clase `org.testng.Assert` se usa para realizar aserciones, comprobando si el resultado de la prueba coincide con lo esperado.

2. **Declaración de la Clase y Variables:**

- La clase `NewsletterSteps` se encarga de definir los pasos de la prueba para la suscripción al boletín.
- Se declaran variables para manejar el navegador (`WebDriver driver`) y las esperas (`WebDriverWait wait`).

3. **Pasos de la Prueba:**

- `@Given("the user is on the index")`: Este paso inicializa el navegador, navega al sitio web de MediaMarkt, acepta las cookies mediante la interacción con el botón correspondiente y establece las condiciones iniciales para la prueba.
- `@When("the user clicks the suscribe button")`: Este paso simula el clic en el botón de suscripción. Localiza el botón por su clase y luego por el texto del enlace y realiza un clic en él.
- `@When("the user enters his name, last name and email")`: En este paso, el usuario (simulado por el script de prueba) rellena los campos de nombre, apellido y correo electrónico en el formulario de suscripción. Utiliza los identificadores de los campos para localizarlos y envía las cadenas de texto correspondientes a cada campo.
- `@When("the user clicks the send button")`: Este paso representa el envío del formulario de suscripción al hacer clic en el botón de enviar. Utiliza el identificador del botón para localizarlo y hace clic en él. La pausa (`Thread.sleep`) permite esperar a que la acción se complete.
- `@Then("appears the text You have successfully subscribed to this newsletter")`: Este paso verifica si el proceso de suscripción fue exitoso. Busca un párrafo específico en la página (usando un selector CSS) que contiene la confirmación de suscripción y comprueba si el texto contiene el correo electrónico proporcionado, lo que indica que el proceso se completó correctamente.

Register

Escenario 1: Registro Exitoso

Descripción:

Este escenario representa el flujo completo de un usuario registrándose en el sitio de manera exitosa. Incluye la navegación al sitio, la interacción con el botón de perfil, el ingreso al formulario de registro, la introducción de un correo electrónico válido, y la verificación de que se muestra el mensaje indicativo de que el correo ha sido enviado.

Pasos del Escenario:

Inicio de la Sesión y Navegación:

- `@Given("the user is on the index menu")`: Se establece el navegador y se navega al sitio web de MediaMarkt, aceptando las cookies.

Acceso al Formulario de Registro:

- `@When("the user clicks the profile button")`: El usuario presiona el botón del perfil.
- `@When("the user clicks the registro button")`: El usuario selecciona la opción de registro.

Ingreso de Datos:

- `@When("the user enters an email")`: El usuario ingresa una dirección de correo electrónico válida en el formulario de registro.

Envío del Formulario:

- `@When("the user clicks the enviar codigo button")`: El usuario envía el formulario de registro.

Confirmación de Registro:

- `@Then("appears the text Comprueba tu buzón de correo")`: Se verifica que aparece el mensaje indicando que se debe comprobar el buzón de correo, confirmando que el proceso de registro ha iniciado correctamente.

Registro con Correo Electrónico Inválido

Descripción:

Este escenario simula el intento de registro con un correo electrónico mal formateado. Se enfoca en validar que el sistema identifica y notifica correctamente al usuario sobre el error en el formato del correo electrónico.

Pasos del Escenario:

Inicio de la Sesión y Navegación:

- `@Given("the user is on the index menu")`: Igual que en el escenario anterior, el usuario inicia la sesión y navega al sitio web, aceptando las cookies.

Acceso al Formulario de Registro y Entrada de Datos Inválidos:

- `@When("the user clicks the profile button")` y `@When("the user clicks the registro button")`: El usuario accede al formulario de registro igual que en el escenario anterior.
- `@When("the user enters bad email")`: El usuario introduce un correo electrónico con un formato incorrecto.

Intento de Envío del Formulario:

- `@When("the user clicks the enviar codigo button")`: El usuario intenta enviar el formulario de registro.

Verificación de Error:

- `@Then("appears the text correo invalido")`: Se verifica que el sistema muestra un mensaje indicando que el formato del correo electrónico es inválido.

Oferta Flash

1. **Configuración de Paquete y Dependencias:**

- El código forma parte del paquete `steps`, lo que favorece su organización y reutilización.
- Se importan clases de `java.time.Duration` para manejar duraciones específicas durante la espera de elementos.
- Se importan clases de `org.openqa.selenium` y `org.openqa.selenium.support.ui` para la interacción con elementos web y la implementación de esperas explícitas.

- Se utilizan las anotaciones de Cucumber `@Given`, `@When`, y `@Then` para estructurar la prueba en pasos legibles y bien definidos.
- La clase `org.testng.Assert` se usa para realizar aserciones, verificando si el resultado de la prueba es el esperado.

2. **Declaración de la Clase y Variables:**

- La clase `AnarFinanciacio` está diseñada para definir los pasos de la prueba para verificar la funcionalidad de financiación en el sitio web.
- Se declaran variables para manejar el navegador (`WebDriver driver`) y las esperas (`WebDriverWait wait`).

3. **Pasos de la Prueba:**

- `@Given("user is on the index page")`: Este paso inicializa el navegador, navega al sitio web de MediaMarkt, acepta las cookies y establece las condiciones iniciales para la prueba. También incluye una pausa (`Thread.sleep`) para asegurar que la página haya cargado completamente.
- `@When("user goes to financiacion")`: Este paso simula la acción del usuario de navegar a la sección de ofertas flash en el sitio web. Se localiza y hace clic en el elemento que lleva a la sección de oferta flash.
- `@Then("Financiacion a tu alcance")`: Este paso verifica si se ha navegado correctamente a la sección de financiación. Busca un título específico en la página que debería estar presente en la sección de financiación y comprueba que efectivamente contiene el texto esperado ("Ofertas"), confirmando que el usuario ha llegado a la sección correcta.

Search Bags

1. **Configuración de Paquete y Dependencias:**

- El código pertenece al paquete `steps`, lo que facilita su organización y reutilización.
- Se importan clases específicas de `io.cucumber` para usar las anotaciones `@Given`, `@When`, y `@Then`, que permiten escribir pruebas en un lenguaje cercano al natural.
- Se importan clases de `org.openqa.selenium` y `org.openqa.selenium.support.ui` para la interacción con elementos web y la implementación de esperas explícitas.
- La clase `org.testng.Assert` se usa para realizar aserciones, comprobando si el resultado de la prueba coincide con lo esperado.

2. **Declaración de la Clase y Variables:**

- La clase `SearchBagsSteps` se encarga de definir los pasos de la prueba para la funcionalidad de búsqueda en el sitio web.
- Se declaran variables para manejar el navegador (`WebDriver driver`) y las esperas (`WebDriverWait wait`).

3. **Pasos de la Prueba:**

- `@Given("the user is on the index page")`: Este paso inicializa el navegador, navega al sitio web de MediaMarkt, acepta las cookies mediante la interacción con el botón correspondiente y establece las condiciones iniciales para la prueba.
- `@When("the user enters bags in the search bar")`: Este paso simula la acción de escribir "ordenador" en la barra de búsqueda, preparando el sistema para realizar una búsqueda basada en este término.
- `@When("the user clicks the search button")`: Este paso representa la ejecución de la búsqueda. El usuario (simulado por el script de prueba) hace clic en el botón de búsqueda. Se utiliza una pausa (`Thread.sleep`) para dar tiempo a que la página muestre los resultados.
- `@Then("the bags list appears")`: Este paso verifica si los resultados de la búsqueda son los esperados. Se busca un elemento específico en la página que debería contener el texto relacionado con la búsqueda ("ordenador") y se comprueba que efectivamente contiene este texto, lo que indica que la búsqueda ha sido exitosa y los resultados son los correctos.

Conclusions

La practica nos ha parecido muy interesante, gracias a las diferentes tecnicas que hemos aprendido podriamos testear de forma semiautomatica cualquier pagina web.

La parte que mas nos ha costado ha sido instalar los drivers de firefox y lo tuvimos que hacerlo con google chrome esto nos alargo bastante el proceso pero gracias a los foros de internet pudimos solucionarlo sin problema.

Lo mas interesante de todo ha sido que si se quiere hacer este tipo de test se tendría que desarrollar la pagina pensando que se van hacer estos test para poner ids i classes correctas a los objetos que son practicas no muy habituales entre programadores.

AutoEvaluacio:

- features realitzados
    - AddShoppingCart Escenarios → 1
    - AnarAOfertaFlash Escenarios → 1
    - SearchBags Escenarios → 1
    - NewLetter Escenarios → 1
    - Register Escenarios → 2
- Reusabilitat dels steps → nota 6
- Enteniment dels steps i features → 10
- Driver management → 7
- Nota global → 6,5