33_技术挑战：如何解决系统内部异常未处理导致返回错误页面问题？

1、开篇

上节课把日志代码通过打包、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。本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。今天的内容：

- 抛出异常
- 打包服务
- 上传服务
- 启动服务

2、抛出异常

在平时开发过程中，需要处理各种各样的异常情况，捕捉异常场景也是千奇百怪。特别是在 Controller 这层如果出现异常，页面会抛出错误信息。这里我们基于上节课的代码加入一个异常看看执行结果是怎么样的。

如图 1 所示，我们在 HealthController 的 health 方法里面加入一个异常语句：“throw new Exception(“exception test”); ”目的是强行抛出一个异常，看看服务返回什么数据。

图 1 抛出异常

后面的过程需要打包并且发布到 ECS 上面查看，运行结果。

3、打包服务

如图 2 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。

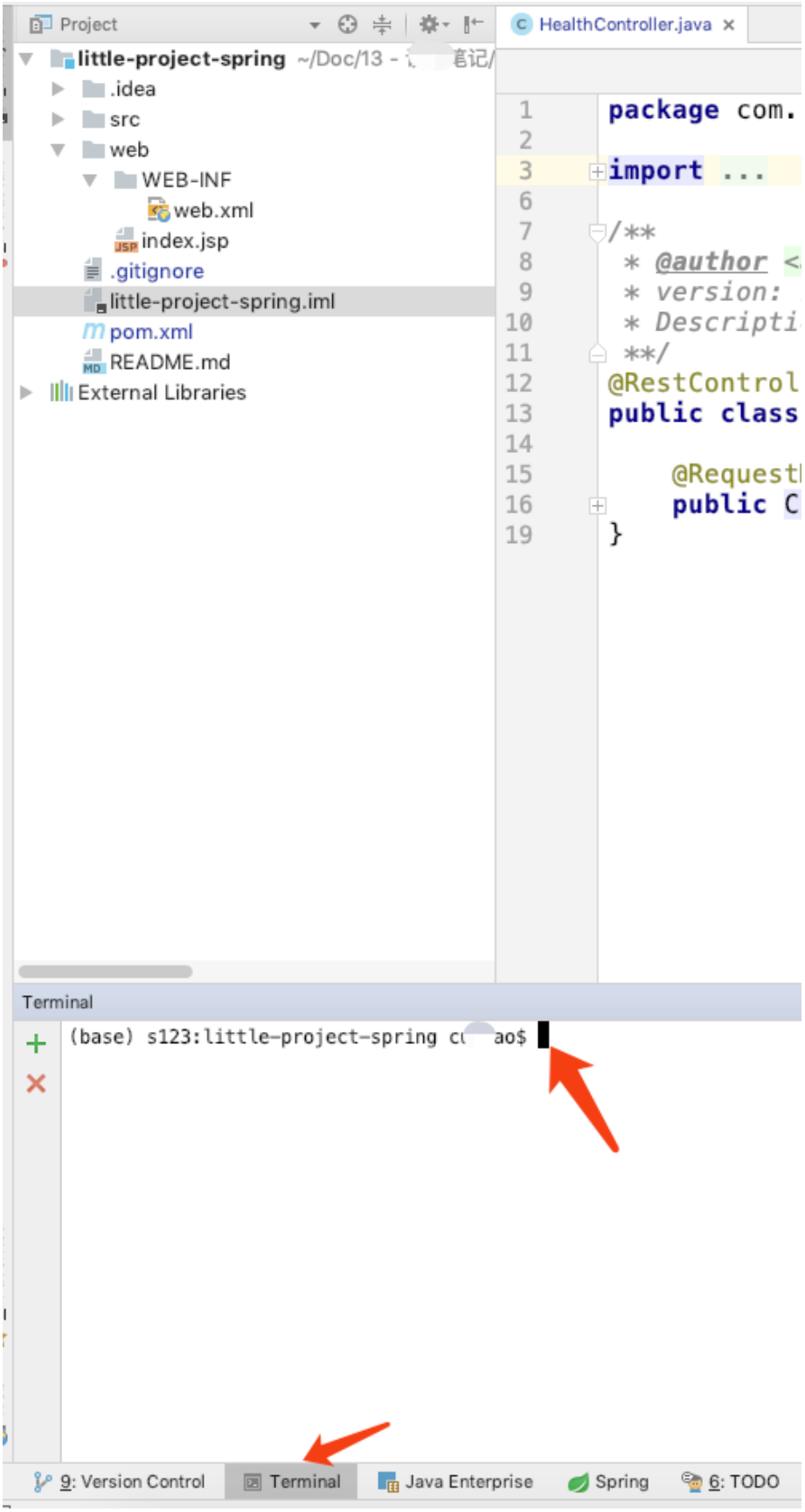
Project
little-project-spring ~/Doc/13 -
.idea
src
web
WEB-INF
web.xml
index.jsp
.gitignore
little-project-spring.iml
pom.xml
README.md
External Libraries
HealthController.java
package com.
import ...
/**
* @author
* version:
* Descripti
**/
@RestControl
public class
@Request
public C
}
Terminal
(base)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

图 2 打开命令行

如图 3 所示，这里输入“mvn clean package” 的命令，这里表明通过 Maven 进行打包。

图 3 输入打包命令

在输入命令以后回车，如图 4 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。

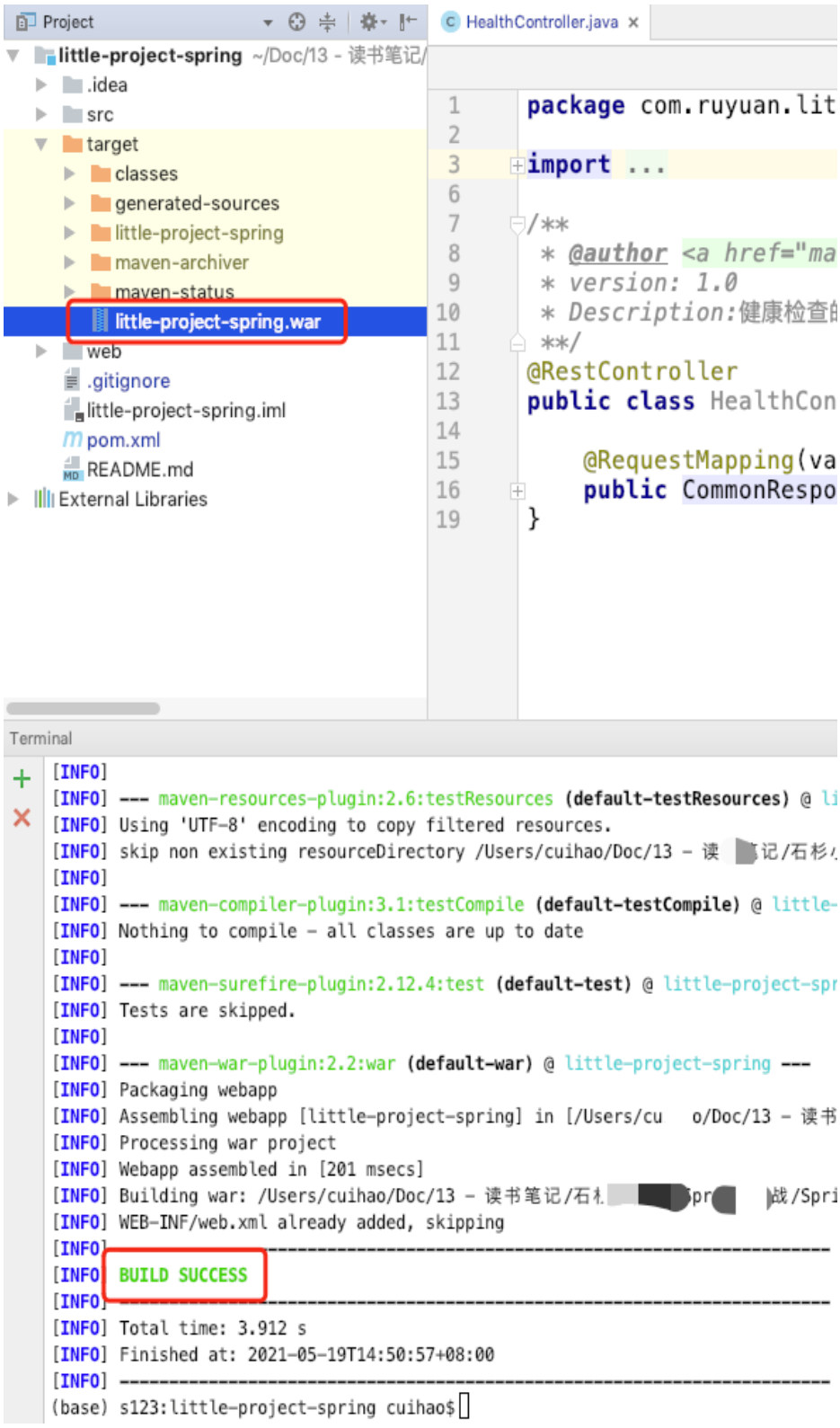
Project
little-project-spring ~/Doc/13 - 读书笔记/
.idea
src
target
classes
generated-sources
little-project-spring
maven-archiver
maven-status
little-project-spring.war
web
.gitignore
little-project-spring.iml
pom.xml
README.md
External Libraries
HealthController.java
package com.ruyuan.lit
import ...
/**
* @author <a href="ma
* version: 1.0
* Description:健康检查
**/
@RestController
public class HealthCon
@RequestMapping(va
public CommonRespo
}
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 记/石杉
[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/石杉 pr 战/Spri
[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 4 打包成功

4、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。

5、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 5 所示的“started java process”字样的时候，就说明服务启动成功了。但是随后的健康检查服务一直在重试，由于我们在 health 方法里面强行抛出了 exception 导致调用这个方法的时候报错，返回 500 错误。

```
[root@iZuf69caqdflvzby0x0dlvZ little-project-spring]# sh deploy.sh restart
stop java process
        -- stopping java lasts 59 seconds.
java process has exited

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.ja
r:/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 500
Wait app to pass health check: 2...
code is 500
Wait app to pass health check: 3...
```

图 5 启动服务成功

如图 6 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。可以看到返回 500 的错误，这个就是由于我们抛出异常造成的，显然将这个错误信息暴露给用户是不友好的，因此我们需要采取统一的异常处理机制来解决这个问题。

图 6 访问健康服务返回错误

6、总结

本节课在 Controller 中的 health 方法里 throw 了一个 new exception，通过这种方式模拟系统内部抛出异常的情况，按照目前代码而言如果没有处理这个异常将会直接展示给用户 500 错误，显然是不友好的。因此，我们需要对系统内部异常处理采取统一的处理，从而才能发挥给前端页面有价值的信息。下节课我们会讲到基于 Spring AOP 机制，实现全局异常处理。下期见，拜拜。