31_阶段测试：验证基于 Spring AOP 机制的日志打印结果是否正确？

1、开篇

上节课首先复习了 Spring AOP 的 Join point、Pointcuts、Advice 的概念，然后介绍了如何在项目中定义日志类 ControllerLogAspect，针对所有 Controller 中的方法进行日志记录。本节课我们会测试日志代码，看看结果如何。本节课看看如何使用 Spring AOP 机制拦截 Controller 打印日志，内容包括：

- 打包服务
- 上传服务
- 启动服务
- 查看日志

2、打包服务

如图 1 所示，打开我们熟悉的酒店管理后台应用服务，在 IntelliJ IDEA 中选择下方的“Teminal”按钮，在显示的命令行中（红色箭头的地方）输入对应的命令，用来对项目进行打包。

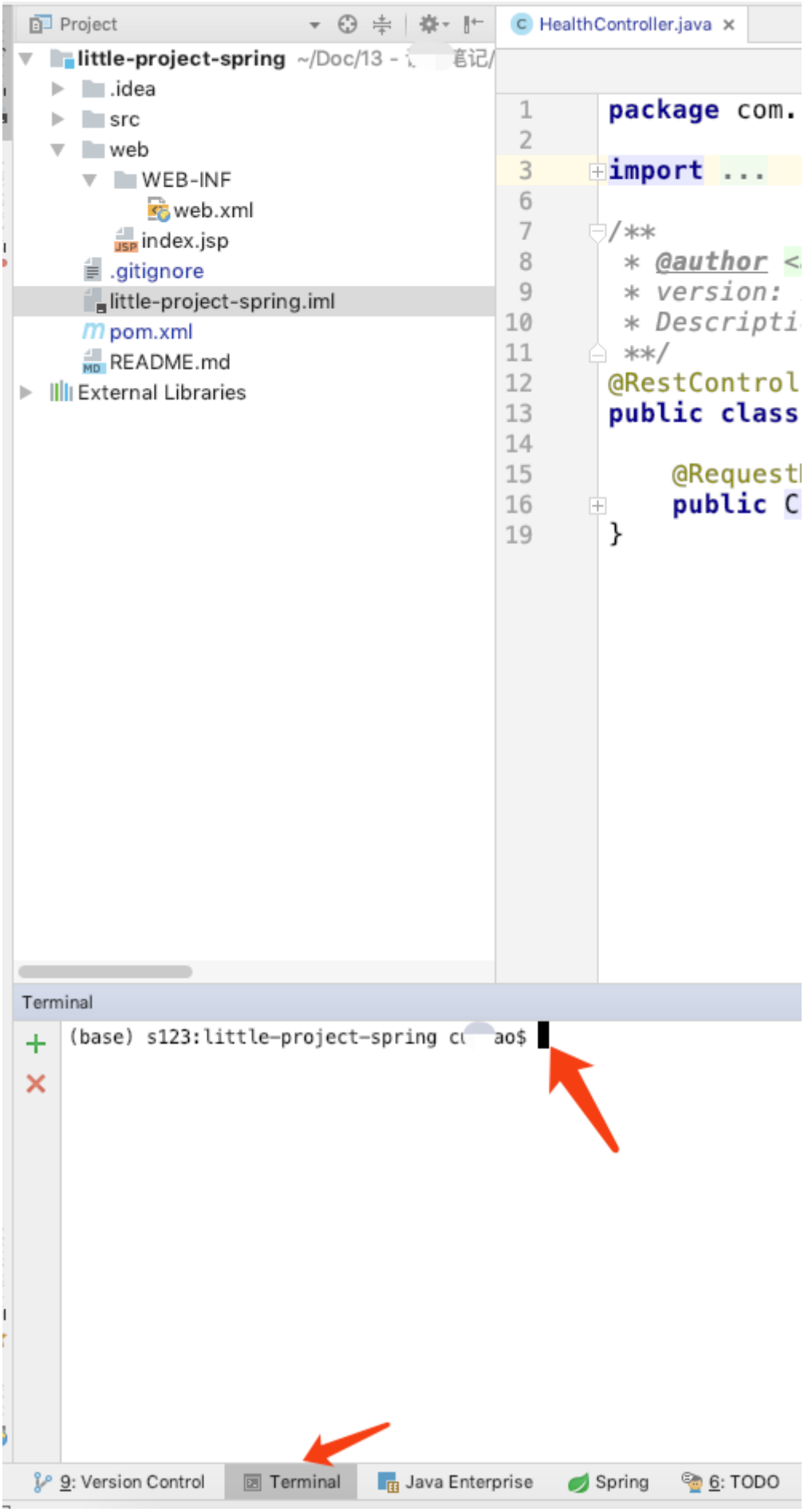
Project
little-project-spring ~/Doc/13 -
.idea
src
web
WEB-INF
web.xml
index.jsp
.gitignore
little-project-spring.iml
pom.xml
README.md
External Libraries
HealthController.java
package com.
import ...
/**
* @author
* version:
* Descripti
**/
@RestControl
public class
@Request
public C
}
Terminal
(base) s123:little-project-spring
ao$
9: Version Control
Terminal
Java Enterprise
Spring
6: TODO

图 1 打开命令行

如图 2 所示，这里输入“mvn clean package” 的命令，这里表明通过 Maven 进行打包。

图 2 输入打包命令

在输入命令以后回车，如图 3 所示，会在输出信息中看到“BUILD SUCCESS”的字样表示，打包成功。同时在项目文件的 target 目录下面会看到一个“little-project-spring.war”的文件，这就是我们将要发布的 war 包。后面的操作会用到它。

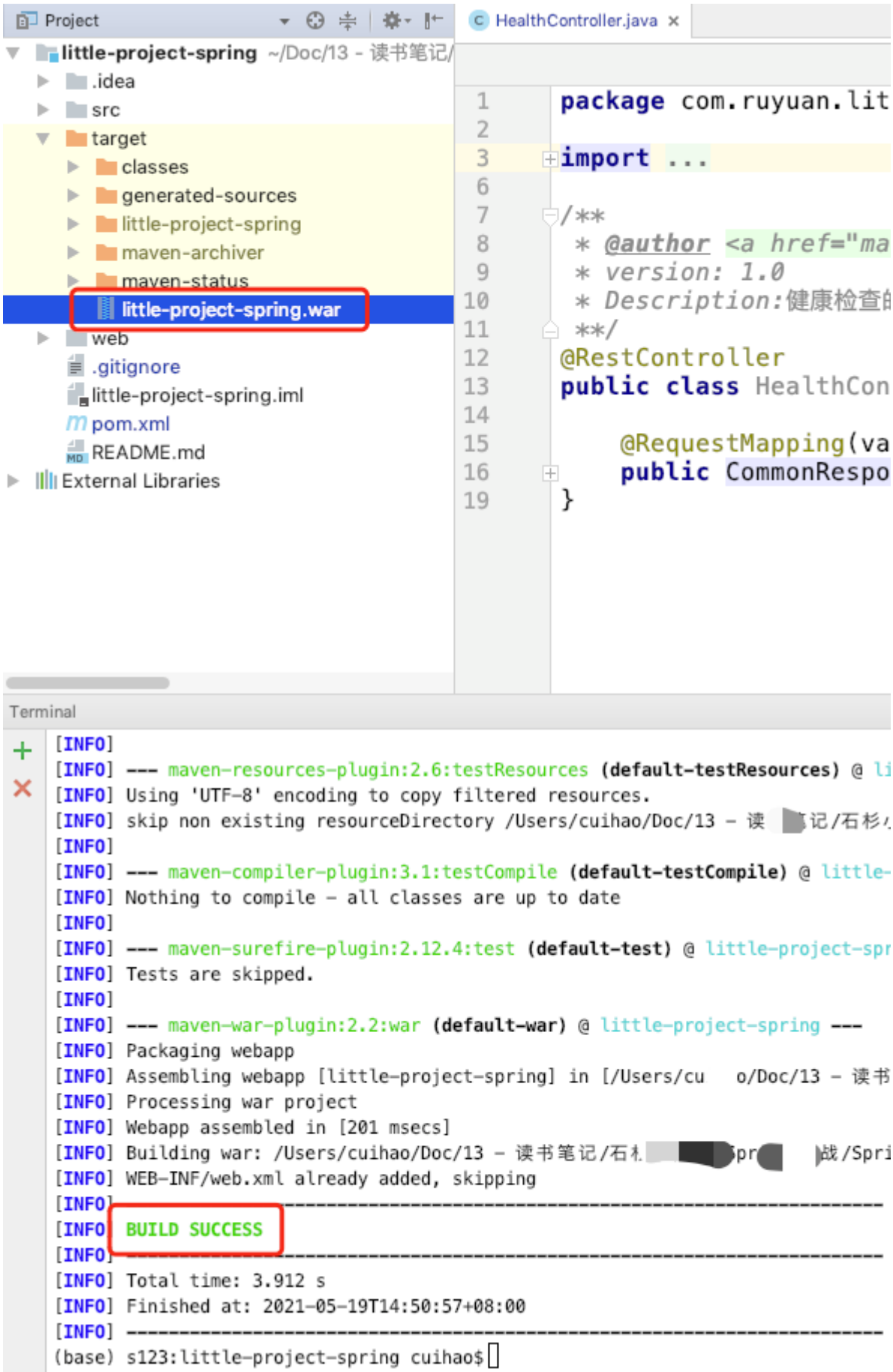
Project
little-project-spring ~/Doc/13 - 读书笔记/
.idea
src
target
classes
generated-sources
little-project-spring
maven-archiver
maven-status
little-project-spring.war
web
.gitignore
little-project-spring.iml
pom.xml
README.md
External Libraries
HealthController.java
1 package com.ruyuan.lit
2
3 import ...
6
7 /**
8 * @author <a href="ma
9 * version: 1.0
10 * Description:健康检查
11 **/
12 @RestController
13 public class HealthCon
14
15 @RequestMapping(va
16 public CommonRespo
19 }
Terminal
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ li
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/cuihao/Doc/13 - 读
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ little-
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ little-project-spr
[INFO] Tests are skipped.
[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ little-project-spring ---
[INFO] Packaging webapp
[INFO] Assembling webapp [little-project-spring] in [/Users/cu o/Doc/13 - 读书
[INFO] Processing war project
[INFO] Webapp assembled in [201 msecs]
[INFO] Building war: /Users/cuihao/Doc/13 - 读书笔记/石
[INFO] WEB-INF/web.xml already added, skipping
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 3.912 s
[INFO] Finished at: 2021-05-19T14:50:57+08:00
[INFO] --
(base) s123:little-project-spring cuihao$

图 3 打包成功

3、上传服务

在上传服务（jar 包）之前需要保证，在对应的 ECS 中建立服务运行的目录如下：

/home/admin/little-project-spring/

这个目录已经由儒猿团队在 ECS 的镜像中创建好了，大家不用手动创建，我们后面的发布就基于这个目录。其中“/home/admin/little-project-spring”是用来存放部署脚本的，注意我们这里使用的 deploy.sh 脚本已经由儒猿团队上传了。

“/home/admin/little-project-spring/”目录是存放服务的 war 包的，这个包是需要我们自己上传的。

在 ECS 上建立好目录结构以后，再回到本地的项目中，依旧是在 IntelliJ IDEA 的命令行中输入以下命令：

scp target/little-project-spring.war root@47.117.120.102:/home/admin/little-project-spring /

命令的意思是通过 scp 命令将刚才打包的“little-project-spring.war”文件 copy 到对应 ECS 服务器的“/home/admin/little-project- spring/”目录中。这里的“47.117.120.102”是我的测试地址，大家可以更换为自己申请的 ECS 的 IP 地址。

这里需要特别说明一下，由于我在执行命令的根目录在“little-project- spring”项目下面，如果你在其他的地方执行上述两条命令，需要指定好源文件的目录。同时在使用 scp 命令以后会让大家输入服务器的密码，该密码可以从实战云平台上获取。

完成上面两个命令以后，服务就已经部署到 ECS 了。

4、启动服务

完成部署以后，需要到 ECS 服务器上面启动部署的酒店管理后台的服务。还是登录 ECS 服务器，通过以下命令进入到“deploy.sh”文件所在的目录。

cd /home/admin/little-project-spring

deploy.sh 文件是儒猿团队为大家生成的发布的脚本文件，通过 Linux shell 脚本完成发布的参数配置和启动命令。包括启动应用等待的时间、应用端口号、健康检查的 URL 以及 jar 包的目录和日志信息。有兴趣的同学可以打开看看，这里就不做展开的介绍了。

保证当前目录下面存在“deploy.sh”文件，使用如下命令启动服务。

```
sh deploy.sh restart
```

命令使用了“restart”作用与“start”是一致的，用“restart”的目的以免在重复发布过程中，学员忘记是否启动过服务。因此使用“restart”，这样即便是已经启动过服务，也会重新加载服务。

运行命令在看到图 4 所示的“success”字样的时候，就说明服务启动成功了。

```
[root@iZuf64qotsm6gzy9jnkw8cZ little-project-spring]# sh deploy.sh restart
no java process
starting java process
Using CATALINA_BASE:   /home/admin/little-project-spring/apache-tomcat-9.0.45
Using CATALINA_HOME:   /home/admin/little-project-spring/apache-tomcat-9.0.45
Using CATALINA_TMPDIR: /home/admin/little-project-spring/apache-tomcat-9.0.45/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/admin/little-project-spring/apache-tomcat-9.0.45/bin/bootstrap.jar:/home/admin/little
-project-spring/apache-tomcat-9.0.45/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
started java process
checking http://127.0.0.1:8090
Wait app to pass health check: 1...
code is 200
check http://127.0.0.1:8090 success
```

图 4 启动服务成功

如图 5 所示，同样可以通过浏览器输入 IP 地址加上端口 8090 来查看健康检查服务是否工作。http://47.117.115.80:8090，其中把 IP 地址换成你自己的。

```
{"code":200,"message":"成功","data":null}
```

图 5 访问健康服务成功

5、查看日志

依旧在登陆的 ECS 服务器中输入“cd /home/admin/little-project-spring/logs”，这个是我们在 Logback 中配置的日志文件的地址。然后通过命令“vim little-project-spring.log”查看文件的内容。如图 6 所示，因为我们访问 http://47.117.115.80:8090 的时候，实际上是访问的 HealthController，虽然我们没有在 HealthController 写任何的 Logback 代码，但是通过 ControllerLogAspect 自定义日志类就可以截获并记录 Controller 中的日志。

这里记录了请求处理方法：

com.ruyuan.little.project.spring.controller.HealthConroller.health。从方法名字可

以看出包含了 Controller 的 namesapce 以及对应的方法名字：health。后面的请求参数名和请求参数值都是空，不过请求处理时间差是 32。同时返回了响应结果：{“code”:200,”message”:”成功”}。

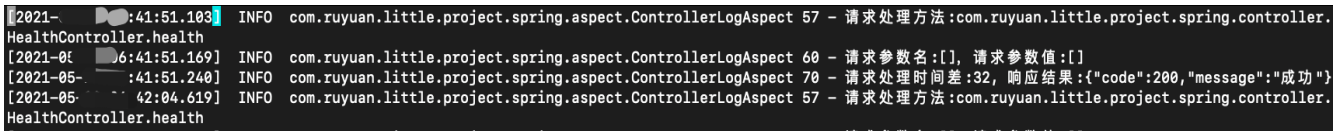

图 6 查看日志内容

6、总结

本节课将上节课写好的代码通过打包、上传、以及启动服务的方式发布到 ECS，同时登陆服务器查看 logs 目录下面的日志文件，可以看到 ControllerLogAspect 中编写的日志代码产生效果。下节课我们会讲到如何解决系统内部异常未处理导致返回错误页面问题。下期见，拜拜。