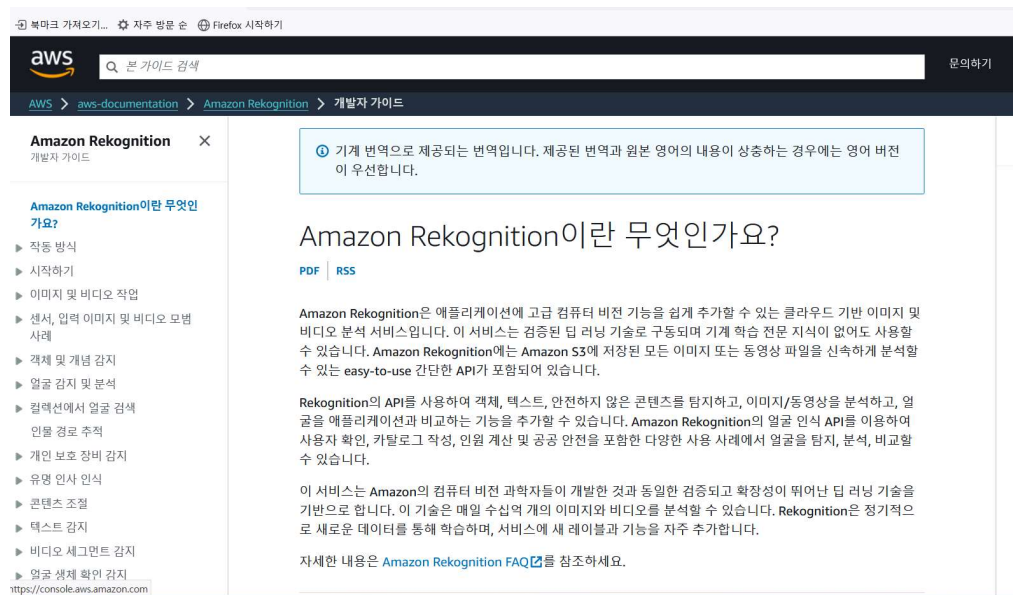



# Amazon Rekognition

- [https://docs.aws.amazon.com/ko\\_kr/rekognition/latest/dg/what-is.html](https://docs.aws.amazon.com/ko_kr/rekognition/latest/dg/what-is.html)



# 예제



[AWS](#) > [aws-documentation](#) > [Amazon Rekognition](#) > [개발자 가이드](#)

대량 분석

- 자습서
- 코드 예시
  - 작업
    - 이미지 안의 얼굴을 참조 이미지와 대조
    - 컬렉션 생성
    - 컬렉션 삭제
    - 컬렉션에서 얼굴 삭제
    - 컬렉션 설명
    - 이미지에서 얼굴 감지
    - 이미지에서 레이블 감지
    - 이미지에서 조정 레이블 감지**
    - 이미지에서 텍스트 감지
    - 유명인 정보 가져오기
    - 컬렉션에 얼굴 인덱싱
    - 컬렉션 나열
    - 컬렉션에서 얼굴 나열
    - 이미지에서 유명인 인식
    - 컬렉션에서 얼굴 검색

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 조정 레이블을 감지하는 방법을 보여줍니다. 조정 레이블은 일부 대상에게 절할 수 있는 콘텐츠를 식별합니다.

자세한 내용은 [부적절한 이미지 감지](#)를 참조하세요.

.NET

**CLI**

Java

Kotlin

Python

### AWS CLI

이미지에서 안전하지 않은 콘텐츠를 탐지하려면


다음 `detect-moderation-labels` 명령은 Amazon S3 버킷에 저장된 지정된 이미지에서 안전하지 않은 콘텐츠를 탐지합니다.

```
aws rekognition detect-moderation-labels \
  --image "S3Object={Bucket=MyImageS3Bucket,Name=gun.jpg}"
```

출력:

```
{
  "ModerationModelVersion": "3.0",
  "ModerationLabels": [
    {
      "Confidence": 97.29618072509766,
      "ParentName": "Violence",
```

# 예제



본 가이드 검색

문의하기 한국어 ▼

AWS > aws-documentation > Amazon Rekognition > 개발자 가이드

대량 분석

▶ 자습서

▼ 코드 예시

▼ 작업

이미지 안의 얼굴을 참조 이미지와 대조

컬렉션 생성

컬렉션 삭제

컬렉션에서 얼굴 삭제

컬렉션 설명

이미지에서 얼굴 감지

이미지에서 레이블 감지

**이미지에서 조정 레이블 감지**

이미지에서 텍스트 감지

유명인 정보 가져오기

컬렉션에 얼굴 인덱싱

컬렉션 나열

컬렉션에서 얼굴 나열

이미지에서 유명인 인식

컬렉션에서 얼굴 검색

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 조정 레이블을 감지하는 방법을 보여줍니다. 조정 레이블은 일부 대상에  
적용할 수 있는 콘텐츠를 식별합니다.

자세한 내용은 [부적절한 이미지 감지](#)를 참조하세요.

.NET

CLI

**Java**

Kotlin

Python

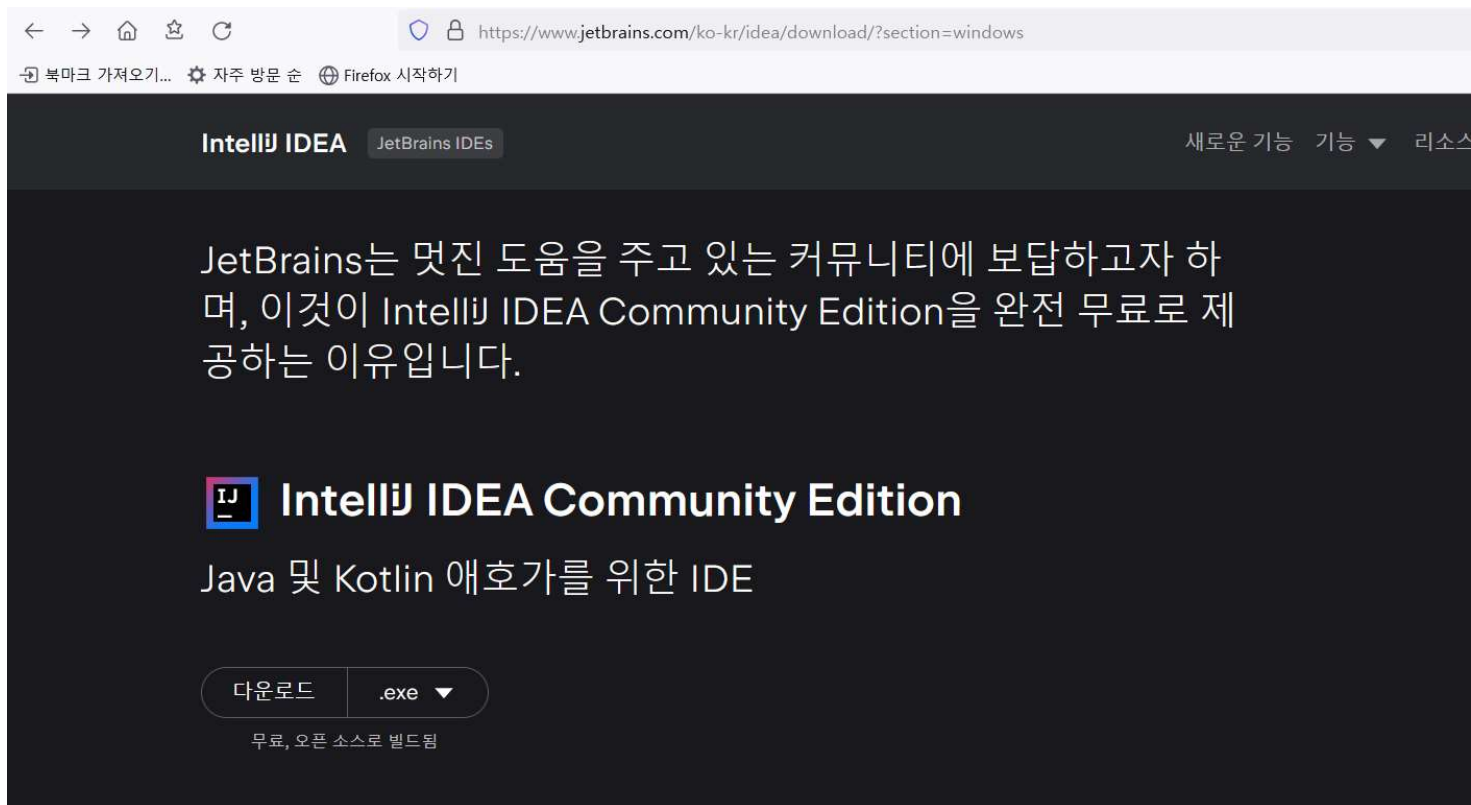
**Java 2.x SDK**

❗ 참고

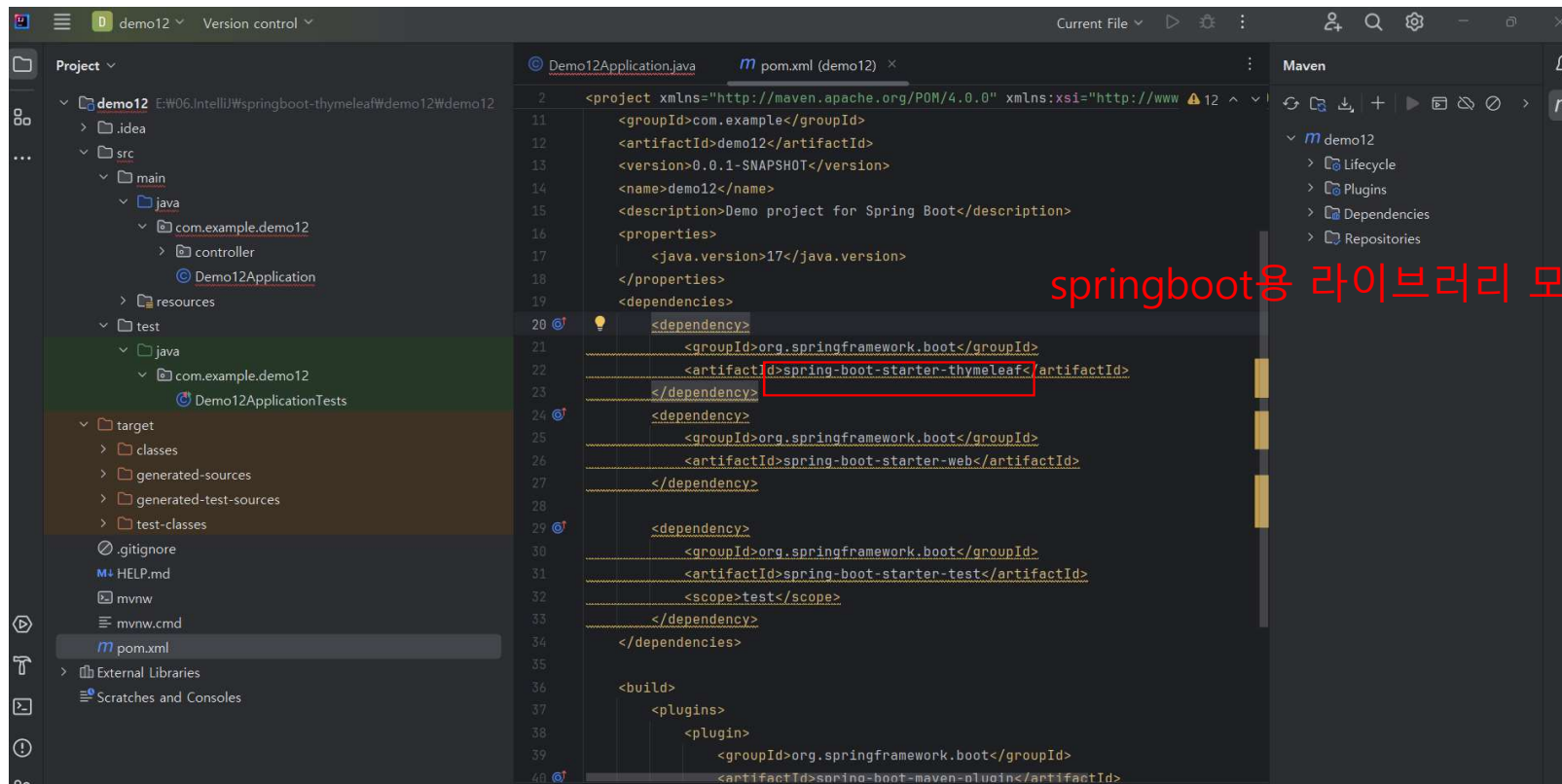
자세한 내용은 예시 확인할 수 GitHub 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을  
배워보세요.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
```

# Intelli J 설치



# pom.xml



# 추가

<https://mvnrepository.com/artifact/log4j/log4j/1.2.17>

**Apache Log4j » 1.2.17**  
Legacy version of Log4j logging framework. Log4j 1 has reached its end of life and is no longer officially supported. It is recommended to migrate to Log4j 2.

License	Apache 2.0
Categories	Logging Frameworks
Tags	logging   log4j   osgi   bundle
Organization	Apache Software Foundation
HomePage	<a href="http://logging.apache.org/log4j/1.2/">http://logging.apache.org/log4j/1.2/</a>
Date	May 26, 2012
Files	<a href="#">pom (21 KB)</a>   <a href="#">bundle (478 KB)</a>   <a href="#">View All</a>
Repositories	<a href="#">Central</a>   <a href="#">Apache Releases</a>   <a href="#">Mulesoft</a>   <a href="#">Apache Staging</a>   <a href="#">BeDataDriven</a>   <a href="#">Redhat GA</a>   <a href="#">Apache Public</a>   <a href="#">Sonatype</a>   <a href="#">Orekit</a>   <a href="#">Hortonworks</a>
Ranking	#18 in MvnRepository (See Top Artifacts) #3 in Logging Frameworks
Used By	18,413 artifacts
Vulnerabilities	<b>Direct vulnerabilities:</b> <a href="#">CVE-2022-23307</a> <a href="#">CVE-2022-23305</a> <a href="#">CVE-2022-23302</a> <a href="#">View 2 more ...</a>

**Popular Categories**

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- JVM Languages
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries
- HTTP Clients
- Logging Bridges
- Dependency Injection
- XML Processing
- Web Frameworks
- I/O Utilities
- Defect Detection Metadata

**Builders:** Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/log4j/log4j -->
<dependency>
  <groupId>log4j</groupId>
  <artifactId>log4j</artifactId>
  <version>1.2.17</version>
</dependency>
```

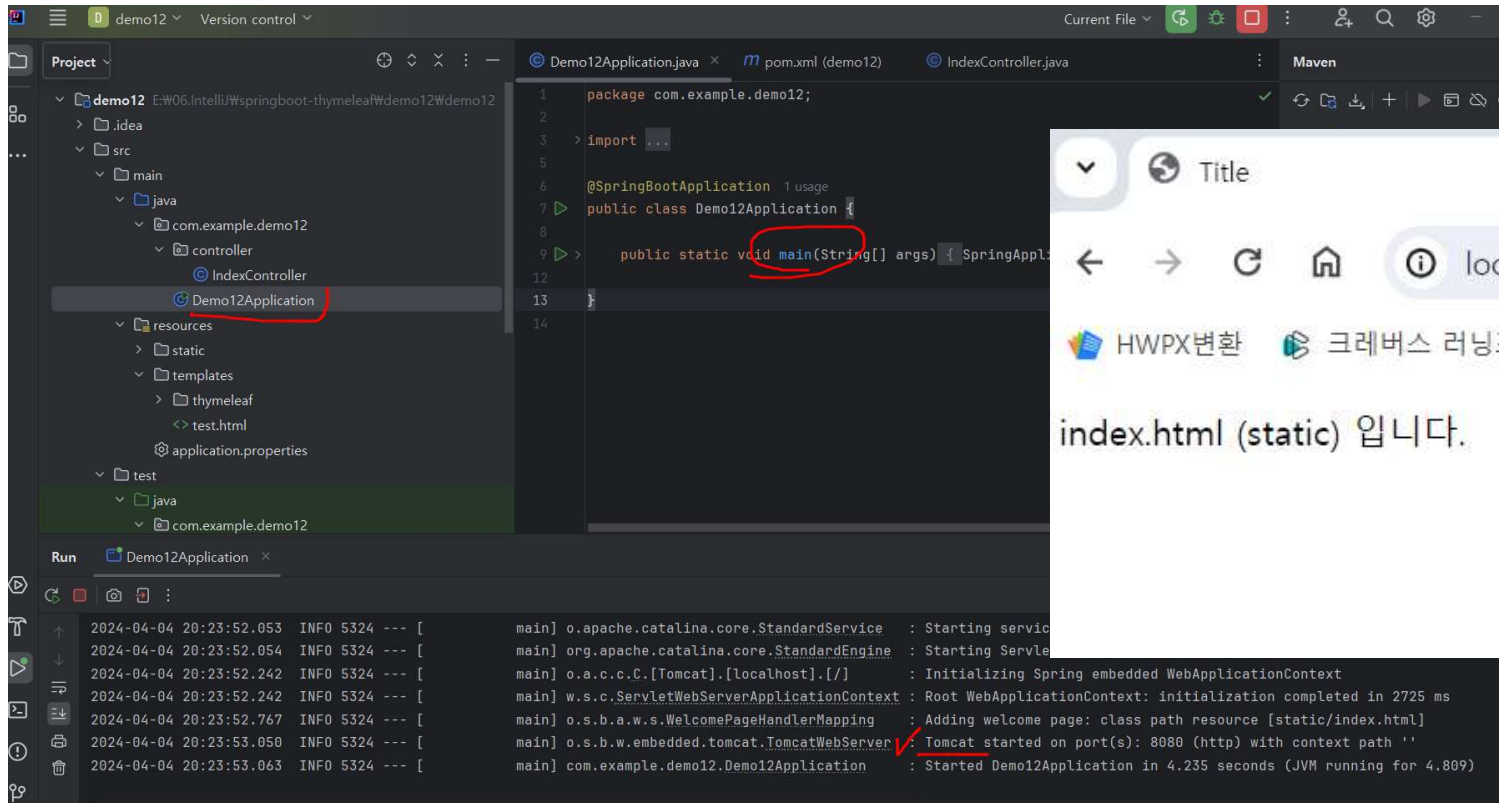
복사해서 추가

- springboot - MVC( data, 화면처리, logic )





# 실행(main있는곳에서 실행)



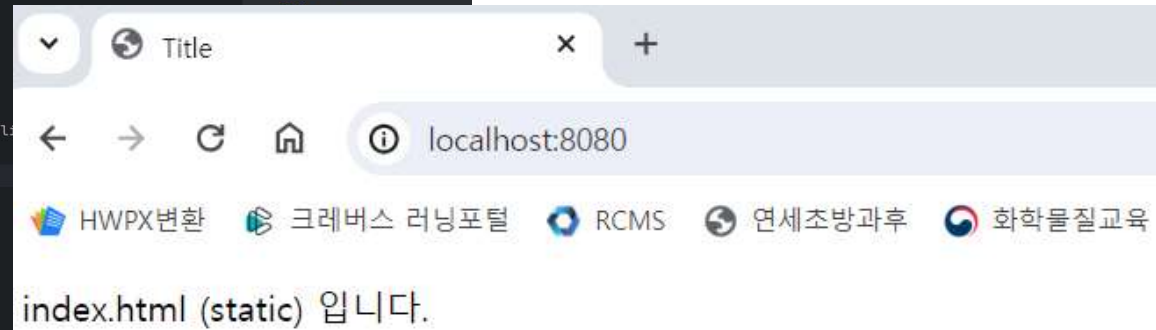
The screenshot shows an IDE with the following components:

- Project View:** The project structure is visible, with the 'Demo12Application' class highlighted in the 'src/main/java/com/example/demo12/controller' package.
- Code Editor:** The 'Demo12Application.java' file is open, showing the following code:

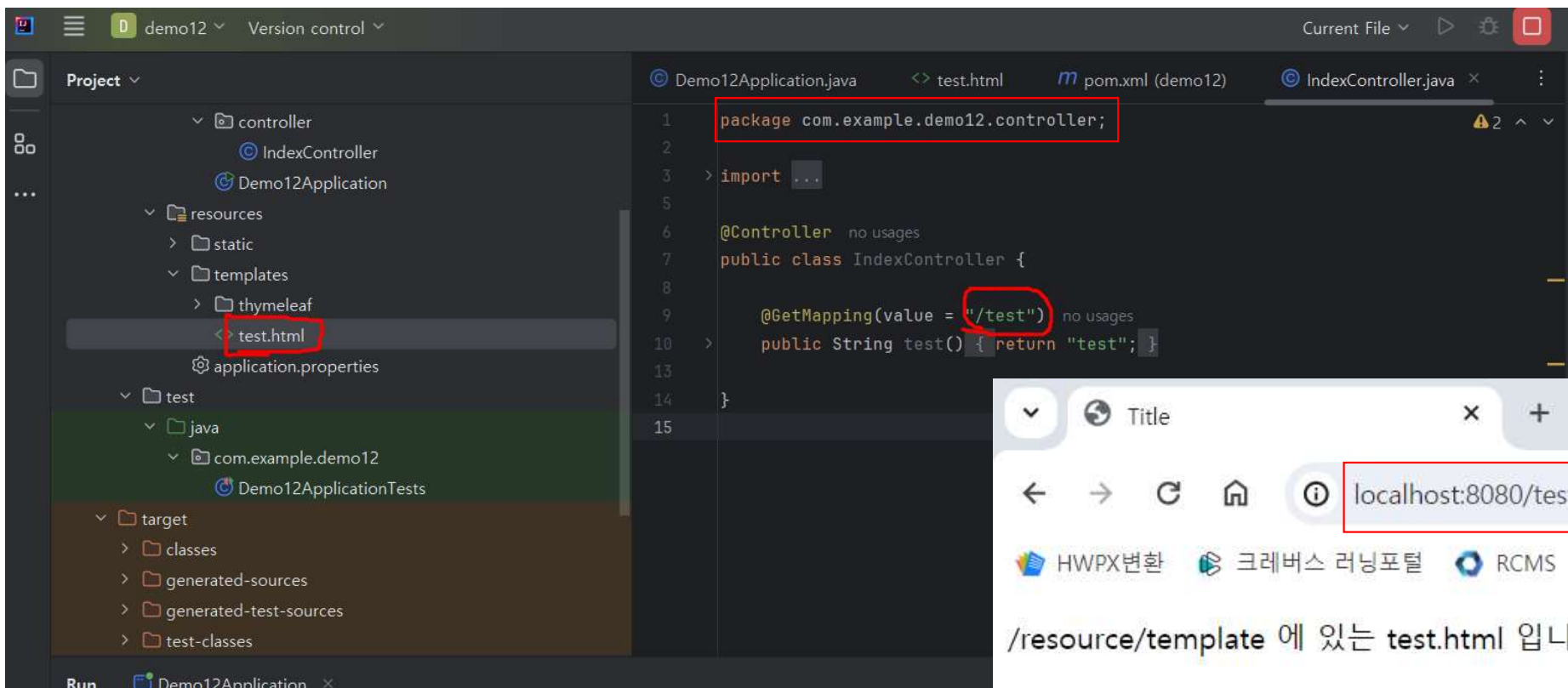
```
1 package com.example.demo12;  
2  
3 import ...  
4  
5 @SpringBootApplication 1 usage  
6 public class Demo12Application {  
7  
8  
9 public static void main(String[] args) { SpringApplication  
10  
11  
12  
13  
14
```

The 'main' method is highlighted with a red circle.
- Run Console:** The output shows the application starting successfully on port 8080. The log includes the following messages:

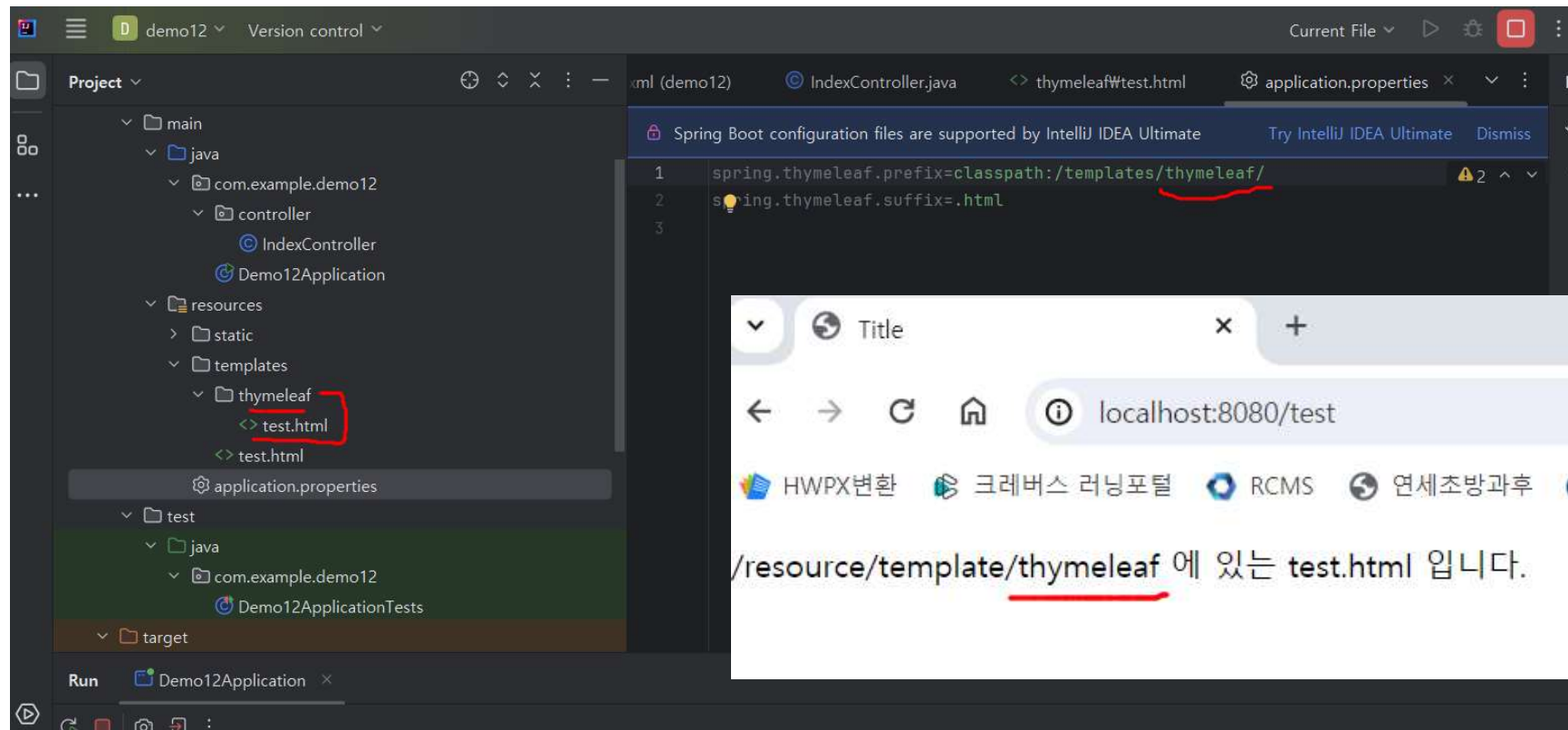
```
2024-04-04 20:23:52.053 INFO 5324 --- [main] o.apache.catalina.core.StandardService : Starting service  
2024-04-04 20:23:52.054 INFO 5324 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet  
2024-04-04 20:23:52.242 INFO 5324 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext  
2024-04-04 20:23:52.242 INFO 5324 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2725 ms  
2024-04-04 20:23:52.767 INFO 5324 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]  
2024-04-04 20:23:53.050 INFO 5324 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''  
2024-04-04 20:23:53.063 INFO 5324 --- [main] com.example.demo12.Demo12Application : Started Demo12Application in 4.235 seconds (JVM running for 4.809)
```



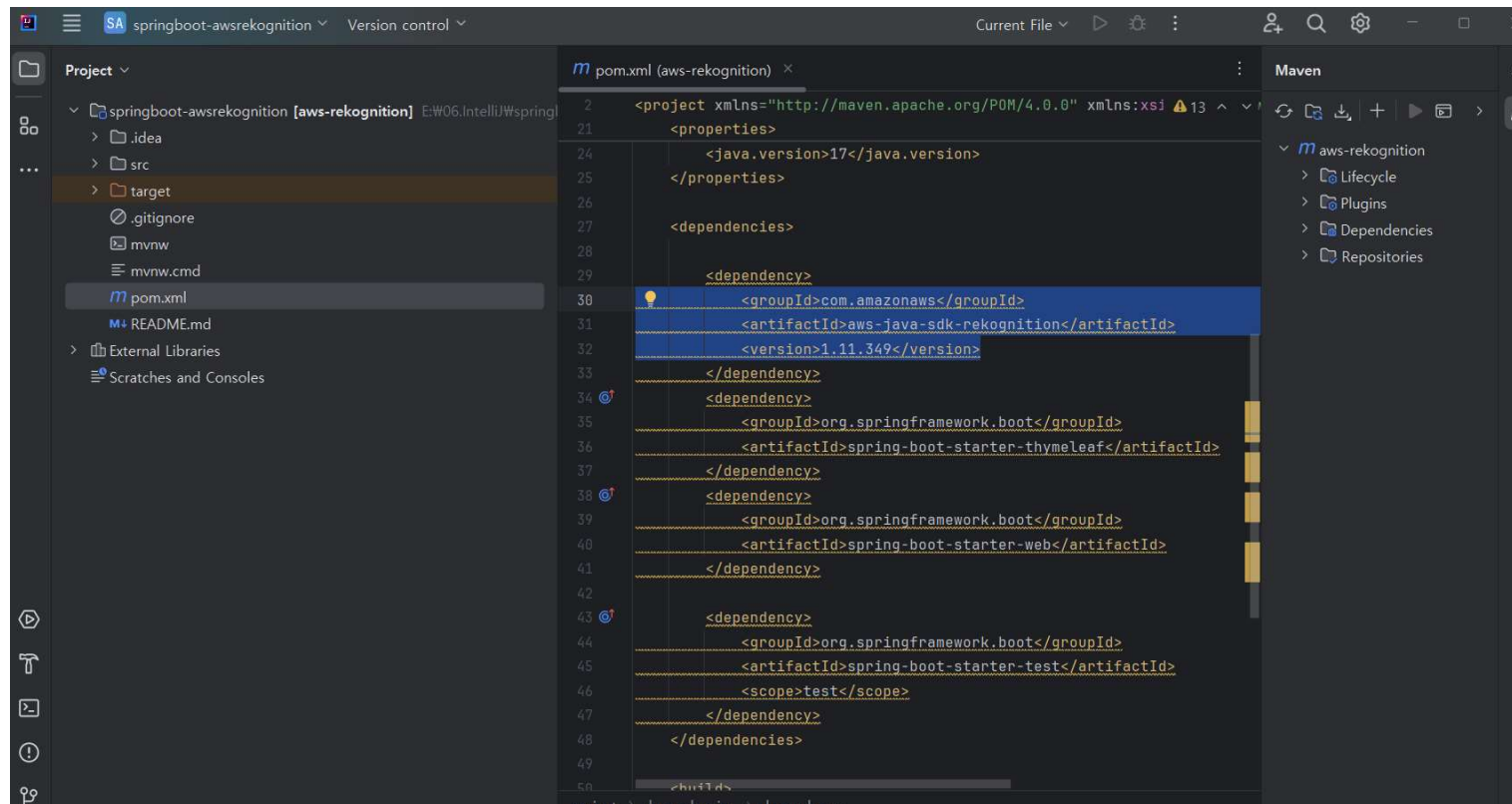
tomcat이 실행되고 나서 localhost:8080 실행



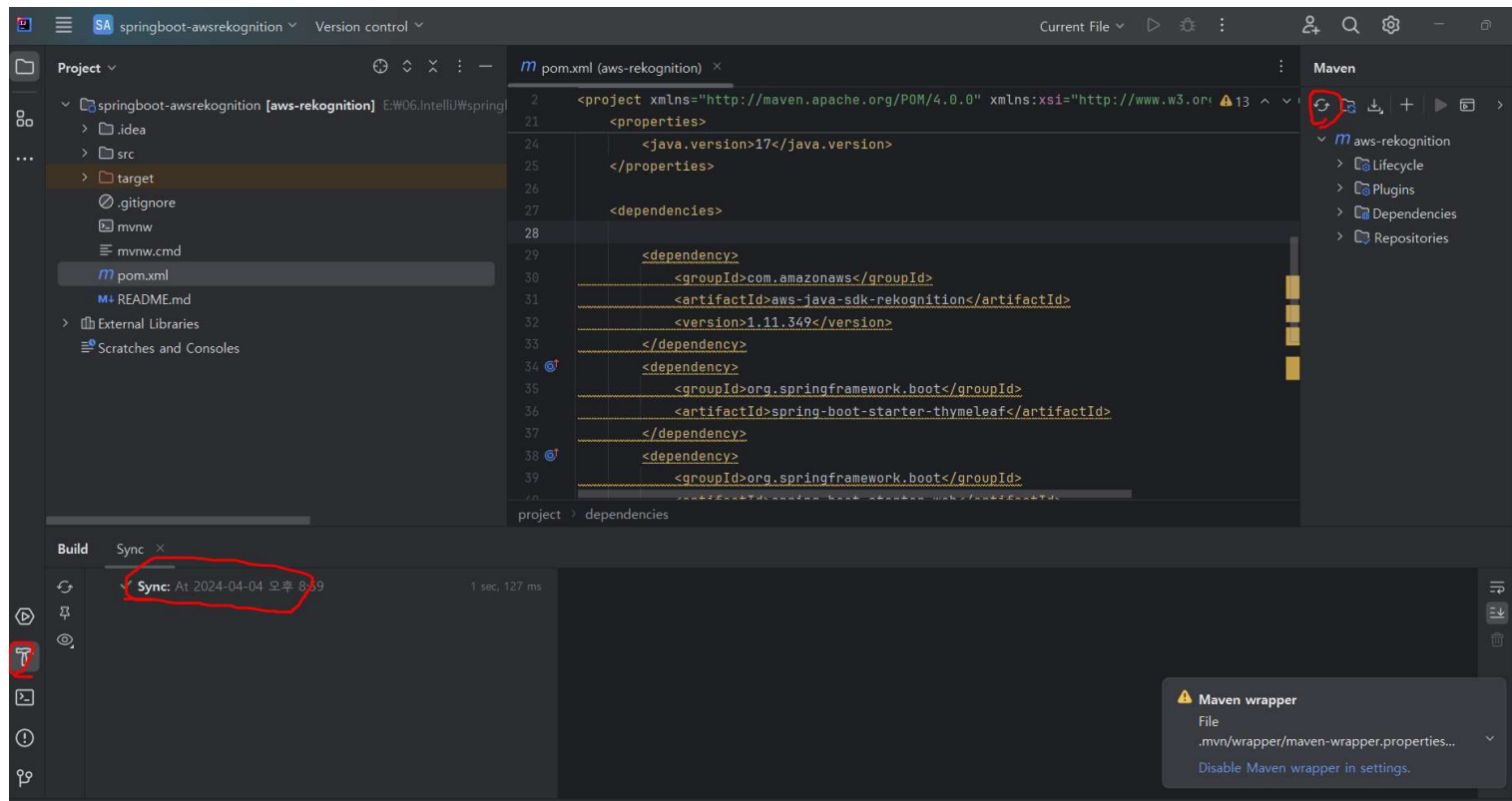
# thymeleaf 폴더안에 test.html 실행



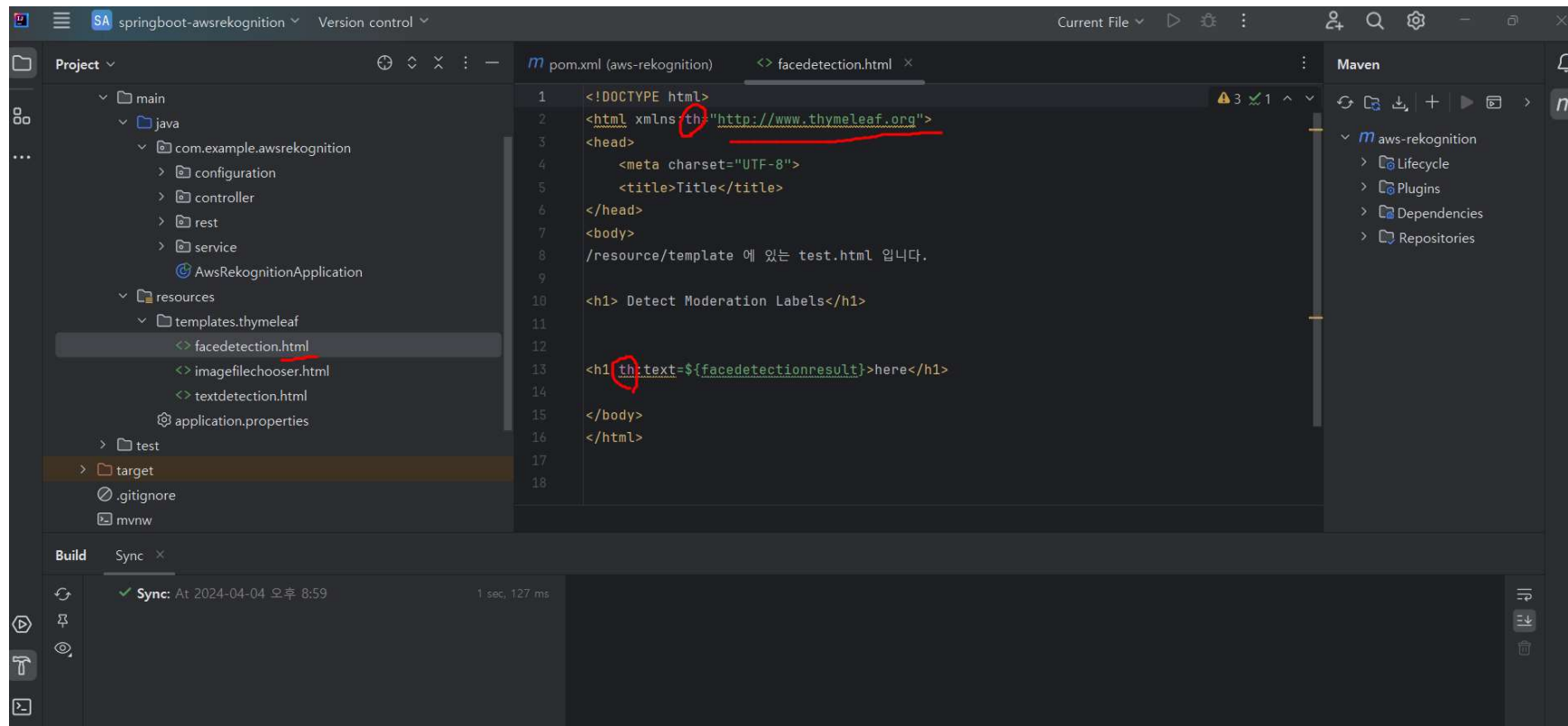
# AWS SDK

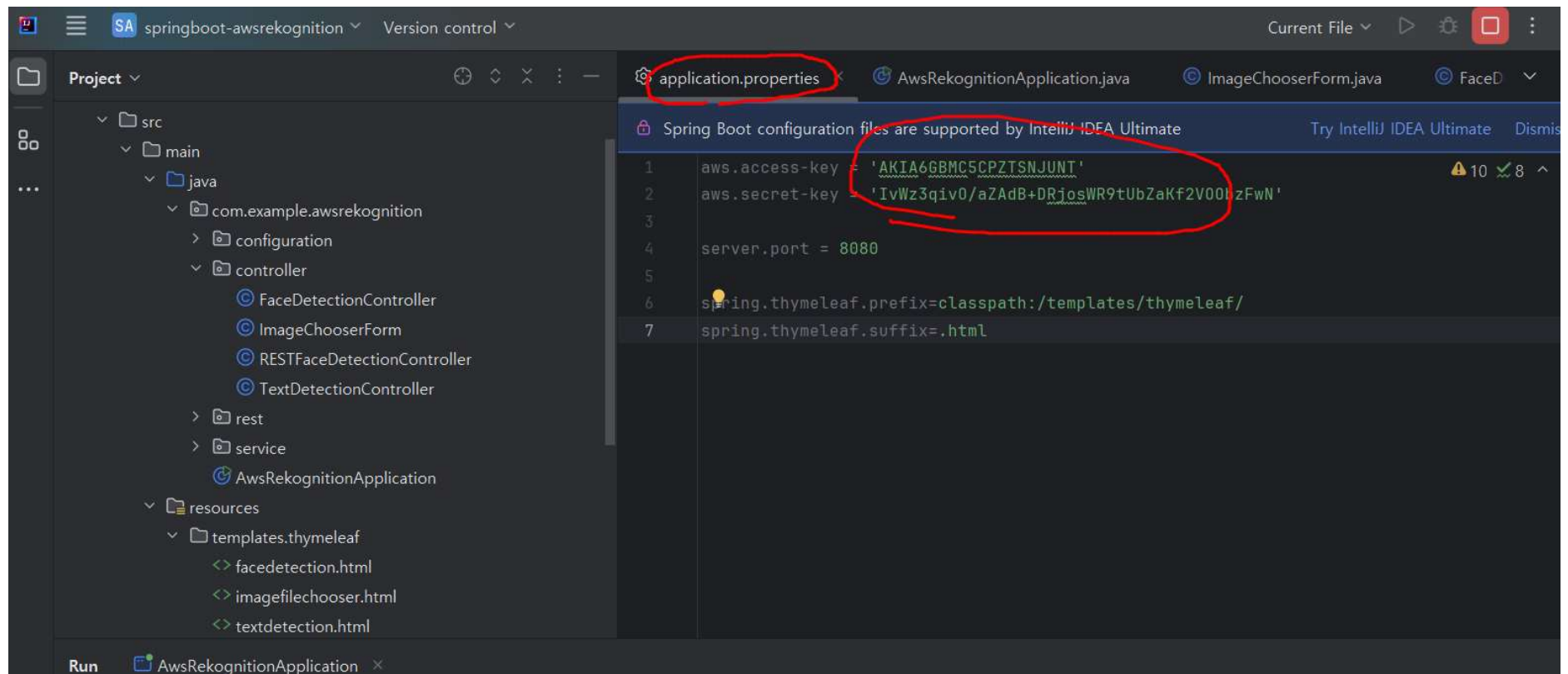


# Sync

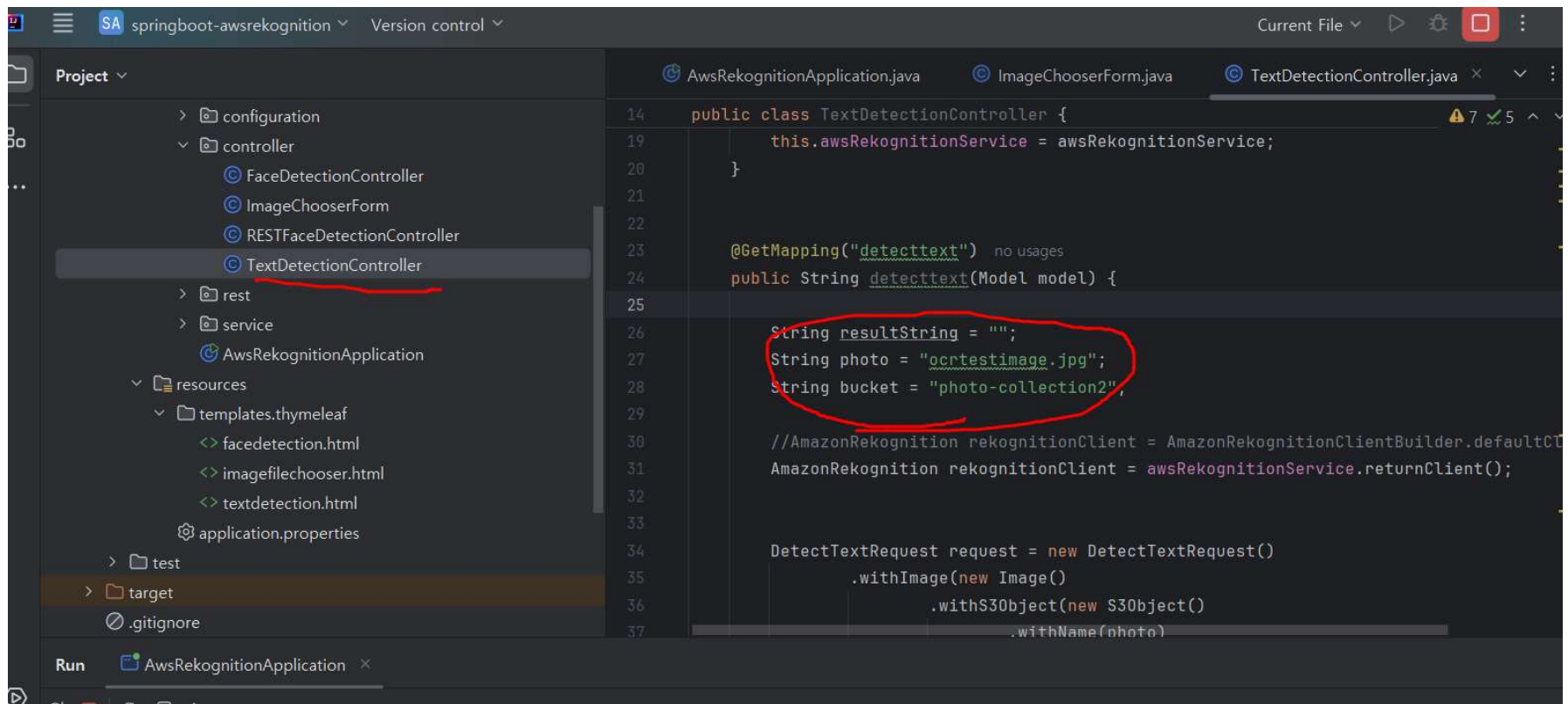


# thymeleaf 문법



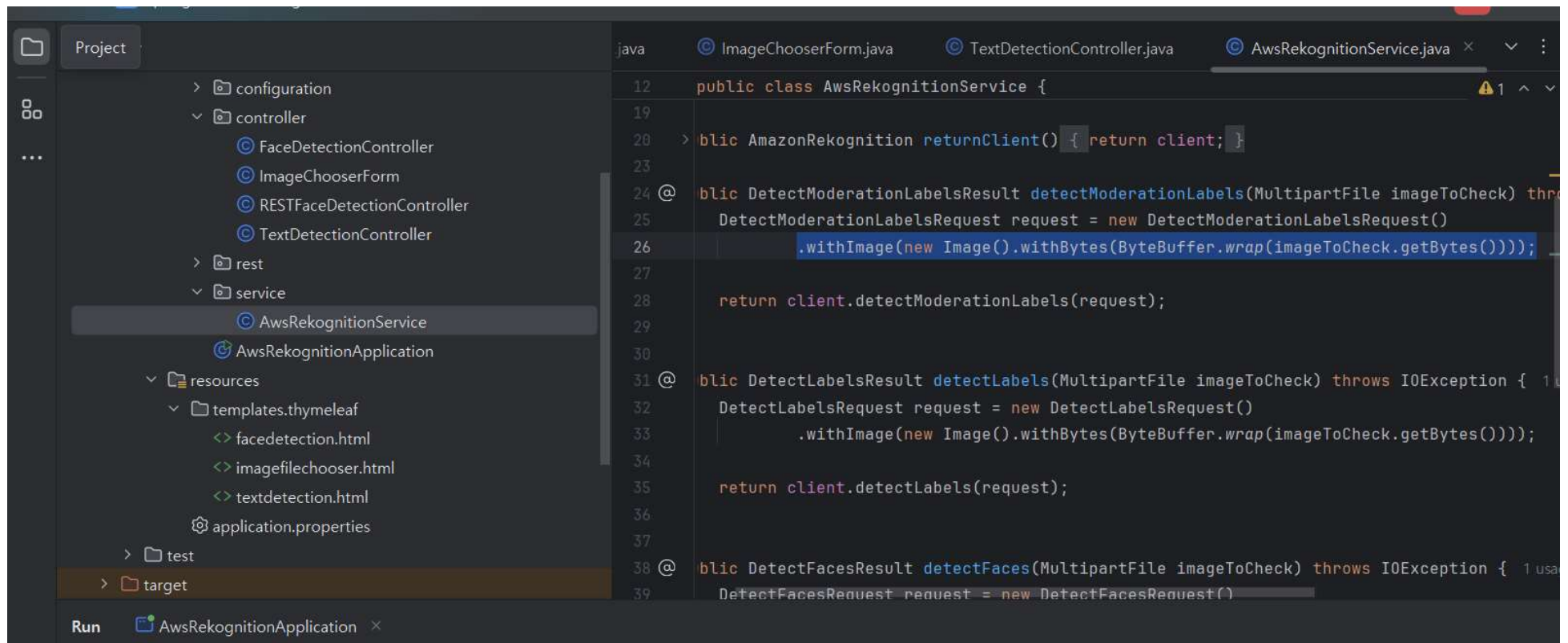




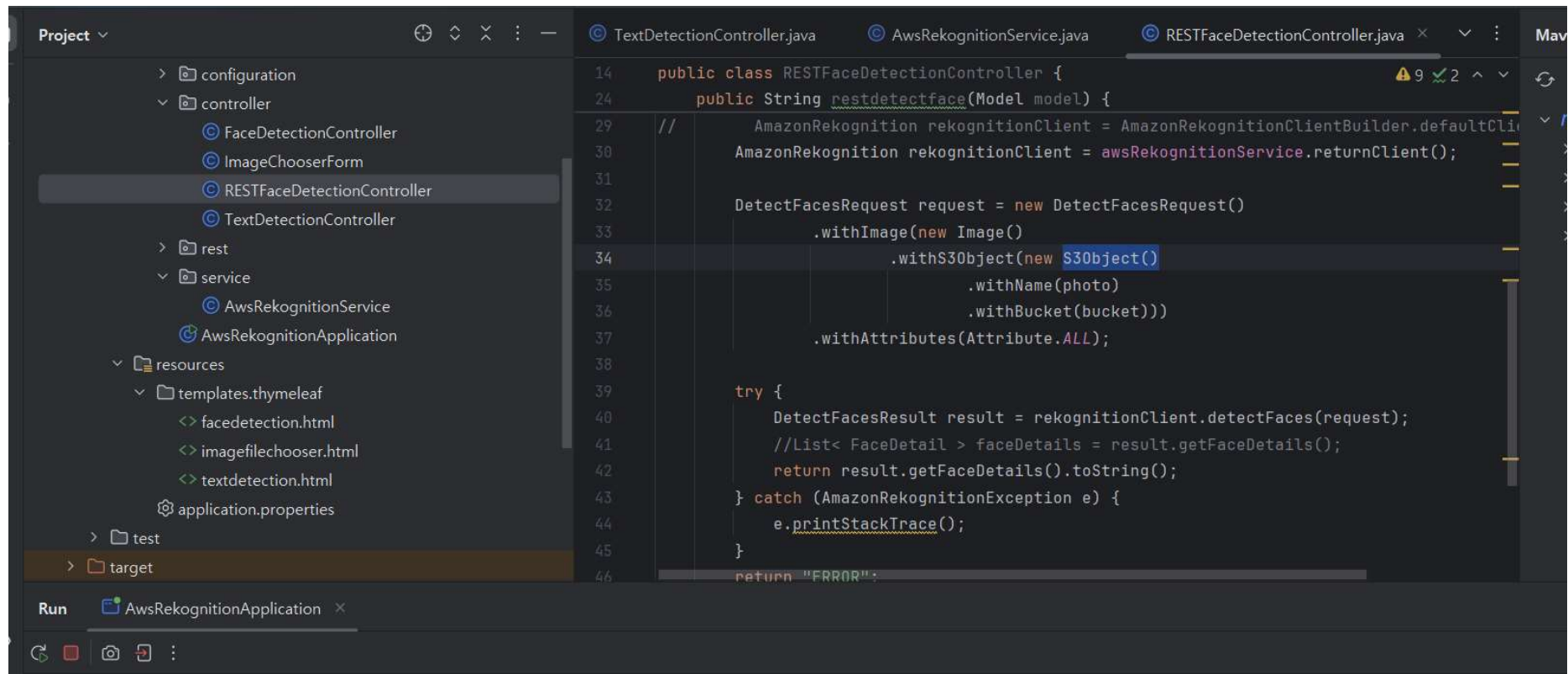




# 바이트어레이로 반환



# S3



# access-key, secret-key

