

# 2025

nguyen.a.tu@gmail.com

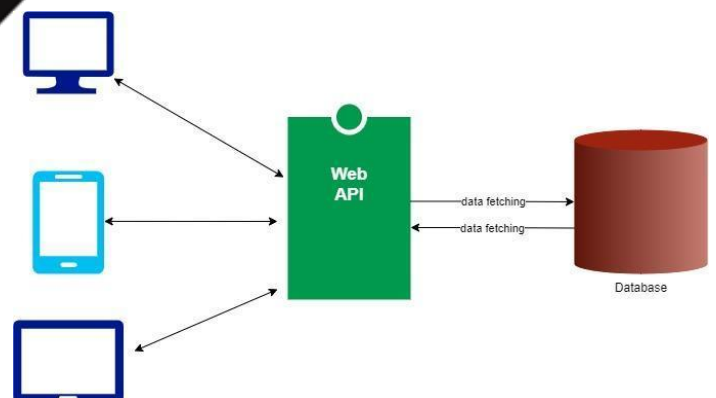
www.facebook.com/nguyenanhtucom

www.youtube.com/@nguyenanhtucom

## EXERCISE

# Internet of Things

## PROGRAMMING



## MỤC LỤC

<b>Setup Development Environment</b> .....	1
Example 0.01 .....	1
<b>Hibernate</b> .....	<b>Error! Bookmark not defined.</b>
Example 1.01 .....	1
Example 1.02 .....	8
Example 1.03 .....	1
<b>Spring Data JPA</b> .....	9
Example 2.01 .....	9
Example 2.02 .....	19
Example 2.03 .....	27
Example 2.04 .....	43
Example 2.05 .....	58
Example 2.06 .....	69
Exercise 2.01 .....	75
<b>Spring Security</b> .....	95
Example 3.01 .....	95
Basic Authentication.....	96
Integration with the database .....	108
Example 3.02 .....	118
Basic Authentication.....	119
Integration with the database .....	129
Example 3.03 .....	138
Integration with the database .....	139
Authentication.....	147
Example 3.04 .....	164
Integration with the database .....	165
<b>ReactJS</b> .....	205
Example 4.01 .....	205
Example 4.02 .....	210
Example 4.03 .....	228
Example 4.04 .....	253
Example 4.05 .....	286
<b>Course Project 01</b> .....	333
Requirement.....	333

Instructions .....	335
Backend .....	335
Front end shop .....	337
Front end administration.....	338
<b>Microservices</b> .....	340
Example 5.01 .....	340
Eureka Server.....	341
API Gateway .....	343
Product Service.....	345
Order Service .....	350
Inventory Service .....	356
Run the Services .....	361
Test API.....	361
Example 5.02 .....	362
Eureka Server.....	364
API Gateway .....	366
Product Service.....	373
Order Service .....	386
Product Recommendation Service .....	414
User Service .....	430
Run the Services .....	449
Test API.....	449
Example 5.03 .....	450
Common .....	452
Advanced Message Queuing Protocol (AMQP).....	459
Eureka Server.....	467
API Gateway .....	471
Product Service.....	480
Order Service .....	525
Inventory Service .....	552
File Service .....	567
Notification Service .....	579
Cart Service .....	591
User Service .....	602
Run the Services .....	657

---

Test API.....	658
---------------	-----

# Setup Development Environment

## Example 0.01

**Mục tiêu:** Thiết lập môi trường lập trình Java Spring Boot và tạo project

**Yêu cầu:**

- ✓ Cài đặt JDK và Visual Studio Code
- ✓ Tạo project có tên project là **example01**

**Hướng dẫn:**

**Bước 1:** Cài đặt JDK và Visual Studio Code

### JDK Development Kit 17.0.7 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions.

JDK 17 will receive updates under these terms, until September 2024, a year after the release of the next LTS.

Linux macOS Windows

Product/file description	File size	Download
x64 Compressed Archive	172.19 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip</a> ( sha256)
x64 Installer	153.28 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.exe</a> ( sha256)
x64 MSI Installer	152.07 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.msi</a> ( sha256)

**Bước 2:** Để cài đặt Extension Java cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **java**



### Extension Pack for Java

v0.25.12

Preview

Microsoft [microsoft.com](https://www.microsoft.com) | 20,296,269 | ★★★★★ (62)

Popular extensions for Java development that provides Java IntelliSense, debugging, testing, Maven/Gradle support, project management and more

Disable

Uninstall

Switch to Pre-Release Version



This extension is enabled globally.

**Bước 3:** Để cài đặt Extension Spring Boot cho Visual Studio Code bạn hãy thực hiện Mở Extensions (Ctrl+Shift+X), tìm kiếm **Spring Boot Extension Pack**



### Spring Boot Extension Pack

v0.2.1

VMware [vmware.com](https://www.vmware.com) | 1,643,692 | ★★★★★ (15)

A collection of extensions for developing Spring Boot applications

Installing



**Bước 4:** Sử dụng Visual Studio Code tạo project **Maven** với thông tin như sau:

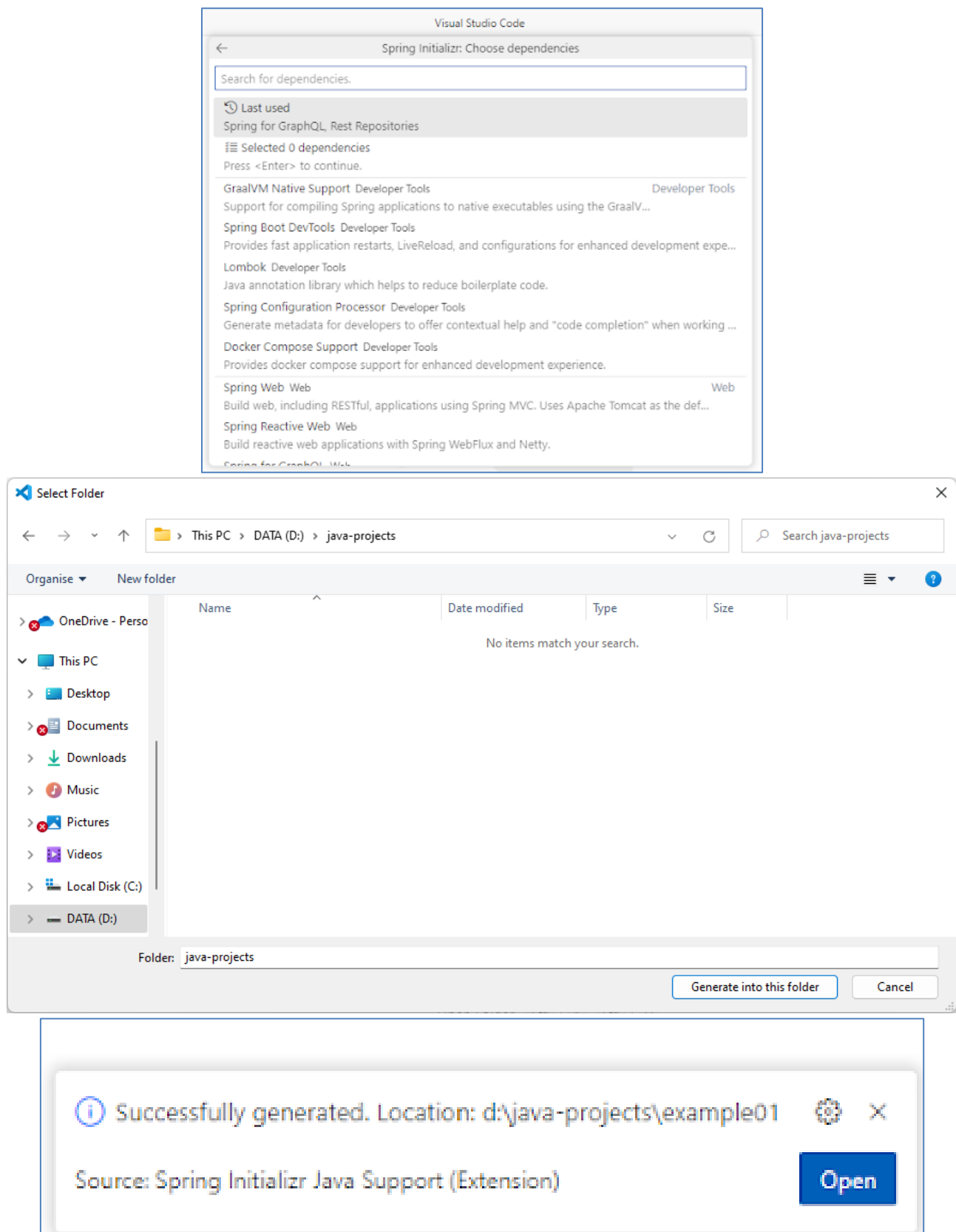
- ✓ Spring Boot version: **3.x.x**
- ✓ Language: **Java**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Artifact Id: **example01**
- ✓ Packaging type: **Jar**
- ✓ Java Version: **17**
- ✓ Dependencies: **Selected 0 dependencies**

✓ Vị trí project: **D:\java-projects**

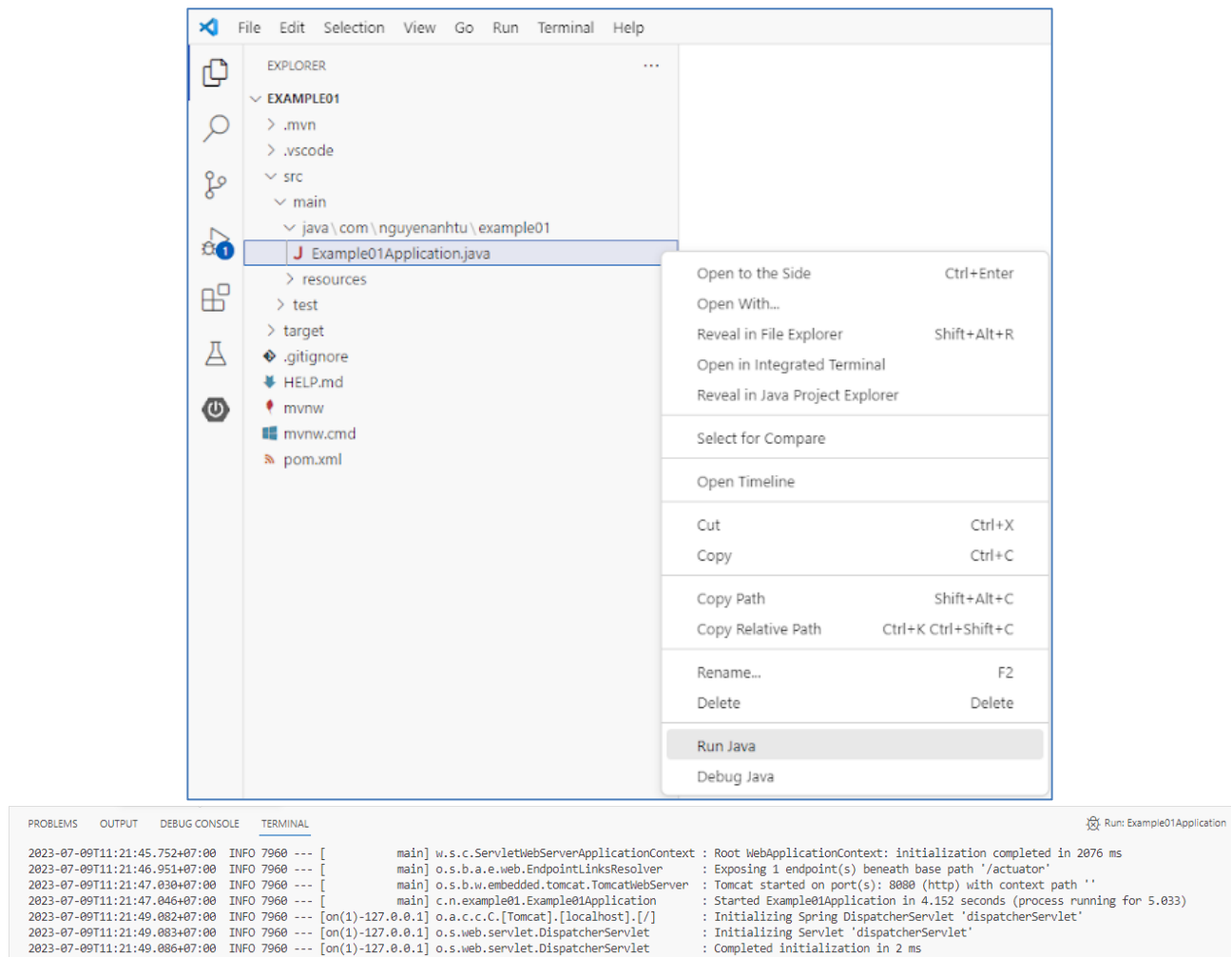
Hãy mở Bảng lệnh (Ctrl+Shift+P) và nhập **Spring Initializr** để bắt đầu tạo dự án **Maven**

The following steps illustrate the Spring Initializr wizard configuration:

- Step 1:** Command palette search results for `>Spring Initializr`.
  - Spring Initializr: Create a Maven Project... (recently used)
  - Spring Initializr: Add Starters...
  - Spring Initializr: Create a Gradle Project...
- Step 2:** Spring Initializr: Specify Spring Boot version.
  - Specify Spring Boot version.
  - 3.1.1 (selected)
  - 3.2.0 (SNAPSHOT)
  - 3.1.2 (SNAPSHOT)
  - 3.0.9 (SNAPSHOT)
  - 3.0.8
  - 2.7.14 (SNAPSHOT)
  - 2.7.13
- Step 3:** Spring Initializr: Specify project language.
  - Specify project language.
  - Java (selected)
  - Kotlin
  - Groovy
- Step 4:** Spring Initializr: Input Group Id.
  - com.nguyenanhtu
  - Input Group Id for your project. (Press 'Enter' to confirm or 'Escape' to cancel)
- Step 5:** Spring Initializr: Input Artifact Id.
  - example01
  - Input Artifact Id for your project. (Press 'Enter' to confirm or 'Escape' to cancel)
- Step 6:** Spring Initializr: Specify packaging type.
  - Specify packaging type.
  - Jar (selected)
  - War
- Step 7:** Spring Initializr: Specify Java version.
  - Specify Java version.
  - 17 (selected)
  - 20
  - 11
  - 8



**Bước 5:** Click chuột phải vào lớp **Example01Application** sau đó Click vào **Run Java**.





# IoT Broker

## Example 1.01

**Mục tiêu:** Tạo và quản lý ứng dụng **Java** sử dụng **MQTT Paho** kết nối **MQTT Broker** có tên kiến trúc như sau:



**Yêu cầu:** Ứng dụng **Java** Thực hiện các chức năng sau:

- ✓ Subscribe với **Broker** với topic có tên `/test/topic`
- ✓ Publish nội dung "Hi from the IoT application" đến **Broker** với topic có tên `/test/topic`

**Hướng dẫn:**

**Bước 1:** Sử dụng Visual Studio Code, mở Command Palette (Ctrl+Shift+P) và nhập **Java: Create Java Project...** để bắt đầu tạo project với các thông tin như sau:

- ✓ Select the project type: **Maven**
- ✓ Vị trí project: **D:\java-projects**
- ✓ Artifact Id: **example101**
- ✓ Group Id: **com.nguyenanhtu**
- ✓ Select an archetype: **No Archetype**

**Bước 2:** Sửa file pom.xml như sau:**pom.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.nguyenanhtu</groupId>
  <artifactId>example102</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>org.eclipse.paho</groupId>
      <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
      <version>1.2.5</version>
    </dependency>
  </dependencies>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>
</project>
```

**Bước 4:** Tao Entity **Employee** như sau:**Main.java**

```
package com.nguyenanhtu.example101;

import java.io.IOException;

import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttClient;

public class Main {
    public static void main(String[] args) {
        String mqttBroker = "tcp://localhost:1883";
        String mqttTopic = "/test/topic";
        String username = "IoTClient";
        String password = "IoTPass";
        String testMsg = "Hi from the IoT application";
        int qos = 1;

        try {
            MqttClient mqttClient = new MqttClient(mqttBroker, "mqttClient");
            MqttConnectOptions mqttOptions = new MqttConnectOptions();
            mqttOptions.setUserName(username);
            mqttOptions.setPassword(password.toCharArray());
            mqttClient.connect(mqttOptions);

            if (mqttClient.isConnected()) {
                mqttClient.setCallback(new MqttCallback() {
                    @Override
```

```
        public void messageArrived(String topic, MqttMessage message) throws Exception {
            System.out.println("Received message: " + new String(message.getPayload()));
        }

        @Override
        public void connectionLost(Throwable cause) {
            System.out.println("Connection is lost: " + cause.getMessage());
        }

        @Override
        public void deliveryComplete(IMqttDeliveryToken token) {
            System.out.println("Message publish is complete: " + token.isComplete());
        }
    });

    /* The client subscribe to a topic */
    mqttClient.subscribe(mqttTopic, qos);
    /* Preparing a message to be published */
    MqttMessage mqttMsg = new MqttMessage(testMsg.getBytes());
    mqttMsg.setQos(qos);
    /* A message is published on the same subscribed topic */
    mqttClient.publish(mqttTopic, mqttMsg);
}

/* Keep the application open, so that the subscribe operation can tested */
System.out.println("Press Enter to disconnect");
System.in.read();
/* Proceed with disconnecting */
mqttClient.disconnect();
mqttClient.close();

} catch (MqttException e) {
    e.printStackTrace();
}
```

```
} catch (IOException e) {  
    throw new RuntimeException(e);  
}  
}  
}
```

**Bước 3:** Sử dụng **MQTT-Explorer** đăng ký topic có tên **/test/topic** với Broker

The screenshot displays the MQTT Explorer web interface. On the left, a sidebar shows a list of connections: 'example101' (selected) and 'test.mosquitto.org'. The main panel is titled 'MQTT Connection' and shows the configuration for 'example101'. The connection URL is 'mqtt://localhost:1883/'. The interface includes fields for 'Name' (example101), 'Protocol' (mqtt://), 'Host' (localhost), and 'Port' (1883). There are also toggle switches for 'Validate certificate' and 'Encryption (tls)'. At the bottom, there are input fields for 'Username' and 'Password', and buttons for 'DELETE', 'ADVANCED', 'SAVE', and 'CONNECT'. Four red callout boxes with numbers 1, 2, 3, and 4 point to the 'Name' field, 'Host' field, 'Port' field, and 'ADVANCED' button respectively.

Connections

example101  
mqtt://localhost:1883/

test.mosquitto.org  
mqtt://test.mosquitto.org:1883/

MQTT Connection mqtt://localhost:1883/

Name example101

Validate certificate

Encryption (tls)

Protocol mqtt://

Host localhost

Port 1883

Username

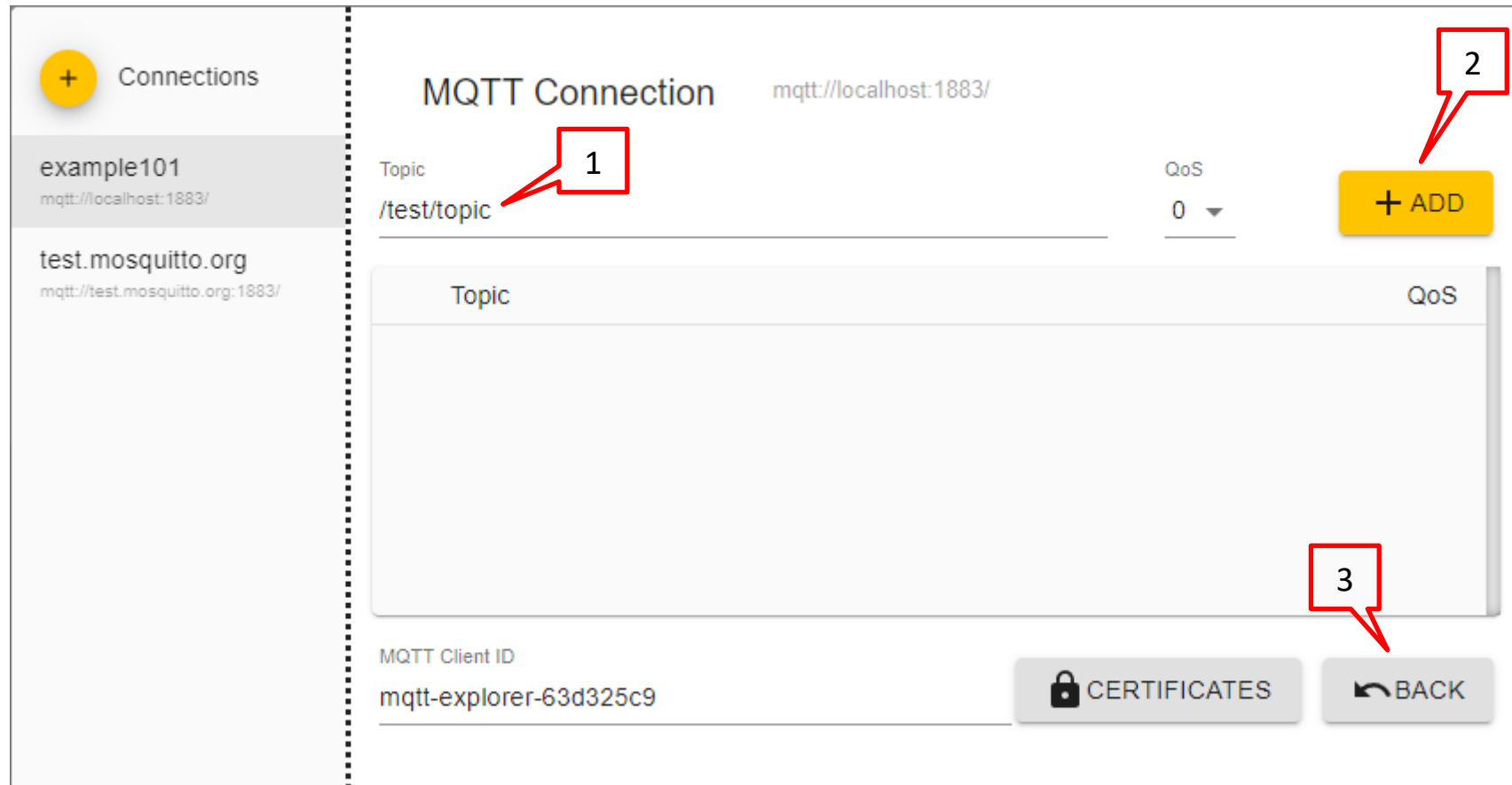
Password

DELETE

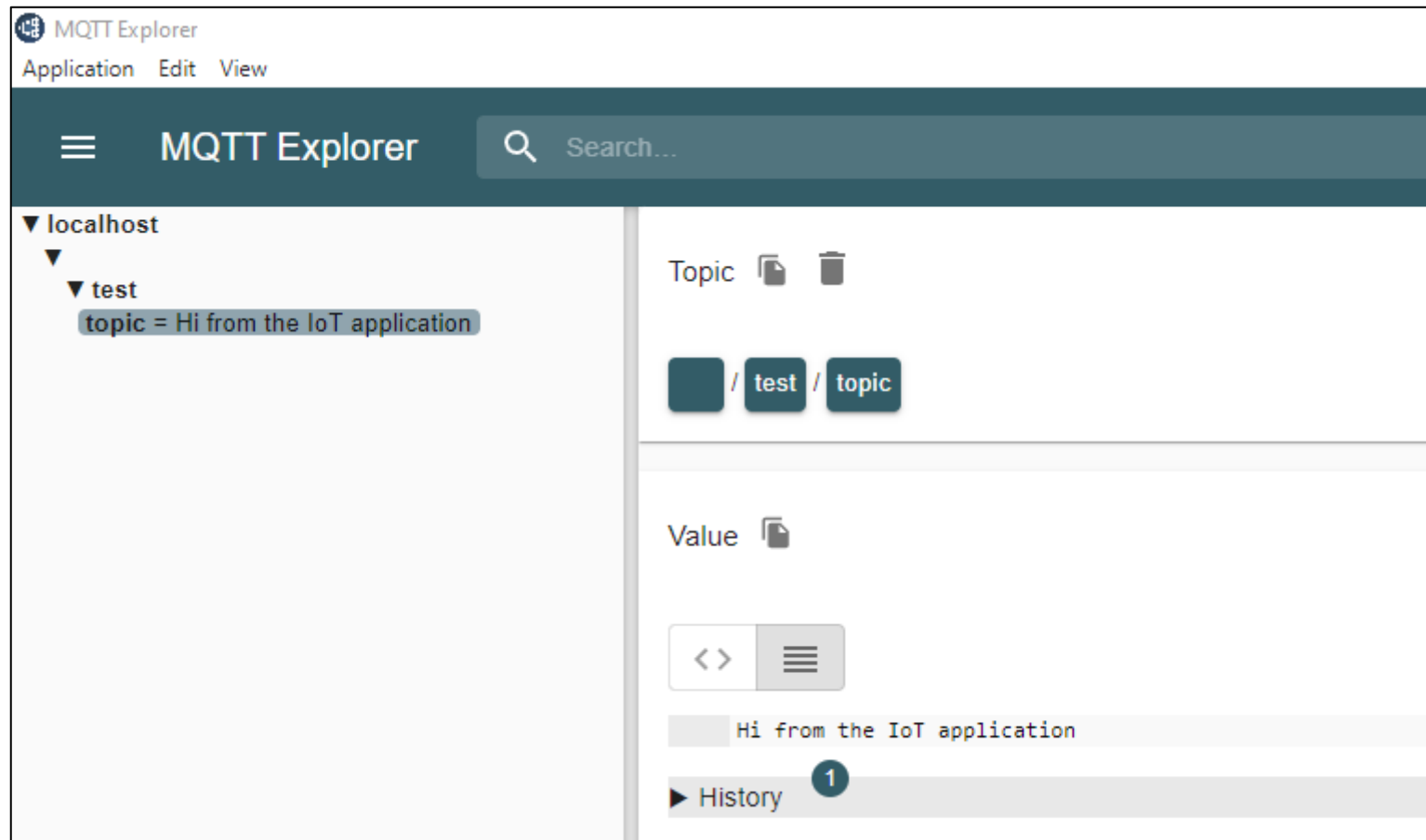
ADVANCED

SAVE

CONNECT



**Bước 4:** Click chuột phải vào lớp **Main** → **Run Java**, sau đó quan sát ứng dụng **MQTT-Explorer** như sau:



**Bước 4:** Sử dụng **MQTT-Explorer** publish tới Broker nội dung **Hello IoT Hutech** với topic có tên **/test/topic**



**Bước 5:** Kiểm tra dữ liệu nhận được trong cửa sổ Terminal của Visual Code

```
Press Enter to disconnect
Received message: Hi from the IoT application
Message publish is complete: true
Received message: Hello IoT Hutech
```