1、archetype模板的介绍

artchetype其实就是个maven项目模板

比如我们之前在eclipse里面用m2eclipse插件自动创建的maven工程，就是用的maven-archetype-quickstart模板，就是给我们创建好最基础的工程结构和pom信息

（1）公司角度

在公司里，如果你是个架构师，那么需要站在架构师的视角，为公司创建一个统一的archetype模板，里面包含了一些必要的依赖和插件等等信息

举个例子吧，比如你公司用的一整套技术栈，可能就是spring boot+spring cloud+mysql+redis+zookeeper+rabbitmq+elasticsearch，包括一些zookeeper基础工具类，redis的基础工具类，mysql的一套公司的DAO框架，elasticseearch访问的基础工具类。同时还有一些maven私服部署的配置，插件的配置，那么你就可以给你的公司创建一个archetype，里面包含这些东西

你可以作为一个架构师，给你的公司，定制一个通用的项目骨架，里面包含了上面说的所有东西。任何一个项目，如果要启动，都不需要任何人手动重新去查阅资料，然后复制粘贴，构建工程，配置插件，光是搭建一个项目骨架，就要好几天的时间。

更痛苦的事情，如果类似的技术架构，不同的人用不同的方式去写一些基础工具类，版本去搭建，会导致各个项目组之间的成员，进行项目的接手和维护会比较复杂。耗时较长，比较麻烦。

archetype项目模板，直接创建一个下项目工程出来，基于此直接开发。公司里面各个项目，很多框架版本，框架整合，基础代码，代码规范和约定，都差不多。

可能你们已经做好了一个项目了，maven这一套东西都很稳定了，此时就可以完全将稳定不变，各个项目可以他通用的东西抽取出来，做成一个archetype模板

然后下次你们如果要新启动一个项目，直接用archetype模板创建出来工程即可，比如就创建一个parent父工程出来，然后那个项目中的每个服务就继承自自己项目的parent工程即可，很快速

而且各种技术的版本都是整合好的，很方便使用，不用自己花时间去想用什么版本，也许还要解决依赖冲突问题什么的，都不用管了，直接上手用就可以，很方便，还便于公司内部统一技术栈

然后每个项目自己可以在基础之上，修改自己的pom配置，或者加入其它依赖

（2）项目角度

可能比如说，类似我们这个oa的项目，之前几个模块，其实一些框架的整合啊什么的，都是差不多的，资源配置文件，也是差不多的，此时，我们刚才搭建工程，还是手工方式去搭建，然后复制黏贴，各种修改，其实是比较麻烦的。而且可能出错。

对于一个项目来说，可能会不断拆分多个模块对应的工程，也可能会有不同的服务。此时架构师完全可以针对这个项目，制定一个项目的骨架，也就是一个archetype，然后呢，项目组内的成员，每次要新构建一个工程，直接用archetype骨架来生成即可。很方便。

（3）练习

用第二种情况来举例子，其实大家学会了之后，就完全可以自己去弄了

2、常用的maven archetype

如果只是一个模块，或者不需要web容器的支持，那就是maven-archetype-quickstart即可

如果是一个web工程，那可以用maven-archetype-webapp

3、自己写一个archetype

（1）用eclipse提供的向导，常见一个maven工程，选择的archetype是maven-archetype-archetype，就是用来创建archetype的一个模板

（2）熟悉archetype自己的pom.xml

创建一个maven工程，将其改造为archetype需要的格式

archetype自己也需要一个pom.xml，用来定位自己的坐标

<groupId>com.zhss.maven.archetypes</groupId>

<artifactId>maven-archetype-parent</artifactId>

<version>1.0.0</version>

（3）要生成的项目的pom.xml

接着编写src/main/resources/archetype-resources/pom.xml，就是基于这个archetype生成的项目的pom.xml，里面要注意的是要用到一些占位符，因为生成项目的时候，groupId和artifactId之类的都是用户输入的，这个就是最终生成的项目的pom.xml文件。里面可以放入任何你需要的依赖和插件。

<groupId>${groupId}</groupId>

<artifactId>${artifactId}</artifactId>

<version>${version}</version>

<name>${artifactId}</name>

下面放入依赖、插件以及部署等通用的信息

（2）定义archetype的元数据

接着要编写src/main/resources/META-INF/maven/archetype-metadata.xml

<?xml version=”1.0” encoding=”UTF-8”?>

<archetype-descriptor name=”parent”>

<fileSets>

<fileSet filtered=”true” packaged=”true”>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.java</include>

</includes>

</fileSet>

<fileSet filtered=”true” packaged=”true”>

<directory>src/test/java</directory>

<includes>

<include>\*\*/\*.java</include>

</includes>

</fileSet>

<fileSet filtered=”true” packaged=”true”>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.properties</include>

</includes>

</fileSet>

</fileSets>

<requiredProperties>

<requiredProperty key=”port” />

<requiredProperty key=”groupId”>

<defaultValue>com.zhss</defaultValue>

</requiredProperty>

</requiredProperties>

</archetype-descriptor>

这个文件两个作用，第一个作用是定义将这个archetype中的哪些java代码和测试代码以及资源文件，都包含到创建好的项目中去；第二个作用是定义创建项目的时候需要输入的参数是什么

（3）编写要给新项目的java代码、测试代码以及资源包和配置文件s

src/main/java之类的，对应的目录是src/main/resources/archetype-resources/src/main/java子目录中的代码，在这里可以写你要预先给好的代码

src/test/java，对应的目录是src/main/resources/archetype-resources/src/test/java子目录中的代码，就是你预先给好的测试代码

src/main/resources之类的，对应的目录是src/main/resources/archetype-resources/src/main/resources目录，在这里给好你预先包含的配置文件

filtered参数表示是否对文件启用占位符替换，packaged表示是否将文件放到包路径下。因为package是创建项目的时候必须输入的，就是你的包基础路径。一般是需要将代码放到包路径下的，而资源文件不需要放到包路径下。

默认情况下，会要求输入groupId，artifactId，version，package，也可以自己额外定义要求输入的参数，都可以在资源文件或者代码中使用${}的占位符方式来引用

同时对于你写的一些基础的类，一般是可以用package占位符的，到时候创建出来就会替换package

比如说

package ${package}

public class Application {

}

（4）部署arthcetype到私服

mvn clean deploy安装到本地仓库和私服，都可以使用了

（5）用archetype创建一个工程

然后就可以用

mvn archetype:generate -DarchetypeGroupId=com.zhss.archetypes -DarchetypeArtifactId=archetype-oa -DarchetypeVersion=1.0.0

然后会提示输入各种参数，就会生成一个项目

3、archetype catalog

也可以做到不需要输入archetype的坐标就可以使用，就是要将archetype加入一个archetype列表供用户选择，这个archetype列表就是在archetype-catalog.xml文件中。

maven默认会从几个地方去读取archetype-catalog.xml的内容：

（1）internal：maven-archetype-plugin内置了几十个archetype

（2）local：从~/.m2/archetype-catalog.xml读取，默认不存在，要自己创建

（3）remote：读取maven中央仓库的archetype-catalog.xml，大概有几百个archetype

（4）file：读取本机任何一个文件

（5）http：读取任何一个网络上的文件

默认maven会从local+remote去读取archetype列表供选择

可以用mvn archetype:crawl来自动化扫描本地仓库中的archetype，然后生成一份archetype-catalog.xml，放在~/.m2/目录下，但是一般不用这种方式。

除非呢你是要做一个开源的archetype，对不对，然后才去关注这个东西，或者是你写了一个archetype，要整合到eclipse里面去，这种都很麻烦，冷门的知识，我都不想讲。

853769620