第二题plus

4

Classpath代表类的字节码文件存放的目录,可以告诉编译器到哪里去找字节码文件。

报错,ClassNotFoundException

按照顺序找, 一旦找到就停止, 不会再找下一个。

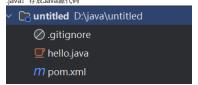
5

打个比喻就是,jar是包含Java资源的容器,classpath则是Java运行环境中用于定位这些资源的导航。 /META-INF/MANIFEST.MF 文件也叫做清单文件,包含了zhenggejar包的一系列属性(包括版本信息等) 作用就是基于提供这些信息来完成的,例如可以帮助管理Jar文件的元数据,指定程序的主类,让JVM根据这个信息找到并运行应用等等

7

1当b包变化的时候,所有依赖b包的包都需要更新 2必须要配置了正确的Classpath 3添加到classpath虽然简单,但随着项目变得复杂,如1所说,可能会引起冲突。

10



11

Maven 生命周期主要分为三部分: clean (清理) ,default (默认) ,site (站点)

clean:

用于清理项目,相当于为了default周期要做的事情做准备。分为三个部分pre-clean: 清理前的准备工作;

clean: 清理上一次构建的结果; post-clean: 清理结束后需要完成的工作。

default

定义了项目真正构建时所需要的所有步骤,它是所有生命周期中最核心,最重要的部分。

阶段	描述
validate	验证项目是否正确以及所有必要信息是否可用。
initialize	初始化构建状态。
generate-sources	生成编译阶段需要的所有源码文件。
process-sources	处理源码文件,例如过滤某些值。
generate-resources	生成项目打包阶段需要的资源文件。
process-resources	处理资源文件,并复制到输出目录,为打包阶段做准备。
compile	编译源代码,并移动到输出目录。
process-classes	处理编译生成的字节码文件
generate-test-sources	生成编译阶段需要的测试源代码。
process-test-sources	处理测试资源,并复制到测试输出目录。
test-compile	编译测试源代码并移动到测试输出目录中。
test	使用适当的单元测试框架(例如 JUnit)运行测试。
prepare-package	在真正打包之前,执行一些必要的操作。
package	获取编译后的代码,并按照可发布的格式进行打包,例如 JAR、WAR 或者 EAR 文件。
pre-integration-test	在集成测试执行之前,执行所需的操作,例如设置环境变量。
integration-test	处理和部署所需的包到集成测试能够运行的环境中。
post-integration-test	在集成测试被执行后执行必要的操作,例如清理环境。
verify	对集成测试的结果进行检查,以保证质量达标。
install	安装打包的项目到本地仓库,以供其他项目使用。
deploy	拷贝最终的包文件到远程仓库中,以共享给其他开发人员和项目。 CSDN @怪 咖@

site:

目的是建立和部署项目站点,起一个辅助作用,对于维护项目,跟踪项目进展等很有用。

pre-site: 准备阶段。在生成站点前所需要做的工作;

site: 生成站点阶段;

post-site: 结束阶段。生成站点结束后所需要做的工作;

site-deploy:发布阶段。我们可以将上面生成的站点发布到对应服务器中。

目录结构

pom.xml 用于maven的配置文件 /src 源代码目录 /src/main 工程源代码目录 /src/main/java 工程java源代码目录

/src/main/resource 工程的资源目录

/src/test 单元测试目录

/src/test/java

/target 输出目录,所有的输出物都存放在这个目录下 /target/classes 编译之后的class文

12

作用: 查看一些信息,管理依赖, 引入依赖包, 查看依赖清单

```
1 <!--project是pom.xml的根元素,还声明了一些POM相关的命名空间以及xsd元素-->
2 project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
3
   http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <!--制定了当前POM模型的版本,目前版本只是4.0.0-->
    <modelVersion>4.0.0</modelVersion>
    <! --坐标-->
6
    <! --属于哪个组, 一般是项目所在组织或公司域名的倒序-->
8
    <groupId>com.videoclass
    <! -- 定义当前项目在组中的唯一ID-->
9
10
    <artifactId>maven-single</artifactId>
11
    <! -- 定义项目当前的版本-->
12
    <version>1.0-SNAPSHOT</version>
13
    <!--项目产生的制品的类型,有jar, pom, war等-->
14
     <packaging>jar</packaging>
    <! --项目的名称-->
15
16
    <name>maven-single</name>
17
    <! --项目的官网-->
18
    <url>http://maven.apache.org</url>
19
                               I
20
    <! -- 定义的属性变量-->
21
    properties>
22
      project.build.sourceEncoding>UTF-8/project.build.sourceEncoding>
23
      <junit.version>3.8.1</junit.version>
24
    </properties>
25
     <! --定义的依赖清单-->
26
27
     <dependencies>
28
      <dependency>
29
        <groupId>junit
        <artifactId>junit</artifactId>
30
        <version>${junit.version}</version>
31
32
        <scope>test</scope>
33
      </dependency>
34
     </dependencies>
```