# 配置三套环境

正式，测试，本地三套环境的配置文件。便于切换使用环境。

## 实际项目的运用

Pom.xml

<build>

<!-- 开发组约定 配置文件 三套环境 begin -->

<resources>

<resource>

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

<includes>

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

<include>\*\*/\*.xml</include>

<include>\*\*/\*.tld</include>

</includes>

<filtering>false</filtering>

</resource>

<resource>

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

<excludes>

<exclude>profile-active/\*\*</exclude>

</excludes>

</resource>

<resource>

<directory>${runtime.env}</directory>

</resource>

</resources>

<!-- 开发组约定 配置文件 三套环境 end -->

<finalName>jx-web-api</finalName>

</build>

<profiles>

<profile>

<id>local</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<runtime.env>src/main/resources/profile-active/local</runtime.env>

</properties>

</profile>

<profile>

<id>dev</id>

<properties>

<runtime.env>src/main/resources/profile-active/dev</runtime.env>

</properties>

</profile>

<profile>

<id>prod</id>

<properties>

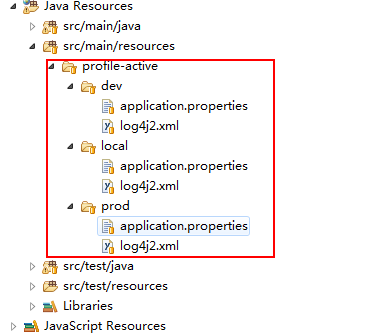
<runtime.env>src/main/resources/profile-active/prod</runtime.env>

</properties>

</profile>

</profiles>

项目工程目录



解释上述pom.xml的配置

前提基本知识：

Maven

Src/main/java目录下只有.java文件编译时会被复制输出到target/classes目录下，其他文件是不会处理的。

Src/main/resources目录默认是所有的配置文件（.xml,.properties），但是不包括图片之类，css,js的静态资源。默认编译时全部原样不处理复制到target/classess。

那么，当mybatis的mapper.xml文件位于src/main/java目录下，或者Src/main/resources下存在配置文件使用maven变量的这种情况怎么办呢？

说明如下。



<directory>：指明资源文件的目录。Eg.指明src/main/java目录下有资源文件，需要输出到目录target/classes目录下。

<include>:指明包含的此文件属于资源文件。

<exclude>:指明包含的此文件不属于资源文件。

<filtering>:即是否需要解析文件的{key},当配置文件使用了maven变量（maven默认只有解析在pom.xml使用的maven变量），那么就需要使用filtering标签。）

当使用了filtering=false,会把不需要过滤的文件输出到target/classes.

当使用了filtering=true,会把需要过滤的文件输出到target/classes.

上图表示：

src/main/java目录下存在资源文件，后缀为.properties,.xml,.tld.但是这些文件不需要maven过滤。不需要过滤的文件直接编译时候直接复制到target/classes.

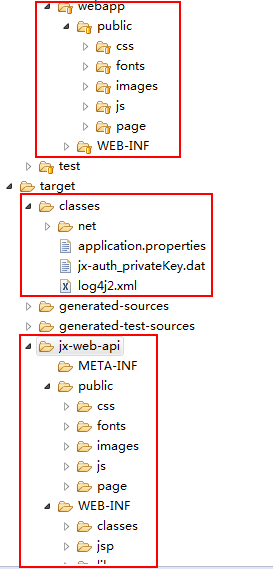
# 2.resources的详解

使用场景：

1. mybatis文件mapper.xml位于src/main/java目录下。（maven 默认情况下值加载src/main/java下的.java文件），所以需要指明资源文件位置
2. src/main/resources 目录下,

* 资源类型
  + 配置文件（properties,xml）

1. 编译存放位置：classpath类路径下，默认字target/classes
2. War包实际位置： WEB-INF/classes
3. 默认：不被maven过滤
   * web静态资源文件（css,js）
4. 编译存放位置：不在classpath类路径上。
5. 默认：不被maven过滤



# <resources>

## <resource> 指定资源文件,指定的打包时会输出到target/classes(默认位置)

* Resource作用：指定哪些文件需要通过maven-resource-plugin复制到目标输出目录下（默认为target/classes）。
* Filter=true:当为true时，会把需要过滤的文件解析复制到输出目录下。同时，解析文件中的{key}
* Filter=false: 当为false时，会把不需要过滤的文件解析复制到输出目录下。不解析文件.

<resources>

<resource>

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

<filtering>true</filtering>

<includes>

<include>context.xml</include>

</includes>

</resource>

<resource>

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

<filtering>false</filtering>

<excludes>

<exclude>context.xml</exclude>

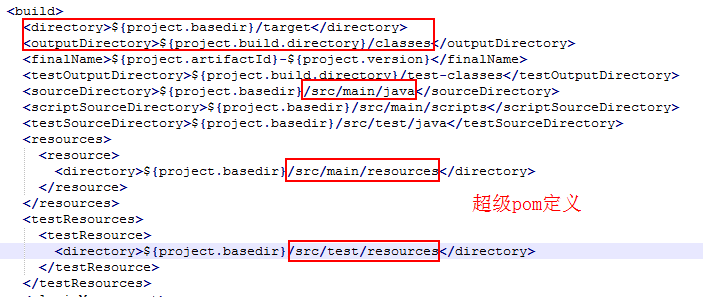
</excludes>

</resource>

第一个resource:表明，仅仅过滤src/main/resources下的文件context.xml，其他文件不过滤。（这一步：只会解析context.xml文件，同时只复制这一个文件到输出目录）

第二个resource: 表明，除了context.xml，src/main/resources下其他的文件都不过滤。（这一步：不解析resource下除了context.xml的其他文件，同时把这些其他文件复制到输出目录下，context.xml不会复制到输出目录。）

So,上两步缺一不可。如果只包含第一个resource,那么输出目录只有context.xml



# <filter> 指定是否解析文件，即解释文件其中的{key}

http://www.cnblogs.com/pixy/p/4798089.html

构建Maven项目的时候，如果没有进行特殊的配置，Maven会按照标准的目录结构查找和处理各种类型文件。

**src/main/java和src/test/java**

这两个目录中的所有\*.java文件会分别在comile和test-comiple阶段被编译，编译结果分别放到了target/classes和targe/test-classes目录中，但是这两个目录中的其他文件都会被忽略掉。

**src/main/resouces和src/test/resources**

这两个目录中的文件也会分别被复制到target/classes和target/test-classes目录中。

**target/classes**

打包插件默认会把这个目录中的所有内容打入到jar包或者war包中。

## ****Maven项目的标准目录结构****

* src
  + main
    - **java**         源文件
    - **resources**    资源文件
    - filters   资源过滤文件
    - config   配置文件
    - scripts   脚本文件
    - webapp   web应用文件
  + test
    - **java**    测试源文件
    - resources    测试资源文件
    - filters    测试资源过滤文件
  + it       集成测试
  + assembly    assembly descriptors
  + site    Site
* target
  + generated-sources
  + classes
  + generated-test-sources
  + test-classes
  + xxx.jar
* **pom.xml**
* LICENSE.txt
* NOTICE.txt
* README.txt