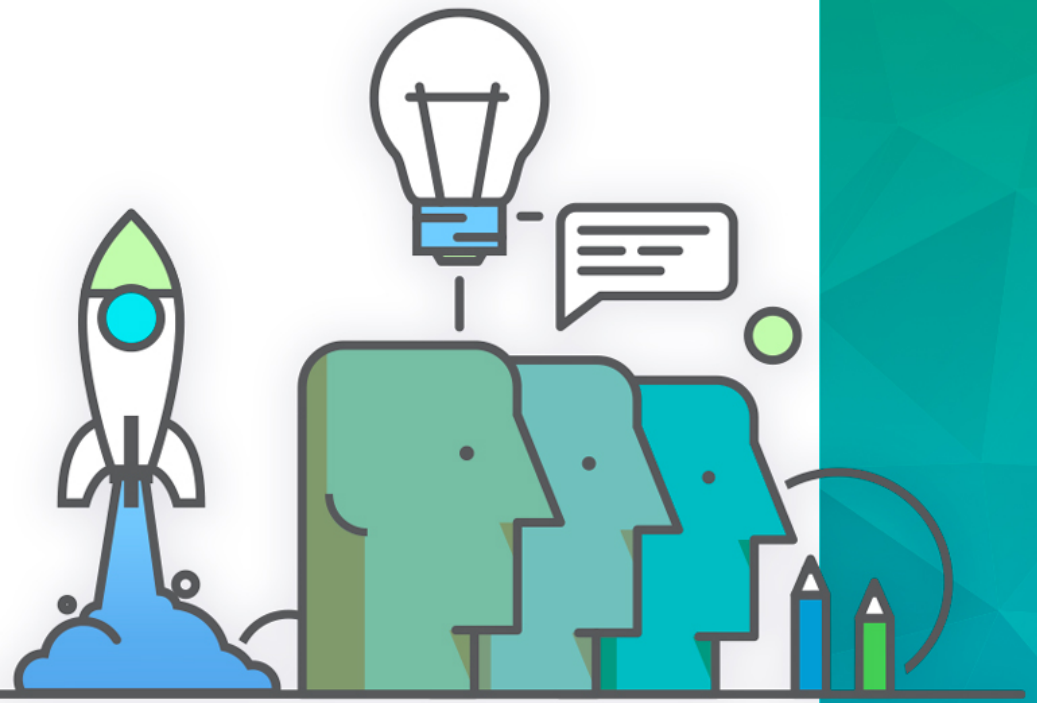
campus
Periferia IT

Periferia IT
Group



Automatización

Módulo 6: POM

Un curso del Área DevOps
Periferia IT Group

Periferia IT
Corp

CONTACT&
BUSINESSIT

COOLEXPRESS

Contenido

- ¿Que es un patrón de diseño?
- Introducción a POM
- Detalles de la implementación
- Definición de Maven

¿QUE ES UN PATRÓN DE DISEÑO?

- Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.
- Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.
- Existen varios patrones de diseño popularmente conocidos:
    - Patrones Creacionales: Inicialización y configuración de objetos.
    - Patrones Estructurales: Separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes.
    - Patrones de Comportamiento: Más que describir objetos o clases, describen la comunicación entre ellos.

PATRONES CREACIONALES

El problema a solucionar por este patrón es el de crear diferentes familias de objetos, como por ejemplo la creación de interfaces gráficas de distintos tipos (ventana, menú, botón, etc.).

Parte del principio de que las subclases determinan la clase a implementar.

PATRONES COMPORTAMIENTO:

- Cadena de responsabilidad (Chain of responsibility): La base es permitir que más de un objeto tenga la posibilidad de atender una petición.
- Orden (Command): Encapsula una petición como un objeto dando la posibilidad de "deshacer" la petición
- Intérprete (Interpreter): Intérprete de lenguaje para una gramática simple y sencilla.
- Iterador (Iterator): Define una interfaz que declara los métodos necesarios para acceder secuencialmente a una colección de objetos sin exponer su estructura interna.
- Mediador (Mediator): Coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.
- Recuerdo (Memento): Almacena el estado de un objeto y lo restaura posteriormente.
- Observador (Observer): Notificaciones de cambios de estado de un objeto.

```
public sealed class Singleton
      {
            private static volatile Singleton instance;
            private static object syncRoot = new Object();
            private Singleton()
            {
                  System.Windows.Forms.MessageBox.Show("Nuevo Singleton");
            }
            public static Singleton GetInstance
            {
                  get
                  {
                        if (instance == null)
                        {
                              lock(syncRoot)
                              {
                                    if (instance == null)
                                          instance = new Singleton();
                              }
                        }
                        return instance;
                  }
            }
       }
```

INTRODUCCIÓN A POM

Es un patrón de diseño que se utiliza en pruebas automatizadas para evitar código duplicado y mejorar el mantenimiento de las mismas.

Un punto importante a tener en cuenta cuando automatizamos es que la UI (interfaz de usuario) puede sufrir cambios. Si nuestras pruebas manipulan directamente los elementos de la página, serán frágiles y requerirán alto mantenimiento.

Cuando diseñamos una prueba automatizada para un sitio web, debemos consultar sus elementos para hacer clic en enlaces/botones y determinar qué se muestra a continuación.

El patrón Page Object encapsula el comportamiento de las páginas, o una parte de ella, con una API específica de la aplicación, lo que nos permite escribir tests y manipular elementos de página sin tener que lidiar con el HTML.

El patrón Page Object se usa para hacer pruebas funcionales automatizadas.

La idea es modelar las páginas y sus comportamientos para lograr pruebas claras de escribir, entender y mantener

Al crear un PageObject, lo que estamos consiguiendo es crear una capa de abstracción entre el « ¿Qué podemos hacer/ver en la página? », y el « Cómo » se realiza esta acción, simplificando enormemente la creación de nuestras pruebas y reutilizando el código con el que interactuamos con la página en concreto.

Cualquier cambio que se produzca en la UI únicamente afectará al PageObject en cuestión, no a los test ya implementados.

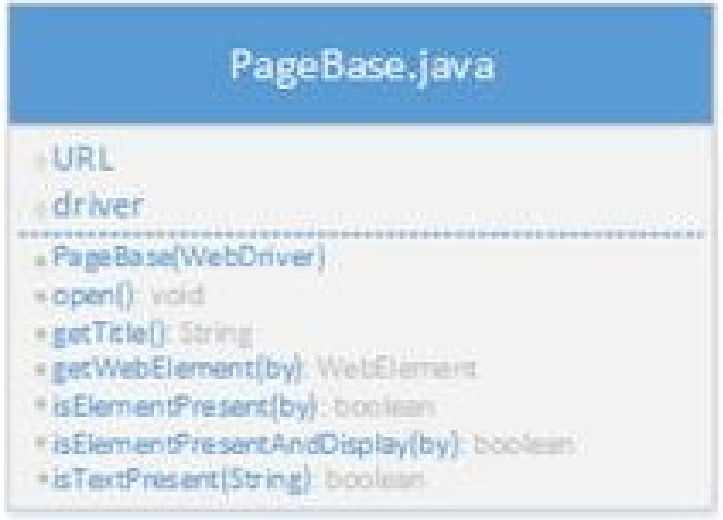
Esto se debe a que un test nunca debe manipular directamente elementos de la página (UI), si no que este manejo debe realizarse a través del PageObject correspondiente que representa la página.

El PageObject se convierte en una API con la que fácilmente podemos encontrar y manipular los datos de la página.

DETALLES DE LA IMPLEMENTACIÓN

Una de las opciones de las que disponemos es crear, en primer lugar, una clase "básica", que posteriormente será la que extenderán cada uno de los distintos **PageObjects** que vayamos implementando.

Esta **PageBase** nos aporta la estructura básica y las propiedades generales que utilizaremos:



Una vez creada esta página, crear los **PageObjects** necesarios. Un ejemplo de una página de "Login" podría ser:

Lógicamente, si lo creemos conveniente nos podemos saltar la creación de la **PageBase**, y crear directamente los distintos **PageObjects** con todas las propiedades y métodos necesarios.

Los métodos que interactúan con la página en si, normalmente o bien no nos devolverán ningún parámetro, o bien serán de tipo sencillo, como boolean, string, etc, ej. returnErrorMessage()

Pero cuando se trata de funciones de navegación entre páginas, estas nos devolverán un nuevo objeto PageObject

DETALLES DE LA IMPLEMENTACIÓN

Assertions :

se incorpora una “assertion” o validación, que será la que indicará si la prueba ha sido satisfactoria o no. No existe una posición única sobre si las assertions deben incorporarse en el PageObject o bien en los tests.

```
@Test
public void testLoginToGmail() {

      GmailLoginPage logPage = new GmailLoginPage();
      EmailPage emailPage = logPage.loginAs("test@gmail.com","pass");

      //Assertions
      Assert.assertEquals("Inbox", emailPage.getTitle());
}
```

Conclusiones:

Sin duda alguna, los patrones de diseño en desarrollo, nos ofrecen una solución previamente utilizada y de efectividad probada, así como la correspondiente documentación, con la que enfrentarse a una serie de problemas comunes dentro del mundo del desarrollo y la automatización de pruebas. Y en concreto, el Page Object nos ofrece las ventajas de:

- Reducir la duplicidad de código.
- Conseguir pruebas más robustas y fácilmente entendibles
- Mejorar la mantenibilidad de nuestras pruebas funcionales automatizadas, punto que se pone de relevancia cuando tenemos una web o aplicación que evoluciona rápidamente.

DEFINICIÓN DE MAVEN

Apache Maven es una herramienta de comprensión y gestión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede administrar la construcción, informes y documentación de un proyecto a partir de una pieza central de información.

Maven es una forma fácil de publicar información del proyecto y una forma de compartir los archivos JAR en varios proyectos.

El resultado es una herramienta que ahora se puede utilizar para crear y gestionar cualquier proyecto basado en Java.

OBJETIVOS MAVEN

El objetivo principal de Maven es permitir que un desarrollador comprenda el estado completo de un esfuerzo de desarrollo en el período de tiempo más corto. Para lograr este objetivo, Maven se ocupa de varias áreas de interés:

- Facilitando el proceso de construcción: Si bien el uso de Maven no elimina la necesidad de conocer los mecanismos subyacentes, Maven protege a los desarrolladores de muchos detalles.
- Proporcionar un sistema de construcción uniforme: Maven crea un proyecto utilizando su modelo de objetos de proyecto (POM)
- Proporcionar información de calidad sobre el proyecto: Maven proporciona información útil del proyecto que en parte se toma de su POM y en parte se genera a partir de las fuentes de su proyecto.
- Fomento de mejores prácticas de desarrollo: Maven tiene como objetivo recopilar los principios actuales para el desarrollo de mejores prácticas y facilitar la orientación de un proyecto en esa dirección.

CARACTERISTICAS MAVEN

- Configuración de proyecto simple que sigue las mejores prácticas: inicie un nuevo proyecto o módulo en segundo.
- Gestión superior de dependencias, incluida la actualización automática, cierres de dependencias.
- Capaz de trabajar fácilmente con múltiples proyectos al mismo tiempo
- Un repositorio grande y en crecimiento de bibliotecas y metadatos para usar de forma inmediata, y arreglos establecidos con los proyectos de código abierto más grandes para la disponibilidad en tiempo real de sus últimas versiones.
- Extensible, con la capacidad de escribir complementos fácilmente en Java o lenguajes de secuencias de comandos
- Acceso instantáneo a nuevas funciones con poca o ninguna configuración adicional.
- Maven puede compilar cualquier cantidad de proyectos en tipos de salida predefinidos, como JAR, WAR o distribución basada en metadatos sobre el proyecto, sin la necesidad de realizar scripts en la mayoría de los casos.
- Maven puede generar un sitio web o un PDF que incluye cualquier documentación que desee agregar, y agrega informes estándar sobre el estado de desarrollo del proyecto
- Maven se integrará con su sistema de control de código fuente (como Subversion o Git) y gestionará la publicación de un proyecto en función de una determinada etiqueta.
- Gestión de dependencias: Maven fomenta el uso de un repositorio central de archivos JAR y otras dependencias. Maven viene con un mecanismo que los clientes de su proyecto pueden usar para descargar cualquier archivo JAR

campus
Periferia IT

G R A C I A S

Periferia IT Group

Periferia IT Corp

CONTACT& BUSINESSIT

COOLEXPRESS