МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. Н. КАРАЗІНА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК

КАФЕДРА БЕЗПЕКИ ІНФОРМАЦІЙНИХ СИСТЕМ І ТЕХНОЛОГІЙ

Лабораторна робота №12

з навчальної дисципліни

«Математичні методи та технології тестування та верифікації програмного забезпечення»

Виконала:

Студентка групи КС-23

Рузудженк С.Р.

Перевірив:

Доцент
Нарєжній О. П.

Харків – 2019

Лабораторна робота №12

Тема: «Тестування API (Application programming interface)»

Мета: вивичити тестування API, написати запити до веб-ресурсу.

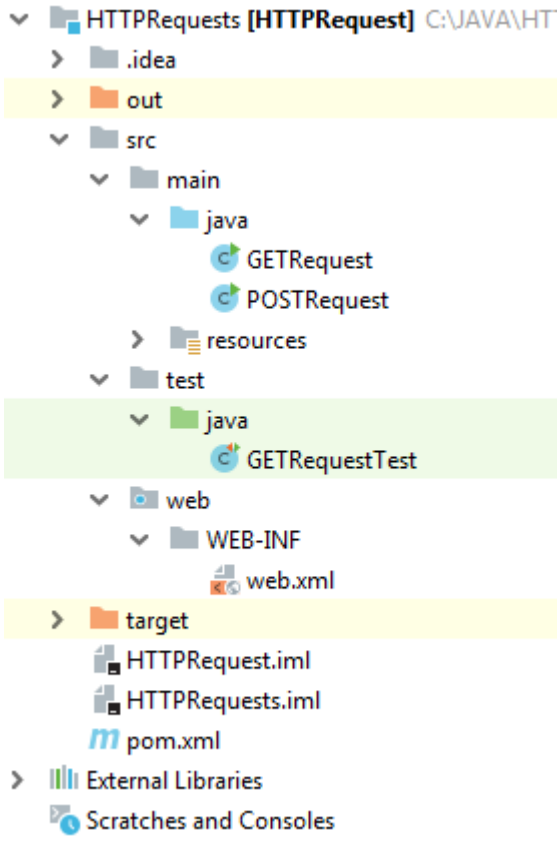
Хід роботи

Створюємо проект Maven, додаємо залежності у файл pom.xml.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>8</source>
                <target>8</target>
            </configuration>
        </plugin>
    </plugins>
</build>

<dependencies>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>2.11.1</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>2.11.1</version>
    </dependency>

    <dependency>
```

```
        <groupId>com.google.code.gson</groupId>
        <artifactId>gson</artifactId>
        <version>2.8.5</version>
        <scope>compile</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.apache.httpcomponents/httpclient
-->
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.3</version>
    </dependency>

</dependencies>
```

Проект має наступну структуру:



Створюємо клас GetRequest, прописуємо необхідний функціонал.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class GETRequest {

    private final static Logger LOG = LogManager.getLogger(GETRequest.class);
    private HttpURLConnection connection;

    public static void main(String[] args) throws IOException {
        GETRequest request = new GETRequest();
        String query = "http://localhost:8080/EE_war_exploded/";
        LOG.info(request.createGETRequest(query));
    }

    public String createGETRequest(String query) throws IOException {
        StringBuilder builder = new StringBuilder();

        connection = (HttpURLConnection) new URL(query).openConnection();
        connection.setRequestMethod("GET");
        connection.setUseCaches(false);
        connection.setConnectTimeout(2500);
        connection.setReadTimeout(2500);
        connection.connect();

        if (checkResponseCode()) {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                builder.append(line);
                builder.append("\n");
            }
        } else {
            builder = new StringBuilder(); // пустая строка
        }
        return builder.toString();
    }

    public boolean checkResponseCode() throws IOException {
        LOG.info(HttpURLConnection.HTTP_OK == connection.getResponseCode() ? "OK" :
"ERROR.. code = " + connection.getResponseCode());
        return HttpURLConnection.HTTP_OK == connection.getResponseCode();
    }
}
```

Далі створюємо клас для PostRequest, також додаємо необхідний функціонал класу.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
```

```
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;

public class POSTRequest {

    private final static Logger LOG = LogManager.getLogger(POSTRequest.class);

    public static void main(String[] args) {
        String query = "http://localhost:8080/index.jsp";
        HashMap<String, String> postDataParams = new HashMap<>();
        postDataParams.put("6", "6");
        postDataParams.put("-", "-");
        postDataParams.put("7", "7");
        postDataParams.put("%3D", "%3D");
        LOG.info(performPostCall(query, postDataParams));
    }

    private static String performPostCall(String requestURL, HashMap<String, String>
postDataParams) {
        URL url;
        StringBuilder response = new StringBuilder();
        try {
            url = new URL(requestURL);

            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.setReadTimeout(15000);
            connection.setConnectTimeout(15000);
            connection.setRequestMethod("POST");
            connection.setDoInput(true);
            connection.setDoOutput(true);

            OutputStream outputStream = connection.getOutputStream();
            BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter(outputStream, StandardCharsets.UTF_8));
            writer.write(getPostDataString(postDataParams));
            writer.flush();
            writer.close();
            outputStream.close();

            int responseCode = connection.getResponseCode();
            LOG.info((responseCode == 200) ? "OK" : "ERROR.. code = " + responseCode);
            if (responseCode == HttpURLConnection.HTTP_OK) {
                String line;
                BufferedReader reader = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
                while ((line = reader.readLine()) != null) {
                    response.append(line);
                    response.append("\n");
                }
            } else {
                response = new StringBuilder();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return response.toString();
```

```
    }

    /* Формирует параметры в строковый вид */
    private static String getPostDataString(HashMap<String, String> params) throws
UnsupportedEncodingException {
        StringBuilder result = new StringBuilder();
        boolean isFirst = true;
        for (Map.Entry<String, String> entry : params.entrySet()) {
            if (isFirst) isFirst = false;
            else result.append("&");

            result.append(URLEncoder.encode(entry.getKey(), "UTF-8"));
            result.append("=");
            result.append(URLEncoder.encode(entry.getValue(), "UTF-8"));
        }
        return result.toString();
    }
}
```

А також створюємо запускаючий клас GetRequestTest для тестування нашої програми за допомогою JUnit.

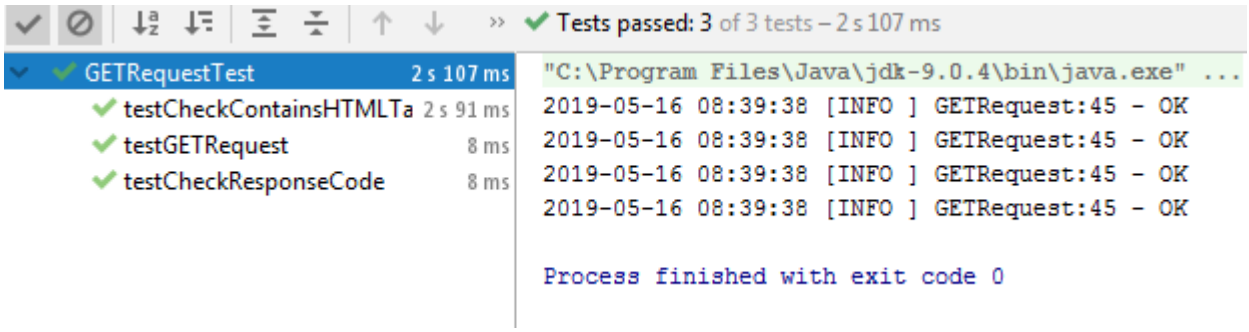
```
import org.junit.Assert;
import org.junit.Test;

import java.io.IOException;

public class GETRequestTest {

    private GETRequest request = new GETRequest();
    private String query = "http://localhost:8080/EE_war_exploded/";

    @Test
    public void testCheckContainsHTMLTag() throws IOException {
        String result = request.createGETRequest(query);
        Assert.assertTrue(result.contains("<html>"));
    }

    @Test
    public void testCheckResponseCode() throws IOException {
        request.createGETRequest(query);
        Assert.assertTrue(request.checkResponseCode());
    }

    @Test
    public void testGETRequest() throws IOException {
        Assert.assertFalse(request.createGETRequest(query).isEmpty());
    }

}
```

Файл *web.xml* має вигляд:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

</web-app>
```

Запускаємо проект лабораторної роботи № 11, а також класс GetRequestTest.

Можемо побачити, що усі тести пройшли успішно.



Висновки

Отже, API (Application Programming Interface) - програмний інтерфейс програми API є посередником між розробником додатків і будь-якої середовищем, з якої цей додаток повинен взаємодіяти. API спрощує створення коду, оскільки надає набір готових класів, функцій або структур для роботи з наявними даними.

Сучасні API часто приймають форму веб-сервісів, які надають користувачам (як людям, так і іншим веб-сервісів) якусь інформацію. Зазвичай процедура обміну інформацією і формат передачі даних структуровані, щоб обидві сторони знали, як взаємодіяти між собою.

Зазвичай при зверненні до веб API використовуються запити HTTP. Існують стандартні методи, які можуть міститися в HTTP запиті. Ці методи також називають HTTP дієсловами:

GET. Напевно, самий популярний тип запиту. Використовується для отримання або читання даних.
PUT. Зазвичай використовується для поновлення ресурсу.
POST. Зазвичай використовується для створення нового ресурсу.
DELETE. Видаляє дані.
Таким чином, у ході даної лабораторної роботи було вивчено тестування API, написані doGet та doPOst запити, а також програма була протестована за допомогою JUnit.