**Integración de ChatGPT con Java, Selenium y TestNG framework de automatización**

Imagen que contiene luz

Descripción generada automáticamente

Este documento tiene como objetivo explicar cómo integrar ChatGPT con Java, Selenium y TestNG frameworks de automatización para pruebas automatizadas basadas en lenguaje natural. Los lectores entenderán mejor ChatGPT y sus capacidades y cómo puede simplificar el proceso de pruebas.

El documento también explora el potencial de ChatGPT como una solución de bajo código para probar la automatización y discute sus limitaciones.

**Introducción**

Como todos sabemos, la inteligencia artificial es el futuro. Actualmente, ChatGPT, un modelo de chatbot/entrenado OpenAI que todavía está siendo creado y entrenado por sus ingenieros, es un tema candente en todo el mundo. Una IA conversacional que puede responder preguntas en lenguaje sencillo, escribir poemas, guiones, publicaciones en redes sociales, ensayos descriptivos y muchas otras cosas. En su primera semana de servicio, ChatGPT rompió todos los récords de Internet al volverse extremadamente popular. Así que me pregunté cómo podría usarlo para facilitar los esfuerzos de prueba de automatización.

Como las pruebas de automatización se pueden realizar con más frecuencia y requieren menos tiempo y esfuerzo que las pruebas manuales, proporcionan una respuesta más rápida. Sin embargo, las pruebas de automatización también presentan desafíos, como el requisito de conocimientos técnicos para escribir y mantener guiones de prueba, el alto coste de las herramientas utilizadas en las pruebas de automatización, La dificultad de desarrollar casos de prueba que sean eficientes y precisos.

Aquí es donde ChatGPT puede ser potencialmente una herramienta valiosa para las pruebas de automatización. Al integrar ChatGPT con marcos de pruebas de automatización, se pueden crear casos de prueba utilizando lenguaje natural. Además, ChatGPT puede generar fragmentos de código y guiar las mejores prácticas, ahorrando tiempo y esfuerzo en el proceso de pruebas de automatización. Sin embargo, es esencial tener en cuenta las limitaciones de ChatGPT, como la necesidad de comprobar y probar a fondo el código generado antes de usarlo en un proyecto.

**ChatGPT**

ChatGPT es un chatbot conversacional de procesamiento de lenguaje natural que utiliza el modelo Generative Pre-Trained Transformer (GPT). ChatGPT, que se basa en GPT-3.5, puede comunicarse usando inglés natural. Es capaz de responder correctamente a las consultas de una manera humana.

ChatGPT es una aplicación web basada en un modelo de IA que permite a los usuarios interactuar con la IA de OpenAI de forma natural y conversacional. Puede interpretar la entrada del lenguaje natural, responder a ella y recordar conversaciones anteriores para dar respuestas más precisas y relevantes.

Para automatizar procesos como la generación de código y las pruebas, ChatGPT puede producir texto en una variedad de formas, incluyendo datos estructurados, fragmentos de código y anotaciones.

Aquí hay algunas maneras en que ChatGPT puede ser usado para aprender a codificar:

* La generación de código es un excelente método para que los principiantes aprendan la sintaxis y estructura fundamental de un lenguaje de programación, ya que puede generar fragmentos de código en respuesta a las indicaciones del lenguaje natural.
* Puede ser usado para terminar el código parcialmente escrito, dar instrucciones sobre qué hacer a continuación y ayudar a los principiantes en la comprensión de cómo manejar un problema específico.
* Puede utilizarse para ilustrar la función y el significado de líneas particulares de código, ayudando a los nuevos programadores a aprender cómo interactúan las diversas partes de un programa.
* Para principiantes que quieran crear un proyecto, puede ofrecer orientación y recomendaciones sobre la estructura organizativa del proyecto, las mejores prácticas y bibliotecas a utilizar.

**Integración de ChatGPT con un marco de pruebas**

El primer paso es la instalación de los programas y bibliotecas necesarios. Java, Selenium y TestNG deben estar instalados en su computadora.

**Paso 1: Requisitos previos**

* [Java](https://www.java.com/en/download/) instalado en su sistema
* [Selenium](https://www.selenium.dev/downloads/) WebDriver
* [TestNG](https://testng.org/doc/download.html) testing framework
* Biblioteca de POI para leer/escribir archivos de Excel
* Acceso a [OpenAPI](https://platform.openai.com/account/api-keys) para clave API

**Paso 2: Hoja de cálculo con detalles del paso de prueba**

El siguiente paso es crear la hoja de cálculo con los pasos de prueba que queremos ejecutar. Podemos utilizar cualquier herramienta, como Google Sheets o Microsoft Excel. Estoy usando una hoja de Excel como herramienta de demostración en este ejemplo. Aquí hay una muestra de cómo podría verse una hoja de Excel:

Tabla

Descripción generada automáticamente

Hoja de cálculo con la información de solicitud y los nombres de casos de prueba

**Paso 3: Código para leer los detalles de la hoja de cálculo**

Cree un método que lea los pasos de prueba de la hoja de Excel. Aquí hay un ejemplo de cómo puedes leer los pasos de prueba usando la biblioteca POI de Apache:

public class ExcelReader {  
 private List<String> testSteps = new ArrayList<String>();  
 private List<String> testCaseNames = new ArrayList<String>();

public ExcelReader() {  
 // Open the Excel file  
 String currentDirectory = System.getProperty("user.dir");  
 // System.out.println("The current working directory is " + currentDirectory);  
 String filePath = currentDirectory + "/src/main/resources/test-steps.xlsx";  
 // System.out.println("The current working file is " + filePath);  
 FileInputStream file = null;  
 XSSFWorkbook workbook = null;  
 try {  
 file = new FileInputStream(new File(filePath));  
 workbook = new XSSFWorkbook(file);  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
 XSSFSheet sheet = workbook.getSheetAt(0);  
 // Iterate through the rows and columns of the sheet to read the test case names  
 for (Row row : sheet) {  
 Cell cell = row.getCell(0);  
 String testStep = cell.getStringCellValue();  
 testCaseNames.add(testStep);  
 }  
 // Iterate through the rows and columns of the sheet to read the test steps  
 for (Row row : sheet) {  
 Cell cell = row.getCell(1);  
 String testStep = cell.getStringCellValue();  
 testSteps.add(testStep);  
 }  
 // Close the workbook and file  
 try {  
 workbook.close();  
 file.close();  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 }  
 } public List<String> readTestSteps() throws Exception {  
 return testSteps;  
 } public List<String> readTestCaseNames() throws Exception {  
 return testCaseNames;  
 }

Este programa Java llamado ExcelReader lee datos del archivo Excel test-steps.xlsx. Dos variables de instancia privada llamadas testSteps y testCaseNames son listas de Strings utilizadas para contener los datos leídos del archivo de Excel.

El constructor de clases inicializa las variables de instancia leyendo el archivo de Excel y llenando las listas con los pasos de prueba y los nombres de caso de prueba. El primer paso es obtener el directorio actual usando el método System.getProperty() y luego agregar el nombre del archivo de Excel para crear la ruta del archivo. Las clases FileInputStream y XSSFWorkbook se utilizan para leer el archivo de Excel.

El código entonces itera a través de cada fila en la primera hoja del archivo de Excel y recupera las columnas primera y segunda de cada fila. La primera columna contiene los nombres de casos de prueba, que se añaden a la lista testCaseNames y la segunda columna contiene los pasos de prueba, que se añaden a la lista testSteps.

La clase proporciona dos métodos públicos, readTestSteps() y readTestCaseNames(), que devuelven las listas pobladas de pasos de prueba y los nombres de casos de prueba, respectivamente. Otras clases pueden utilizar estos métodos para recuperar los datos del archivo de Excel.

**Paso 4: Crear los archivos de prueba y enviar un mensaje a ChatGPT**

Cree un método que genere archivos de prueba Java basados en los pasos de prueba leídos de la hoja de Excel. El método chatGPTCodeGenerator() es responsable de acceder a las API de chatGPT para generar el código para el prompt proporcionado y leer la respuesta.

Aquí hay un ejemplo de cómo crear archivos de prueba en Java:

public void CreateTestCasesForScenarios() {  
 try {  
 List<String> testSteps = reader.readTestSteps();  
 List<String> testCaseNames = reader.readTestCaseNames();  
 int index =0;  
 for (String testStep : testSteps) {  
 // Create a new test case for the test step  
 String testCaseName = testCaseNames.get(index);

// Generate the code for the test step using the ChatGpt API  
 String generatedCode = GPT.chatGPTCodeGenerator(testStep, testCaseName); // Create a test file  
 String fileName = testCaseName + ".java";  
 String filePath = folderPath + "/" + fileName;  
 File file = new File(filePath);  
 file.getParentFile().mkdirs(); // Create parent directories if they don't exist  
 FileWriter writer = new FileWriter(file); String testCase = generatedCode;  
 writer.write(testCase);  
 System.out.println("Test Case for " + testStep + ": " + testCase);  
 index++;  
 writer.close();  
 }  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }

Este es un método Java llamado **CreateTestCasesForScenarios()** que crea casos de prueba basados en los pasos de prueba leídos desde un archivo de Excel. El método utiliza una instancia de la clase ExcelReader para leer los pasos de prueba y los nombres de caso de prueba del archivo de Excel.

El código itera a través de cada paso de prueba en la lista testSteps y genera un nuevo caso de prueba para cada paso de prueba. Utiliza la lista testCaseNames para obtener el nombre del caso de prueba correspondiente al paso de prueba actual utilizando la variable index para realizar un seguimiento de la posición actual en la lista.

A continuación, el método utiliza una API externa llamada ChatGPT para generar el código para el paso de prueba. El método GPT.chatGPTCodeGenerator() toma en la etapa de prueba y el nombre del caso de prueba y el código Java para el caso de prueba. El código generado se almacena en una variable de cadena llamada generatedCode.generates

Luego, el método crea un nuevo archivo Java para el caso de prueba usando el nombre del caso de prueba y escribe el código generado en el archivo. El archivo se guarda en una carpeta especificada utilizando la clase FileWriter.

Finalmente, el método imprime el caso de prueba generado a la consola para fines de depuración, incrementa la variable index para mover al siguiente caso de prueba y cierra el objeto FileWriter.

Si se produce alguna excepción durante la ejecución del código, el método lanza una RuntimeException.

Aquí encontrarás el fragmento de código para el método chatGPTCodeGenerator().

public String chatGPTCodeGenerator(String text, String className) throws Exception {

String apiKey = "Your API\_KEY";  
 String endpointUrl = "https://api.openai.com/v1/completions";  
 String prompt = "Create java, selenium and testNG code to " + text + "Consider " + className + "for class name. You can skip the logic for driver setup and cleaning activity as I already have it created as a part of framework"; HttpURLConnection con = (HttpURLConnection) new URL(endpointUrl).openConnection();  
 con.setRequestMethod("POST");  
 con.setRequestProperty("Content-Type", "application/json");  
 con.setRequestProperty("Authorization", "Bearer "+apiKey); JSONObject data = new JSONObject();  
 data.put("model", "text-davinci-003");  
 data.put("prompt", prompt);  
 data.put("max\_tokens", 4000);  
 data.put("temperature", 1.0);  
 con.setDoOutput(true);  
 con.getOutputStream().write(data.toString().getBytes()); String output = new BufferedReader(new InputStreamReader(con.getInputStream())).lines()  
 .reduce((a, b) -> a + b).get();  
 String generatedCode = new JSONObject(output).getJSONArray("choices").getJSONObject(0).getString("text");  
 return generatedCode;  
 }

Cuando se ejecute este código, se creará un nuevo archivo con el nombre que leemos de la hoja de Excel. Así es como aparecerá el archivo creado. De acuerdo con la forma en que se implementa mi framework, hice algunos cambios al archivo creado. Por ejemplo, para obtener la información del conductor, he ampliado la clase de la clase BaseTest, que contiene el código de configuración y desmontaje.

public class TC\_GithubLoginValidation extends BaseTest{  
 WebDriver driver;  
 @Test  
 public void loginValidation() {  
 driver.get("http://www.github.com");  
 WebElement username = driver.findElement(By.xpath("//input[@id='login\_field']"));  
 username.sendKeys("emailAddress");  
 WebElement password = driver.findElement(By.xpath("//input[@id='password']"));  
 password.sendKeys("Password");  
 WebElement signin = driver.findElement(By.xpath("//input[@value='Sign in']"));  
 signin.click();  
 WebElement errorMessage = driver.findElement(By.className("text-red-600"));  
 Assert.assertEquals(errorMessage.getText(), "Incorrect username or password.");  
 }  
}

**Limitaciones de ChatGPT para pruebas de automatización**

Aunque ChatGPT tiene mucho potencial para ser una solución de bajo código para pruebas automatizadas, todavía tiene problemas.

* El código generado por ChatGPT puede no siempre funcionar correctamente ya que no entiende el contexto o los requisitos de la tarea y también es menos libre de errores o optimizado. Antes de utilizar el código generado en cualquier proyecto, es crucial comprobarlo y probarlo a fondo.
* Dado que ChatGPT es un modelo de aprendizaje automático, puede ser difícil entender cómo produce determinados resultados, lo que hace que sea difícil hacer responsable al modelo por cualquier error o inexactitud.
* ChatGPT no puede encontrar métodos obsoletos, y no hay ninguna solución práctica para solucionar el problema.
* Normalmente, el código generado por ChatGPT sigue las estructuras de página estándar. Los usuarios deben, por tanto, identificar el problema para solicitar una versión actualizada del código.

**Conclusión**

A pesar de estos inconvenientes, ChatGPT tiene un tremendo potencial como una solución de bajo código para pruebas de automatización. El ChatGPT es un modelo de lenguaje natural muy poderoso con un enorme potencial. Tiene tremendas capacidades y probablemente se convertirá en el estándar para soluciones de prueba de bajo código para una amplia gama de problemas. Aunque tiene la capacidad de hacerlo para las pruebas, el uso de ChatGPT como está todavía requiere que el usuario tenga una comprensión básica tanto del idioma que se utiliza y la aplicación que se está probando. Sin embargo, no debemos subestimar el potencial de ChatGPT porque puede generar código impresionante que no era alcanzable con modelos anteriores de generación de lenguaje natural.

**Referencias**

[plataforma abierta API](https://platform.openai.com/) : Esto proporcionará información detallada sobre chatGPT

[Automatización de pruebas : Esto dará información sobre la automatización de pruebas](https://www.tutorialspoint.com/software_testing_dictionary/test_automation.htm)