1.使用java命令运行时jar包指定main函数所在的类

windows下(使用分号分割在classpath下的jar包)

java –cp <目录名>;x1.jar;xx.jar (主方法类)类的全路径

linux下(使用冒号分割在classpath下的jar包)

java –cp dirName:x1.jar:xx.jar (主方法类)类的全路径 （如果加载配置文件的目录，请放到第一位，否则会出现null的情况）

2.maven中scope的各个含义

* compile：默认值 他表示被依赖项目需要参与当前项目的编译，还有后续的测试，运行周期也参与其中，是一个比较强的依赖。打包的时候通常需要包含进去
* test：依赖项目仅仅参与测试相关的工作，包括测试代码的编译和执行，不会被打包，例如：junit
* runtime：表示被依赖项目无需参与项目的编译，不过后期的测试和运行周期需要其参与。与compile相比，跳过了编译而已。例如JDBC驱动，适用运行和测试阶段
* **provided：打包的时候可以不用包进去**，别的设施会提供。事实上该依赖理论上可以参与编译，测试，运行等周期。相当于compile，但是打包阶段做了exclude操作
* system：从参与度来说，和provided相同，不过被依赖项不会从maven仓库下载，而是从本地文件系统拿。需要添加systemPath的属性来定义路径

3.maven打包时把时添加依赖的插件

<**plugin**>  
 <**artifactId**>maven-assembly-plugin</**artifactId**>  
 <**configuration**>  
 <**archive**>  
 <**manifest**>  
 <**mainClass**></**mainClass**>  
 </**manifest**>  
 </**archive**>  
 <**descriptorRefs**>  
 <**descriptorRef**>jar-with-dependencies</**descriptorRef**>  
 </**descriptorRefs**>  
 </**configuration**>  
</**plugin**>

使用 mvn assembly:assembly进行打包

哪些依赖不需要打进包里，设置scope为provide

4.maven中配置多个编译目录

<**plugin**>  
 <**groupId**>org.codehaus.mojo</**groupId**>  
 <**artifactId**>build-helper-maven-plugin</**artifactId**>  
 <**version**>1.10</**version**>  
 <**executions**>  
 <**execution**>  
 <**id**>add-source</**id**>  
 <**goals**>  
 <**goal**>add-source</**goal**>  
 </**goals**>  
 <**configuration**>  
 <**sources**>src/main/java</**sources**>  
 <**sources**>src/main/scala</**sources**>  
 </**configuration**>  
 </**execution**>  
 </**executions**>  
</**plugin**>

5.maven的scala编译插件

<!--scala的编译插件-->  
<**plugin**>  
 <**groupId**>net.alchim31.maven</**groupId**>  
 <**artifactId**>scala-maven-plugin</**artifactId**>  
 <**version**>3.2.2</**version**>  
 <**configuration**>  
 <**recompileMode**>incremental</**recompileMode**>  
 </**configuration**>  
 <**executions**>  
 <**execution**>  
 <**goals**>  
 <**goal**>compile</**goal**>  
 <**goal**>testCompile</**goal**>  
 </**goals**>  
 </**execution**>  
 </**executions**>  
</**plugin**>

6.首次向github上的仓库提交代码

在需要提交的目录中打开git bash

按照以下输入

**git init //初始化仓库**

**git add fileName/dirName //添加到暂存区**

**git commit –m “description” //提交**

**git remote add origin** [**git@github.com:GhhuangYn/masterBigdata.git**](mailto:git@github.com:GhhuangYn/masterBigdata.git)

**//关联远程仓库**

**git push / git push origin master //提交**

7.删除远程仓库中的文件/目录

**git rm --cached filename / -r directory**

**git commit –m “description”**

**git push**

8.重新生成SSH密钥

**ssh-keygen –t rsa –b 4096 –C “your-email@email.cm”**

**如果不想要密码，则直接回车，不要输入密码**

**在 C:/user/HuangYn-PC/.ssh/ 找到公钥 rsa\_pub 文件，复制内容到github的设置中**

9.springboot项目打包

**使用插件**

<**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 <**executions**>  
 <**execution**>  
 <**goals**>  
 <**goal**>repackage</**goal**>  
 </**goals**>  
 </**execution**>  
 </**executions**>  
 <**configuration**>  
 <**mainClass**>callLog.CallLogApp</**mainClass**>  
 </**configuration**>  
 </**plugin**>  
 </**plugins**>  
</**build**>

**打包时请移除assembly插件，用package/install命令直接打包，使用java –jar命令运行即可**

10.springboot整合thymeleaf

*/\*  
 设置视图解析  
 ThymeleafViewResolver是Spring MVC中ViewResolver的一个实现类。  
 像其他的视图解析器一样，它会接受一个逻辑视图名称，并将其解析为视图  
 \*/*@Bean  
**public** ViewResolver viewResolver(**SpringTemplateEngine** springTemplateEngine) {  
 **ThymeleafViewResolver** thymeleafViewResolver =

**new** ThymeleafViewResolver();  
 thymeleafViewResolver.setCharacterEncoding("UTF8");  
 thymeleafViewResolver.setContentType("text/html");  
 thymeleafViewResolver.setTemplateEngine(springTemplateEngine);  
 **return** thymeleafViewResolver;  
}  
  
/\*\*  
 \* 模板解析引擎  
 \* TemplateResolver会最终定位和查找模板。与之前配置InternalResourceViewResolver类似，  
 \* 它使用了prefix和suffix属性。前缀和后缀将会与逻辑视图名组合使用，进而定位Thymeleaf引擎。  
 \* 它的templateMode属性被设置成了HTML 5，这表明我们预期要解析的模板会渲染成HTML 5输出  
 \* @return  
 \*/  
@Bean  
**public** AbstractTemplateResolver templateResolver() {  
 **SpringResourceTemplateResolver** resolver =

**new** SpringResourceTemplateResolver();  
 resolver.setPrefix("classpath:/templates/");  
 resolver.setSuffix(".html");//设置后缀  
 resolver.setCacheable(**false**);//设置不缓存  
 resolver.setTemplateMode("HTML5");  
 **return** resolver;  
}  
  
//设置模板引擎  
@Bean  
**public SpringTemplateEngine** springTemplateEngine() {  
 **SpringTemplateEngine** springTemplateEngine =

**new** SpringTemplateEngine();  
 springTemplateEngine.setTemplateResolver(templateResolver());  
 **return** springTemplateEngine;  
}

11.Linux的awk命令

http://www.runoob.com/linux/linux-comm-awk.html

AWK是一种处理文本文件的语言，是一个强大的文本分析工具。

语法

awk [选项参数] 'script' var=value file(s)

或

awk [选项参数] -f scriptfile var=value file(s)

基本用法

log.txt文本内容如下：

2 this is a test

3 Are you like awk

This's a test

10 There are orange,apple,mongo

用法一：

awk '{[pattern] action}' {filenames} # 行匹配语句 awk '' 只能用单引号

实例：

# 每行按空格或TAB分割，输出文本中的1、4项

$ awk '{print $1,$4}' log.txt

---------------------------------------------

2 a

3 like

This's

10 orange,apple,mongo

# 格式化输出

$ awk '{printf "%-8s %-10s\n",$1,$4}' log.txt

---------------------------------------------

2 a

3 like

This's

10 orange,apple,mongo

用法二：

awk -F #-F相当于内置变量FS, 指定分割字符

实例：

# 使用","分割

$ awk -F, '{print $1,$2}' log.txt

---------------------------------------------

2 this is a test

3 Are you like awk

This's a test

10 There are orange apple

# 或者使用内建变量

$ awk 'BEGIN{FS=","} {print $1,$2}' log.txt

---------------------------------------------

2 this is a test

3 Are you like awk

This's a test

10 There are orange apple

# 使用多个分隔符.先使用空格分割，然后对分割结果再使用","分割

$ awk -F '[ ,]' '{print $1,$2,$5}' log.txt

---------------------------------------------

2 this test

3 Are awk

This's a

10 There apple

用法三：

awk -v # 设置变量

实例：

$ awk -va=1 '{print $1,$1+a}' log.txt

---------------------------------------------

2 3

3 4

This's 1

10 11

$ awk -va=1 -vb=s '{print $1,$1+a,$1b}' log.txt

---------------------------------------------

2 3 2s

3 4 3s

This's 1 This'ss

10 11 10s

用法四：

awk -f {awk脚本} {文件名}

实例：

$ awk -f cal.awk log.txt

12.Linux常用命令

netstat

显示详细的网络状况

# netstat -a

显示当前户籍UDP连接状况

# netstat -nu

**df**命令用于显示目前在**Linux系统上**的**文件系统**的磁盘使用情况统计。

* 文件-h, --human-readable 使用人类可读的格式(预设值是不加这个选项的...)
* 文件-a, --all 包含所有的具有 0 Blocks 的文件系统

du命令用于显示目录或文件的大小。du会显示**指定的目录或文件**所占用的磁盘空间。

u [-abcDhHklmsSx][-L <符号连接>][-X <文件>][--block-size][--exclude=<目录或文件>][--max-depth=<目录层数>][--help][--version][目录或文件]

* -a或-all 显示目录中个别文件的大小。
* -h或--human-readable 以K，M，G为单位，提高信息的可读性。
* -m或--megabytes 以1MB为单位。