

java-agent学习

java-agent介绍

java-agent是一种能够在不影响正常编译的情况下，修改字节码。java作为一种强类型的语言，不通过编译就不能够进行jar包的生成。而有了java-agent技术，就可以在字节码这个层面对类和方法进行修改。同时，也可以把javaagent理解成一种代码注入的方式。但是这种注入比起spring的aop更加的优美，也就是通过代理实现。

java-agent主要作用

- 1.可以在加载java文件之前做拦截把字节码做修改 (premain方法) 也就是静态的使用方法
- 2.可以在运行java文件中做拦截把字节码做修改 (agentmain方法) 也就是动态的使用方法

java-agent使用

premain

我们可以创建一个springboot项目叫agent-demo-web，实现一个线程每隔500毫秒就调用一次Test了的test方法。

```
package com.example.agentdemoweb;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class AgentDemowebApplication {
    /**
     * 一个线程每隔500毫秒就调用一次Test了的test方法
     * vm 参数 ( -javaagent:E:\java安全\java内存马\agent-example\agent-
demo\target\java-agent.jar) a9
     * @param args
     */
    public static void main(String[] args) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                while (true){
                    new Test().test();
                    try {
                        Thread.sleep(500);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        }).start();
    }
}
```

```
package com.example.agentdemoweb;

public class Test {
    private String hello="hello agent";
    public void test() {
        System.out.println(hello);
    }
}
```

之后在resources目录下放一个Test.class 文件。一定是class文件

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.example.agentdemoweb;

public class Test {
    private String hello = "hello agent";

    public Test() {
    }

    public void test() {
        System.out.println("我被悄悄的插入了一些东西  (◡‿◡) (◡‿◡)");
        System.out.println(this.hello);
    }
}
```

然后需要创建应该agent-demo项目，里面实现AgentDemo方法。

```
package com;

import java.io.IOException;
import java.lang.instrument.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.ProtectionDomain;

public class AgentDemo {
    //java agent 入口
    public static void premain(String agentOps, Instrumentation inst) {
        System.out.println("=====premain方法执行=====");
        simpleDemo(agentOps, inst);
        System.out.println("=====simpleDemo方法执行完=====");
    }

    public static void agentmain(String agentOps, Instrumentation inst) {
        System.out.println("=====agentmain方法执行=====");
        simpleDemo(agentOps, inst);
        //transform是会对尚未加载的类进行增加代理层，这里是已经运行中的jvm，所以类以及被加载
        //必须主动调用retransformClasses让jvm再对运行中的类进行加上代理层
    }
}
```

```

        for (Class allLoadedClass : inst.getAllLoadedClasses()) {
            //这里的Test路径，修改成你自己机器agent-demo-web工程的Test类的路径

            if(allLoadedClass.getName().contains("com.example.agentdemoweb.Test")){
                try {
                    inst.retransformClasses(allLoadedClass);
                } catch (UnmodifiableClassException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    public static void simpleDemo(String agentOps, Instrumentation inst) {
        inst.addTransformer(new ClassFileTransformer() {
            @Override
            public byte[] transform(ClassLoader loader, String className,
                Class<?> classBeingRedefined, ProtectionDomain protectionDomain, byte[]
                classfileBuffer) throws IllegalClassFormatException {
                //判断是指定的class
                if ("com/example/agentdemoweb/Test".equals(className)) {
                    try {
                        //获取更改后的类class 字节数组
                        String path="E:\\java安全\\java内存马\\agent-
                        example\\agent-demo-web\\src\\main\\resources\\Test.class";
                        classfileBuffer = Files.readAllBytes(Paths.get(path));
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                return classfileBuffer;
            }
        },true);
    }
}

```

并且添加依赖

```

<dependencies>
    <dependency>
        <groupId>javassist</groupId>
        <artifactId>javassist</artifactId>
        <version>3.12.1.GA</version>
        <type>jar</type>
    </dependency>
    <dependency>
        <groupId>net.bytebuddy</groupId>
        <artifactId>byte-buddy</artifactId>
        <version>1.9.2</version>
    </dependency>

    <dependency>
        <groupId>net.bytebuddy</groupId>
        <artifactId>byte-buddy-agent</artifactId>
        <version>1.9.2</version>
    </dependency>
</dependencies>

```

为了生成jar包并且需要有MANIFEST.MF，就将内容放入pom.xml中

```
<build>
  <finalName>java-agent</finalName>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
      <configuration>
        <archive>
          <manifest>
            <addClasspath>true</addClasspath>
          </manifest>
          <manifestEntries>
            <Premain-Class>
              com.AgentDemo
            </Premain-Class>
            <Agent-Class>
              com.AgentDemo
            </Agent-Class>
            <Can-Redefine-Classes>
              true
            </Can-Redefine-Classes>
            <Can-Retransform-Classes>
              true
            </Can-Retransform-Classes>
          </manifestEntries>
        </archive>
      </configuration>
    </plugin>
  </plugins>
</build>
```

直接在agent-demo项目中执行mvn install 命令就在target目录下生成java-agent.jar
之后配置运行的参数。

VM options: -javaagent:E:\java安全\java内存马\agent-example\agent-demo\target\java-agent.jar + ↵

执行就会发现成功在main方法执行前执行，并且成功修改了Test类

```
=====premain方法执行=====
=====simpleDemo方法执行完=====
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)
hello agent
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)
hello agent
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)
hello agent
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)
hello agent
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)
hello agent
```

agentmain

在创建一个attach-web项目，写入Attach代码

```
package com.example.attach;

import com.sun.tools.attach.*;

import java.io.IOException;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

public class Attach {
    public static void main(String[] args) {
        //查找所有jvm进程，排除attach测试工程
        List<VirtualMachineDescriptor> attach = VirtualMachine.list()
            .stream()
            .filter(jvm -> {
                return !jvm.displayName().contains("Attach");
            }).collect(Collectors.toList());
        for (int i = 0; i < attach.size(); i++) {
            System.out.println "[" + i + "] " +
attach.get(i).displayName() + ":" + attach.get(i).id());
        }
        System.out.println("请输入需要attach的pid编号");
        Scanner scanner = new Scanner(System.in);
        String s = scanner.nextLine();
        VirtualMachineDescriptor virtualMachineDescriptor =
attach.get(Integer.valueOf(s));
        try {
            VirtualMachine virtualMachine =
VirtualMachine.attach(virtualMachineDescriptor.id()); //执行agentmain
            virtualMachine.loadAgent("E:\\java安全\\java内存马\\agent-
example\\agent-demo\\target\\java-agent.jar", "param"); //附加
            virtualMachine.detach(); //附加完成之后分离
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

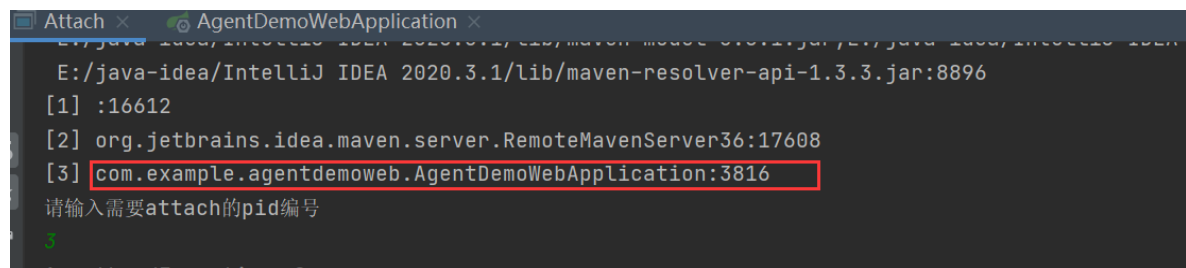
    } catch (AttachNotSupportedException e) {
        System.out.println("AttachNotSupportedException: " +
e.getMessage());
    } catch (IOException e) {
        System.out.println("IOException: " + e.getMessage());
    } catch (AgentLoadException e) {
        System.out.println("AgentLoadException: " + e.getMessage());
    } catch (AgentInitializationException e) {
        System.out.println("AgentInitializationException: " +
e.getMessage());
    }
}
}
}

```

要修改运行中的类，就需要获得运行类jvm的pid，然后loadAgent,也就是我们最开始生成的jar。

然后运行AgentDemoWebApplication的同时取消之前的配置 -javaagent:xxx

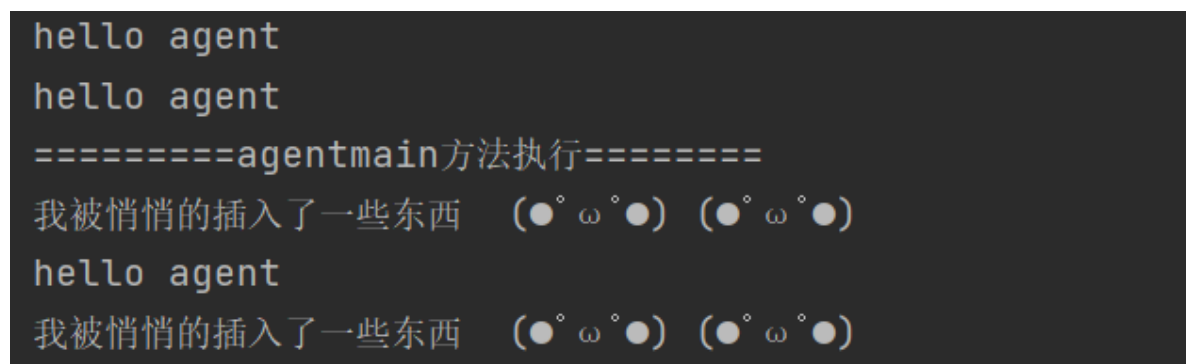
并且运行Attach，输入AgentDemoWebApplication运行的pid就可以成功修改AgentDemoWebApplication运行时的加载的test类。



```

Attach x AgentDemoWebApplication x
E:/java-idea/IntelliJ IDEA 2020.3.1/lib/maven-resolver-api-1.3.3.jar:8896
[1] :16612
[2] org.jetbrains.idea.maven.server.RemoteMavenServer36:17608
[3] com.example.agentdemoweb.AgentDemoWebApplication:3816
请输入需要attach的pid编号
3

```



```

hello agent
hello agent
====agentmain方法执行====
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)
hello agent
我被悄悄的插入了一些东西  (●° ω °●) (●° ω °●)

```