

[ASSIGNMENT 3] [STP304X_01 KIỂM THỬ TỰ ĐỘNG]

Họ và tên : Phạm Hải Thanh
MSV : FX20171

TP HCM.2023

MỤC LỤC

KIỂM TRA PHẢN HỒI API - CÁCH 2	3
PHẦN 1: TẠO PROJECT, PACKAGE CHO ASM3	4
PHẦN 2: TẠO CÁC THƯ MỤC VÀ VIẾT CÁC CÀI ĐẶT BỔ TRỢ	7
PHẦN 3: TẠO CÁC PACKAGES VÀ CLASSES TRONG PROJECT	8
3.1 Tạo package com.automation.testcase	8
3.1.1 Chứa class TC_API_Login_WithJson.java	8
3.1.2 Chứa class TC_API_CreateWork_WithJson.java	10
PHẦN 4: KẾT QUẢ CHẠY CHƯƠNG TRÌNH	13
4.1 Trường hợp TC_API_Login_WithJson.java	13
4.2 Trường hợp TC_API_CreateWork_WithJson.java	14

KIỂM TRA PHẢN HỒI API - CÁCH 2

Giải thích: Ở cách làm này, chúng ta sẽ sử dụng file dạng JSON trong folder **RequestPayload** để tiến hành lưu thông tin user, password, lưu token.

Trình tự làm như sau:

- *Đối với yêu cầu đăng nhập vào tài khoản:*

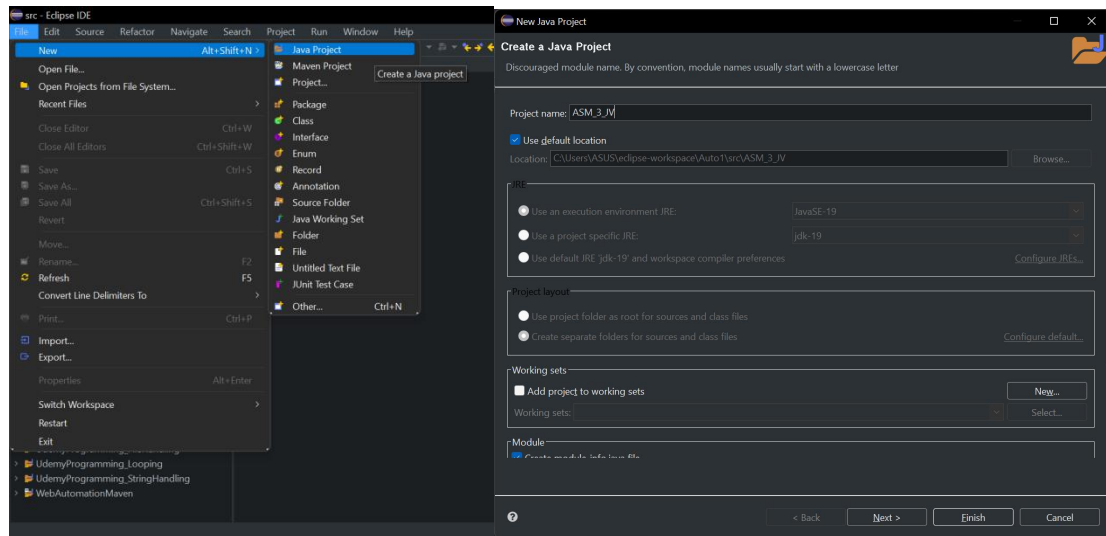
- B1. Lưu các thông tin account, password vào 1 file dạng JSON tên là "AddUserLoginRequest.json" (thuộc folder RequestPayload)
- B2. Lấy các thông tin cần thiết để đăng nhập account, password từ file AddUserLoginRequest.json để sử dụng
- B3. Lấy dữ liệu phản hồi và so sánh với các yêu cầu đề ra
- B4. Lấy dữ liệu token (nếu có) ghi vào file tên là "token.json" (thuộc folder RequestPayload)

- *Đối với tạo thông tin công việc mới cho tài khoản:*

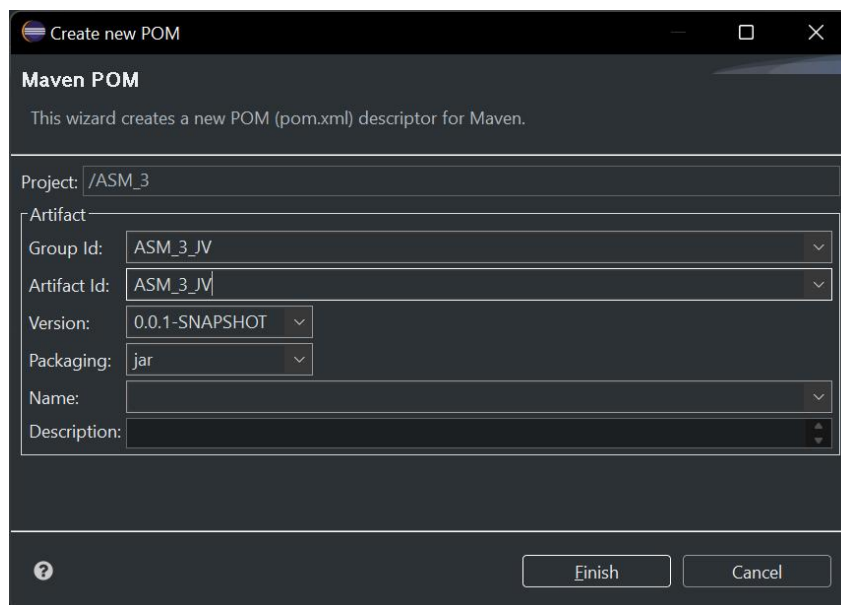
- B1. Lấy dữ liệu token (nếu có) ghi vào file tên là "token.json" (thuộc folder RequestPayload) ghi được ở bên trên để sử dụng làm header cho phần này
- B2. Tạo thông tin công việc mới và lưu thông tin này vào 1 file dạng JSON tên là "CreateWork.json" (thuộc folder RequestPayload)
- B3. Sử dụng thông tin trong file "CreateWork.json" để tạo công việc mới
- B4. Lấy dữ liệu phản hồi và so sánh với các yêu cầu đề ra

PHẦN 1: TẠO PROJECT, PACKAGE CHO ASM3

- B1: Chọn dạng dự án: Java Project



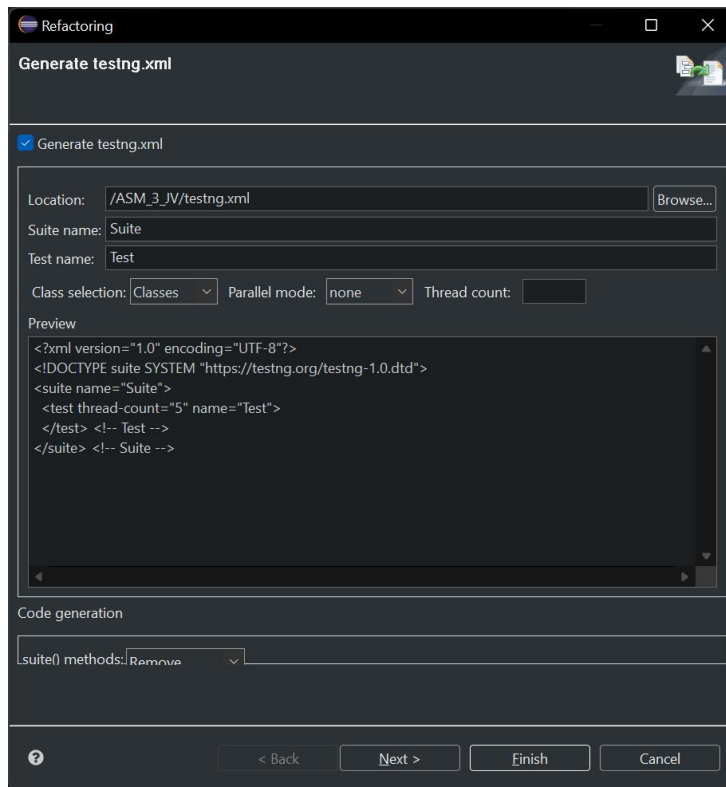
- B2: Convert sang Maven



- B3: Truy cập <https://mvnrepository.com/> để cài đặt các Dependencies vào pom.xml của project để sử dụng

1. Cài poi và poi-ooxml
<pre><dependency> <groupId>org.apache.poi</groupId> <artifactId>poi</artifactId> <version>5.2.3</version> </dependency></pre>
<pre><dependency> <groupId>org.apache.poi</groupId> <artifactId>poi-ooxml</artifactId> <version>5.2.3</version> </dependency></pre>
2. Cài testng
<pre><dependency> <groupId>org.testng</groupId> <artifactId>testng</artifactId> <version>7.4.0</version> <scope>test</scope> </dependency></pre>
3. Cài rest-assured và dependencies hỗ trợ đọc json
<pre><dependency> <groupId>io.rest-assured</groupId> <artifactId>rest-assured</artifactId> <version>4.4.0</version> <scope>test</scope> </dependency></pre>
<pre><dependency> <groupId>com.google.inject</groupId> <artifactId>guice</artifactId> <version>3.0</version> </dependency></pre>
<pre><dependency> <groupId>org.hamcrest</groupId> <artifactId>hamcrest</artifactId> <version>2.2</version> </dependency></pre>
<pre><dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> <version>2.13.4.2</version> </dependency></pre>
<pre><dependency> <groupId>org.apache.logging.log4j</groupId> <artifactId>log4j-core</artifactId> <version>2.20.0</version> </dependency></pre>

- B4: Convert sang TestNG



PHẦN 2: TẠO CÁC THƯ MỤC VÀ VIẾT CÁC CÀI ĐẶT BỔ TRỢ

Tạo các thư mục chứa các file ,drive bổ trợ

Tên folder	File con	Giải thích
RequestPayload	AddUserLoginRequest.json	Chứa các thông tin như tên user, password để đăng nhập
	CreateWork.json	Chứa các thông tin tạo mới công việc
	token.json	Chứa giá trị token lấy thành công, sau khi login thành công

Xác định và thêm thông tin vào AddUserLoginRequest.json và CreateWork.json

- AddUserLoginRequest.json chứa các thông tin như tên user, password để đăng nhập

```
{
  "account": "testerFunix",
  "password": "Abc13579"
}
```

- CreateWork.json chứa các thông tin để tạo công việc mới trong phần sau

```
{
  "nameWork": "Giang vien",
  "experience": "3",
  "education": "Thac si"
}
```

- token.json chứa giá trị token sau khi login, dùng cho phần CreateWork ở sau. File này để trống và sẽ được ghi tự động khi login thành công, lấy giá trị token và ghi thành công

PHẦN 3: TẠO CÁC PACKAGES VÀ CLASSES TRONG PROJECT

Tạo folder packages trong file ASM_3_JV: là src.test

Tên packages	Mục đích	Các class con
src.test		
com.automation.testcase	chứa các class test sử dụng TestNG	TC_API_Login_WithJson.java
		TC_API_CreateWork_WithJson.java

3.1 Tạo package com.automation.testcase

3.1.1 Chứa class TC_API_Login_WithJson.java

- Chứa class TC_API_Login_WithJson.java là nơi để thực hiện việc test API login. Chúng ta sẽ sử dụng dữ liệu đã cho là URL; lấy và đọc dữ liệu user, password từ file "AddUserLoginRequest.json" để đăng nhập và kiểm tra phản hồi như status code, các trường dữ liệu trong phản hồi

```
package com.api.auto.testcase.JSON;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

import io.restassured.RestAssured;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.response.ResponseBody;
import io.restassured.specification.RequestSpecification;

public class TC_API_Login_WithJson {
    String postEndPoint = "http://13.228.39.1:5000";
    String basePath = "/api/users/login";

    RequestSpecification request;
    Response postresponse;
    ResponseBody responseBody;
    JsonPath bodyJson;
    String jsonContent;
    private String token ;
    File jsonBody;
    FileWriter file;

    @BeforeMethod
    public void setConfiguration() throws IOException {

        String inputData = "";

        File jsonBody = new File("./RequestPayload/AddUserLoginRequest.json");
        FileReader fr = new FileReader(jsonBody);
```



```

BufferedReader br = new BufferedReader(fr);
jsonContent = br.readLine();

while (jsonContent != null) {
    inputData = inputData + jsonContent;
    jsonContent = br.readLine();
}

request = RestAssured.given();
request.baseUrl(postEndPoint);
request.basePath(basePath);

request.header("content-type", "application/json");
request.body(inputData);

postresponse = request.post();
resBody = postresponse.getBody();
bodyJson = resBody.jsonPath();

}
//THỰC HIỆN TESTCASE, CÀI ĐẶT THỨ TỰ ƯU TIÊN TEST NẾU CẦN
@Test
public void T01_StatusCodeTest() {
// Kiểm chứng status code có là 200 hay không
    assertEquals(200, postresponse.getStatusCode(), "Status Check Failed!");
    System.out.println(" " + postresponse.asPrettyString());
}

@Test
public void T02_MessageValidateChecked() {
// Kiểm chứng response body có chứa trường message hay không
    assertTrue(postresponse.asString().contains("message"), "message field check Failed!");
}

@Test
public void T03_MessageContentChecked() {
// Kiểm chứng trường message có hiển thị nội dung "Đăng nhập thành công" hay không
    String resMessage = bodyJson.get("message");
    assertEquals(resMessage, "Đăng nhập thành công", "message field check Failed!");
}

@Test(priority = 3)
public void TC04_ValidateToken() throws Exception {
// Kiểm chứng response body có chứa trường token hay không
    token = bodyJson.getString("token");
    assertTrue(postresponse.asString().contains("token"), "Token field check Failed!");
    FileWriter file = new FileWriter("./RequestPayload/token.json");
    file.write("{\"token\":\"" + token + "\"}");
    file.close();
}

@Test(priority = 4)
public void TC05_ValidateUser() {
// Kiểm chứng response chứa thông tin user hay không
    assertTrue(postresponse.asString().contains("user"), "User field check Failed!");
}

```

```

        @Test(priority = 5)
        public void TC06_ValidateUserType() {
// Kiểm chứng response body có chứa thông tin trường type hay không
        assertTrue(postresponse.asString().contains("type"), "Type field check Failed!");
        }

        @Test(priority = 6)
        public void TC07_VerifyOnMatchType() {
// Kiểm chứng trường type nội dung có phải là "UNGVIENT" hay không
        assertEquals(bodyJson.getString("user.type"),"UNGVIENT", "Type check Failed!");
        }

        @Test(priority = 7)
        public void TC08_ValidateAccount() {
// Kiểm chứng response body có chứa thông tin trường account hay không
        assertTrue(postresponse.asString().contains("account"), "account field check Failed!");
        }

        @Test(priority = 8)
        public void TC09_VerifyOnMatchAccount() {
// Kiểm chứng trường account có khớp với account đăng nhập đã cho hay không
        assertEquals(bodyJson.getString("user.account"), "testerFunix", "Account check Failed!");
        }

        @Test(priority = 9)
        public void TC10_ValidatePassword() {
// Kiểm chứng response chứa thông tin trường password hay không
        assertTrue(postresponse.asString().contains("password"), "Type field check Failed!");
        }

        @Test(priority = 10)
        public void TC11_VerifyOnMatchPassword() {
// Kiểm chứng trường password có khớp với password đăng nhập đã cho hay không
        assertEquals(bodyJson.getString("user.password"), "Abc13579", "Password check
Failed!");
        }
    }
}

```

3.1.2 Chứa class TC_API_CreateWork_WithJson.java

- Chứa class TC_API_CreateWork_WithJson.java là nơi để thực hiện việc tạo công việc. Chúng ta sẽ sử dụng dữ liệu token trong file token.json đã lấy được từ lúc login thành công để làm header và tạo dữ liệu. Sau khi tạo dữ liệu, nhận được phản hồi thì sẽ kiểm tra phản hồi như status code, các trường dữ liệu trong phản hồi

```

package com.api.auto.testcase.JSON;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

import org.testng.annotations.BeforeMethod;
import org.testng.annotations.Test;

```

```

import io.restassured.RestAssured;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.response.ResponseBody;
import io.restassured.specification.RequestSpecification;

import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

public class TC_API_CreateWork_WithJson {
    String postEndPoint = "http://13.228.39.1:5000";
    String createWorkPath = "/api/work-user/createWork-user";

    RequestSpecification request;
    Response postresponse;
    ResponseBody responseBody;
    JsonPath bodyJson;
    String jsonContent;
    File jsonBody;
    Object JsonObject;

    @BeforeMethod
    public void setConfiguration() throws IOException {
        request = RestAssured.given();

        File tokenFile = new File("./RequestPayload/token.json");
        FileReader fileReader = new FileReader(tokenFile);
        JsonParser jsonParser = new JsonParser();
        JsonObject jsonObject = jsonParser.parse(fileReader).getAsJsonObject();
        String tokenValue = jsonObject.get("token").getString();
// Cài đặt giá trị token vào header
        request.header("token", tokenValue);

// Đọc dữ liệu từ file "./RequestPayload/CreateWork.json"
        jsonBody = new File("./RequestPayload/CreateWork.json");
        String inputData = "";

        FileReader fr = new FileReader(jsonBody);
        BufferedReader br = new BufferedReader(fr);
        jsonContent = "";

        while (jsonContent != null) {
            inputData = inputData + jsonContent;
            jsonContent = br.readLine();
        }

        request.baseUrl(postEndPoint);
        request.basePath(createWorkPath);

        request.body(inputData);
        request.header("content-type", "application/json");

        postresponse = request.post();
        responseBody = postresponse.getBody();
        bodyJson = responseBody.jsonPath();
    }
}

```

```

//THỰC HIỆN TESTCASE, CÀI ĐẶT THỨ TỰ ƯU TIÊN TEST NẾU CẦN
@Test(priority = 0)
public void TC01_Validate201Created() {
// Kiểm chứng status code có là 201 hay không
    assertEquals(postresponse.getStatusCode(), 201);
    System.out.println("\nStatus code is: " + postresponse.getStatusCode()+"\n");
    System.out.println(" " + postresponse.asPrettyString());
}

@Test(priority = 1)
public void TC02_ContainWorkId() {
// Kiểm chứng response body có chứa trường id hay không
    assertTrue(postresponse.asString().contains("id"), "Id field check Failed!");
}

@Test(priority = 2)
public void TC03_ContainNameOfWork() {
// Kiểm chứng response body có chứa trường nameWork hay không
    assertTrue(postresponse.asString().contains("nameWork"), "nameWork field check Failed!");
}

@Test(priority = 3)
public void TC04_ValidateNameOfWorkMatched() {
// Kiểm chứng trường nameWork có khớp với thông tin đã tạo hay không
    assertEquals(bodyJson.getString("nameWork"), "Giang vien", "nameWork check Failed!");
}

@Test(priority = 4)
public void TC05_ContainExperienceOfWork() {
// Kiểm chứng response body có chứa trường experience hay không
    assertTrue(postresponse.asString().contains("experience"), "experience field check Failed!");
}

@Test(priority = 5)
public void TC06_ValidateExperienceMatched() {
// Kiểm chứng trường experience có khớp với thông tin myExperience đã tạo hay không
    assertEquals(bodyJson.getString("experience"), "3", "experience check Failed!");
}

@Test(priority = 6)
public void TC07_ContainEducationOfUser() {
// Kiểm chứng response body có chứa trường education hay không
    assertTrue(postresponse.asString().contains("education"), "education field check Failed!");
}

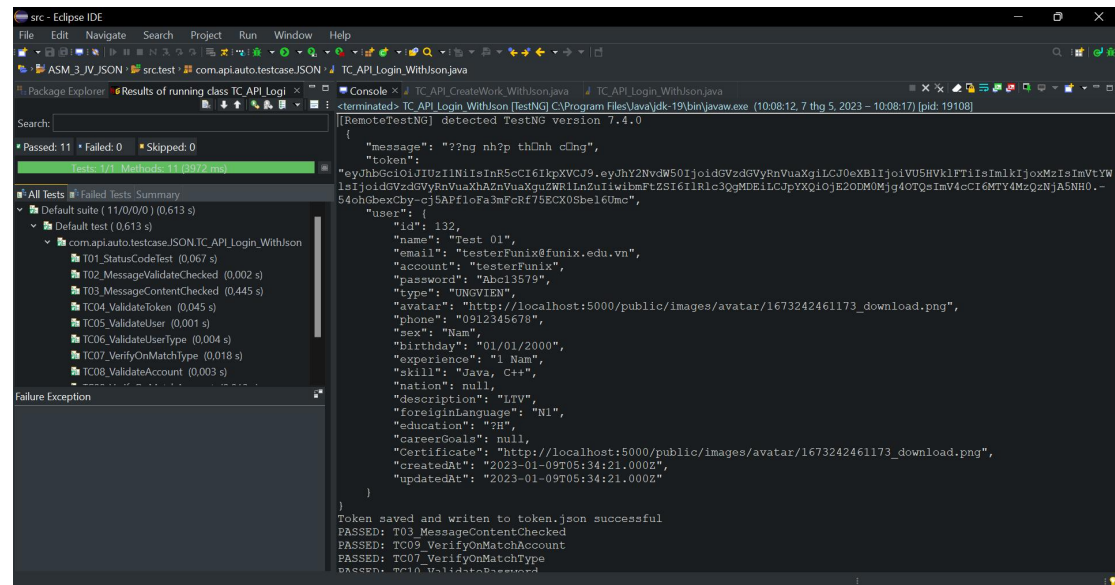
@Test(priority = 7)
public void TC08_ValidateEducationMatched() {
// Kiểm chứng trường education có khớp với thông tin myEducation đã tạo hay không
    assertEquals(bodyJson.getString("education"), "Thac si", "education check Failed!");
}
}

```

PHẦN 4: KẾT QUẢ CHẠY CHƯƠNG TRÌNH

4.1 Trường hợp TC_API_Login_WithJson.java

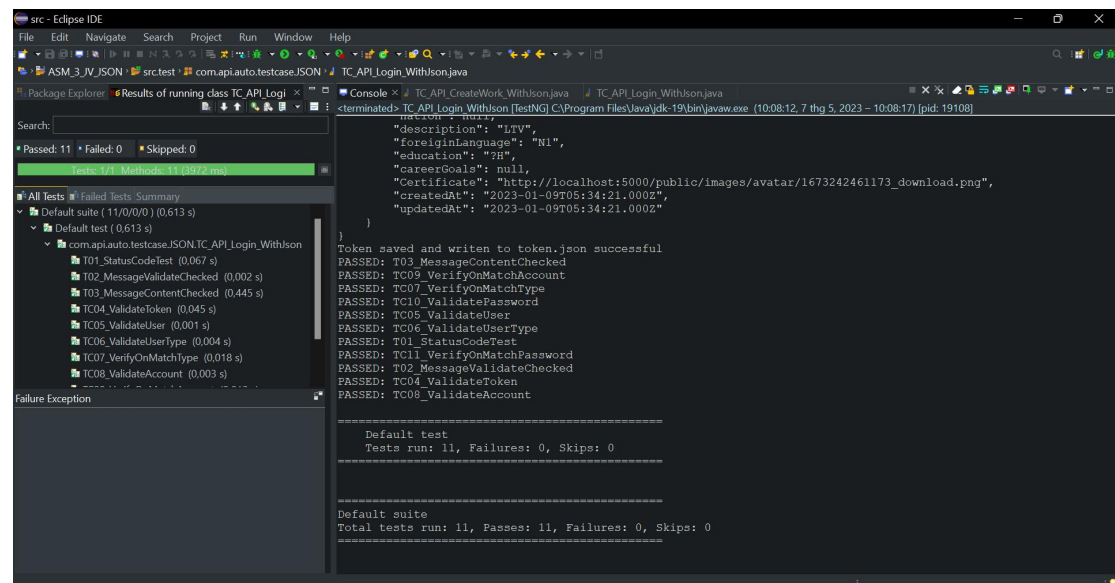
Khi chạy chương trình sẽ in ra các thông tin hiển thị của phản hồi sau khi login thành công như bên dưới.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure with 'TC_API_Login_WithJson.java' selected. The Console window on the right shows the output of the test run, including a JSON response from the API and a success message. The Test Results window on the left shows that all tests passed.

```
Token saved and written to token.json successful
PASSED: T03_MessageContentChecked
PASSED: TC09_VerifyOnMatchAccount
PASSED: TC07_VerifyOnMatchType
PASSED: TC10_ValidatePassword
```

Token được ghi thành công cũng sẽ được thông báo



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure with 'TC_API_Login_WithJson.java' selected. The Console window on the right shows the output of the test run, including a JSON response from the API and a success message. The Test Results window on the left shows that all tests passed.

```
Token saved and written to token.json successful
PASSED: T03_MessageContentChecked
PASSED: TC09_VerifyOnMatchAccount
PASSED: TC07_VerifyOnMatchType
PASSED: TC10_ValidatePassword
PASSED: TC05_ValidateUser
PASSED: TC06_ValidateUserType
PASSED: T01_StatusCodeTest
PASSED: TC11_VerifyOnMatchPassword
PASSED: T02_MessageValidateChecked
PASSED: TC04_ValidateToken
PASSED: TC08_ValidateAccount
```

4.2 Trường hợp TC_API_CreateWork_WithJson.java

