Maven的常用属性设置

1、全局变量设置

在Maven的pom.xml文件中，<properties>标签用来定义全局变量

--如maven的属性设置，如下：
其中<project.build.sourceEncoding>表示的是属性名或者说是一个变量名，UTF-8表示该属性的值

```
  <properties>
    <!--设置Maven构建项目时用的编码方式，设为utf-8避免中文乱码-->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <!--设置Maven编译代码时用的JDK版本-->
    <maven.compiler.source>1.7</maven.compiler.source>
    <!--设置Maven运行程序时用的JDK版本-->
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
```

```
--Maven全局变量设置
    1、自定义属性
    在<properties>标签中通过自定义标签声明变量（标签名就是变量名），然后把值赋上去
    2、使用
    在pom文件中的其他位置，如依赖中使用"${自定义标签名}"来表式变量值

--这样做的优点和作用如下展示：
有时候我们需要用到一个框架，例如spring框架，这个框架有许多的项目，我们的项目只需要spring框架中的一部分项目，所以我们需要在依赖中添加spring框架各个项目的坐标依赖，我们要保证这些项目的版本一致，那么坐标中的version属性都是一样的，所以我们自定义一个属性名为<spring.version>的变量，赋上值，然后在spring各项目的依赖的坐标的version属性中用${spring.version}代替就行了，到时候需要修改使用版本的时候在properties中修改对应属性值就可以了，非常省事。
--所以自定义全局变量一般是定义依赖的版本号
```

```
 <properties>
    <!--设置Maven构建项目时用的编码方式，设为utf-8避免中文乱码-->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <!--设置Maven编译代码时用的JDK版本-->
    <maven.compiler.source>1.7</maven.compiler.source>
    <!--设置Maven运行程序时用的JDK版本-->
    <maven.compiler.target>1.7</maven.compiler.target>
    <!--自定义属性，spring框架版本-->
    <spring.version>5.2.5</spring.version>
  </properties>

<dependencies>
    <dependency>
        <groupId>org.springframwork</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
```

```
        <groupId>org.springframwork</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.version}</version>
    </dependency>

    <dependency>
        <groupId>org.springframwork</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
```

2、指定资源位置插件设置

```
<build>
   <resources>
     <resource>
       <!--表示编译时在src/main/java目录下的-->
       <directory>src/main/java</directory>
       <!--所有.properties、.xml类型的文件可以被扫描到-->
       <includes>
         <include>**/*.properties</include>
         <include>**/*.xml</include>
       </includes>
       <!--filtering的值false表示不启用过滤器，上面include已经是过滤操作了-->
       <filtering>false</filtering>
     </resource>
   </resources>
 </build>
```

如上面build属性所示:
1、在上面没有设置resources属性时，maven执行编译时，默认只把resources中的所有文件拷贝到生成的target目录下的classes目录中，而如果src/main/java/com/studymyself目录中除了有java文件外还有一个a.propertis文件，maven是不予理睬的。
2、如果有了上面的设置，Maven进行编译时就会把要求的文件类型扫描到，并且拷贝到对应目录下，1中就会将a.properties文件拷贝到target/classes/com/studymyself目录下，并不是拷贝到classes下。
--这样做是因为有时候开发，有些文件不一定会放在resources目录中，而是放到java目录中的一些程序包下。这就需要配置resources属性。