一、新建 java-agent

1、新建一个 Maven 项目

在 IntelliJ IDEA 创建一个 Maven 项目，用于编写 Java Agent

2、项目结构

```
java-agent/
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── com/
│   │   │   │   ├── example/
│   │   │   │   │   ├── agent/
│   │   │   │   │   │   ├── controller
│   │   │   │   │   │   │   ├──ApiInfoCollector
│   │   │   │   │   │   ├── model
│   │   │   │   │   │   │   ├──ApiInfo
│   │   │   │   │   │   │   ├──ApiLog
│   │   │   │   │   │   ├── transformer
│   │   │   │   │   │   │   ├──ApiInfoCache
│   │   │   │   │   │   │   ├──SpringControllerTransforer
│   │   │   │   │   ├── ApiCollectorAgent
├── pom.xml
```

3、关键文件解释

（1）ApiInfoCollector 文件说明

- **收集 API 信息：**
  - 从 HttpServletRequest 对象中提取以下信息：

        - **URL**：请求的路径（request.getRequestURI()）。
        - **HTTP 方法**：如 GET、POST 等（request.getMethod()）。
        - **方法名称**：调用时传入的 methodName 参数。
        - **请求参数**：从 request.getParameterMap() 中获取请求参数。
    - 将这些信息封装为一个 JSONObject 对象。
- **防止重复收集**：
    - 通过构建一个唯一键（method + ":" + url + ":" + methodName），确保相同的 API 请求不会被重复收集。
- **写入文件**：
    - 将收集到的 API 信息保存到一个 JSON 文件中。
    - 如果文件已存在，会将新的 API 信息追加到文件中。

（2）ApiInfo 文件说明

- **存储 API 信息**：
    - 该类用于表示一个 API 的元数据，包括：
        - **HTTP 方法**（method）：如 GET、POST 等。
        - **路径**（path）：API 的 URL 路径。
        - **类名**（className）：处理该 API 的类名。
        - **方法名**（methodName）：处理该 API 的方法名。
        - **参数类型**（parameterTypes）：API 方法的参数类型列表。

（3）ApiLog 文件说明

- **存储 API 日志信息**：
    - 该类用于表示一个 API 调用的日志记录，包括：
        - **时间戳**（timestamp）：记录 API 调用的时间。
        - **HTTP 方法**（method）：如 GET、POST 等。
        - **请求参数**（parameters）：API 调用的请求参数，以键值对形式存储。

（4）ApiInfoCache

- **缓存 API 信息**：

- 使用一个静态的 ConcurrentHashMap 来存储 ApiInfo 对象，键为 String 类型，值为 ApiInfo 类型。
- 提供方法将 ApiInfo 对象添加到缓存中，并支持获取所有缓存的 API 信息。

（5）SpringControllerTransforer

- **动态修改类：**
  - 使用 javassist 库在运行时修改 Spring 控制器类的方法。
  - 在每个公共方法的前面插入代码，用于记录 API 调用的日志信息。
- **记录 API 日志：**
  - 记录以下信息：
    - **Web API 路径：**从控制器类和方法注解中提取的路径。
    - **HTTP 方法：**从方法注解中提取的 HTTP 方法（如 GET、POST 等）。
    - **请求参数：**记录方法的参数信息。
    - **返回值：**记录方法的返回值信息。
    - **控制器名称：**记录处理请求的控制器类名。
- **保存日志：**
  - 将日志信息保存到内存（ConcurrentHashMap）中，并定期写入 JSON 文件。

（6）ApiCollectorAgent 文件说明

- **Java Agent 入口：**
  - 实现 premain 方法，这是 Java Agent 的标准入口方法。
  - 在 JVM 启动时，该方法会被自动调用。
- **注册类文件转换器：**
  - 通过 Instrumentation 接口的 addTransformer 方法，注册 SpringControllerTransformer 类文件转换器。
  - 注册后，SpringControllerTransformer 会在类加载时动态修改目标类的字节码。

4、打包 java-agent，生成 JAR 包

在 target/ 目录下生成 java-agent-1.0-SNAPSHOT.jar

二、动态加载工具 DynamicAttach

1、创建项目

创建一个独立的 Maven 项目 dynamic-attach，用于动态加载 Java Agent

2、项目结构

dynamic-attacht-project/

```
├── src/
|   ├── main/
|   |   ├── java/
|   |   |   ├── com/
|   |   |   |   ├── example/
|   |   |   |   |   ├── DynamicAttach.java/
|   |   ├── resources/
|   |   |   ├── META-INF/
|   |   |   |   ├── MANIFEST.MF
├── pom.xml
```

3、关键文件解释

（1）MANIFEST.MF 指定 Main-Class: com.example.DynamicAttach

4、打包 dynamic-attacht

生成 dynamic-attach-1.0-SNAPSHOT.jar

三、启动时获取 Api

（1）输入 java -javaagent 启动命令并加载

java -javaagent:D:/Java_workspcae/mianshi/java-agent/target/java-agent-1.0-SNAPSHOT.jar -jar D:/Java_workspcae/mianshi/javaweb-vuln-master/vuln-springboot2/target/vuln-springboot2-3.0.3.jar

```
C:\Users\19800>java -javaagent:D:/Java_workspcae/mianshi/springboot-agent/api-agent/target/api-agent-1.0-SNAPSHOT.jar -j
ar D:/Java_workspcae/mianshi/springboot-agent/javaweb-vuln-master/vuln-springboot2/target/vuln-springboot2-3.0.3.jar
```

（2）配置成功，运行命令后输出

Agent loaded at startup!

API info saved to api_info_startup.json

（3）查看 api_info_ startup.json

四、启动后获取 Api

（1）获取目标 JVM 进程 PID

使用 jps 命令查看目标 JVM 进程的 PID，例如 12345 project-a.jar

（2）输入 java -jar 动态加载命令

java -jar D:/Java_workspcae/mianshi/dynamic-attach/target/dynamic-attach-1.0-SNAPSHOT.jar 12345 java -javaagent:D:/Java_workspcae/mianshi/springboot-agent/api-agent/target/api-agent-1.0-SNAPSHOT.jar

```
C:\Users\19800>java -jar D:/Java_workspcae/mianshi/dynamic-attach/target/dynamic-attach-1.0-SNAPSHOT.jar 22052 java -jav
aagent:D:/Java_workspcae/mianshi/springboot-agent/api-agent/target/api-agent-1.0-SNAPSHOT.jar
```

（3）配置成功，运行命令后输出

Agent loaded successfully!

API info saved to api_info_runtime.json

（4）查看 api_info_runtime.json

五、示例演示

下面以静态加载为示例进行演示

（1）cmd 命令行运行：

java -javaagent:D:/Java_workspcae/mianshi/springboot-agent/api-agent/target/api-agent-1.0-SNAPSHOT.jar -jar D:/Java_workspcae/mianshi/springboot-agent/javaweb-vuln-master/vuln-springboot2/target/vuln-springboot2-3.0.3.jar

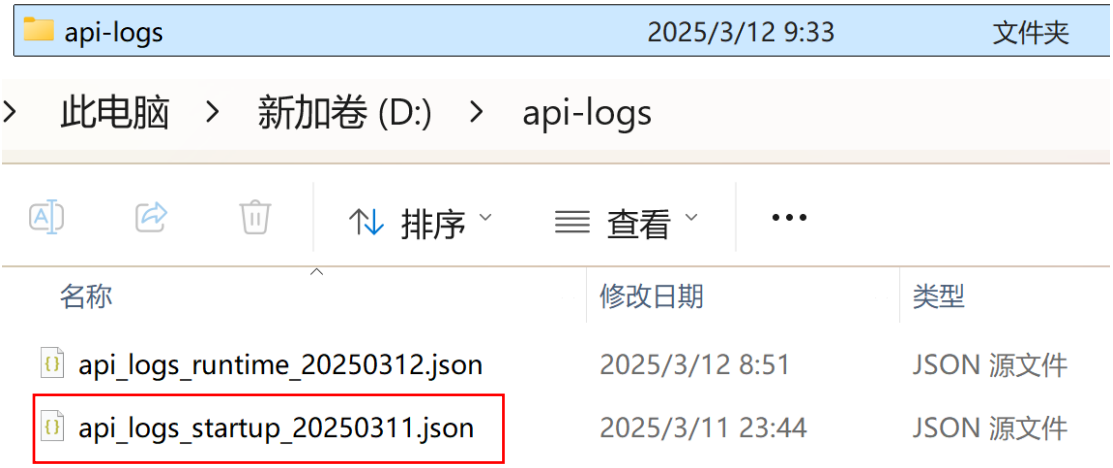
（2）运行成功示例：

```
C:\Users\19800>java -javaagent:D:/Java_workspcae/mianshi/springboot-agent/api-agent/target/api-agent-1.0-SNAPSHOT.jar -j
ar D:/Java_workspcae/mianshi/springboot-agent/javaweb-vuln-master/vuln-springboot2/target/vuln-springboot2-3.0.3.jar
API collector agent started!
 _   _ _  _ _    _ _      ___ ____ ____ ___
|  | | |  | |    |\ |       |  |___ [__   |
 \/  |_| |__| |___ | \|       |  |___ ___]  |   [JavaWeb-Vuln Initializing...]

2025-03-12 08:51:55.975  INFO 21184 --- [           main] o.j.v.c.SpringBoot2Application           : Starting SpringBoot
2Application using Java 11 on LX with PID 21184 (D:\Java_workspcae\mianshi\springboot-agent\javaweb-vuln-master\vuln-spr
ingboot2\target\vuln-springboot2-3.0.3.jar started by 19800 in C:\Users\19800)
2025-03-12 08:51:55.977  INFO 21184 --- [           main] o.j.v.c.SpringBoot2Application           : No active profile s
et, falling back to 1 default profile: "default"
2025-03-12 08:51:57.112  INFO 21184 --- [           main] o.s.b.w.e.t.TomcatWebServer              : Tomcat initialized
with port(s): 8002 (http)
08:51:57,142 [main] INFO  org.apache.coyote.http11.Http11NioProtocol.log(173) - Initializing ProtocolHandler ["http-nio-
8002"]
08:51:57,142 [main] INFO  org.apache.catalina.core.StandardService.log(173) - Starting service [Tomcat]
08:51:57,142 [main] INFO  org.apache.catalina.core.StandardEngine.log(173) - Starting Servlet engine: [Apache Tomcat/9.0
.69]
08:51:57,216 [main] INFO  org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/].log(173) - Initializing Spring
 embedded WebApplicationContext
2025-03-12 08:51:57.216  INFO 21184 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplication
Context: initialization completed in 1206 ms
```

（3）查看 json

json 文件目录:



json 文件内容：