第三章 Maven项目管理

-Maven(三)2018/11/20 [泽林.王峰]

授课目标

1、 上章回顾

2、 Maven分模块开发（非常重要）-idea版本

3、 私服（nexus）搭建（了解）

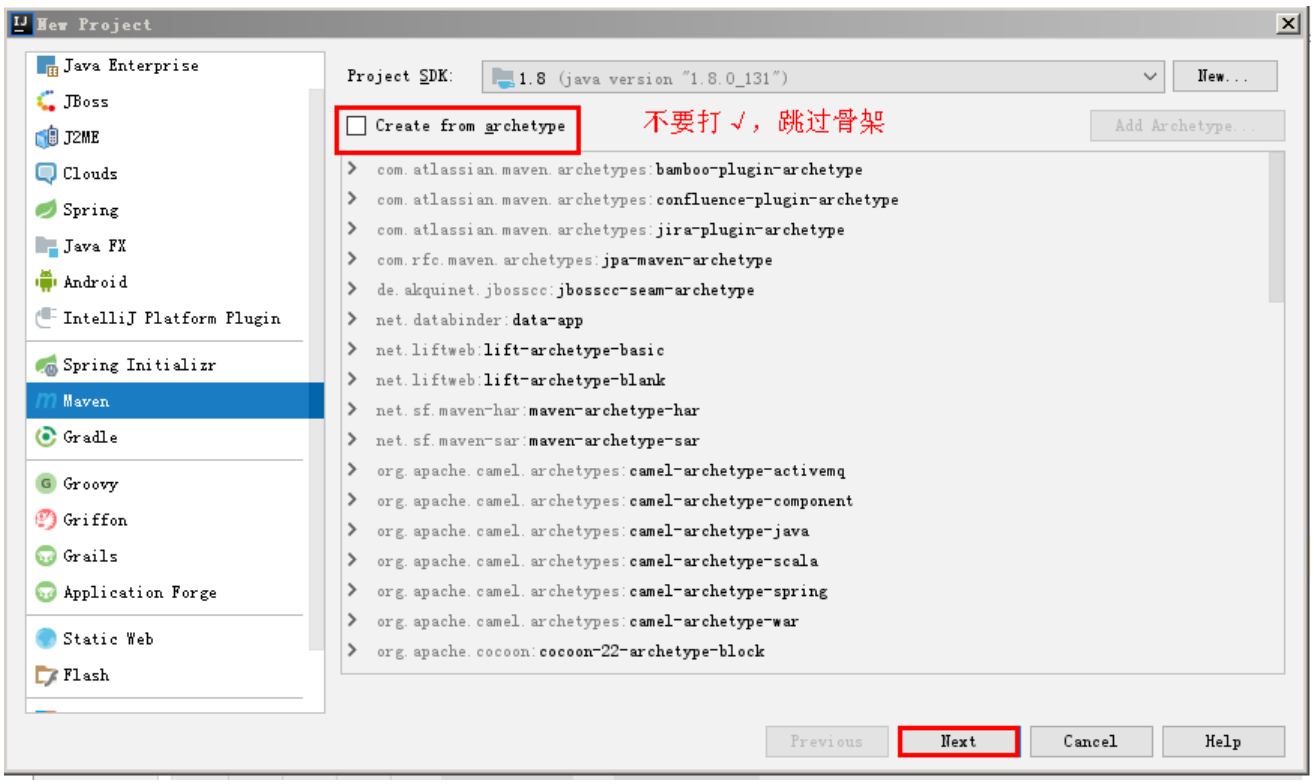
授课内容

1、 上章回顾

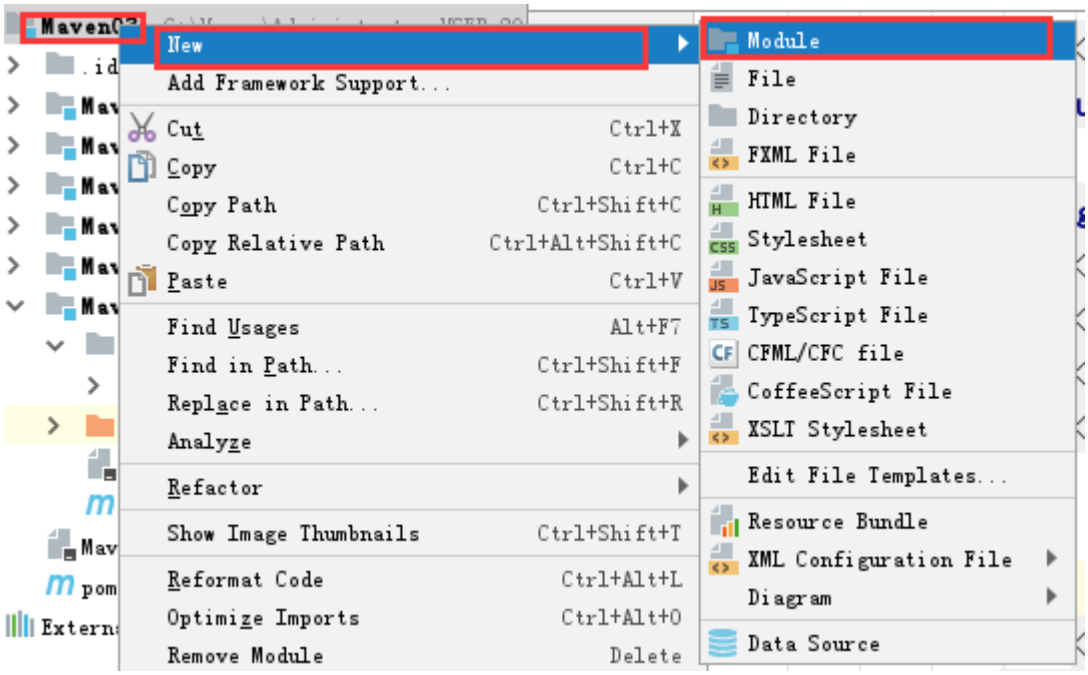
2、Maven分模块开发（非常重要）-idea版本

第一步：新建Maven工程及其下的各个模块：

1.1）新建Maven工程（不选择骨架）



1.2）在此工程下新建Maven-Pojo模块（与上面一样，选择跳转骨架）



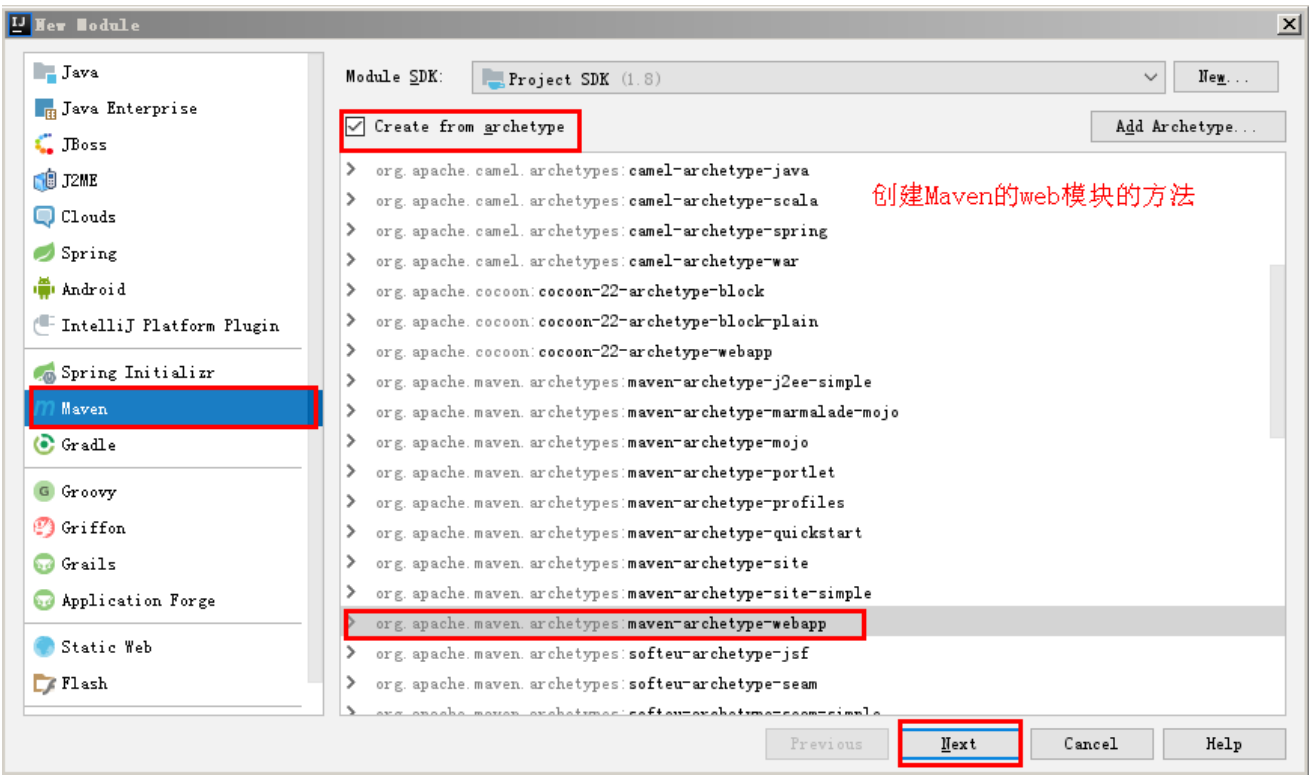
1.3）在此工程下新建Maven-Interface模块（与上面一样，选择跳转骨架）

1.4）在此工程下新建Maven-Dao模块（与上面一样，选择跳转骨架）

1.5）在此工程下新建Maven-Service模块（与上面一样，选择跳转骨架）

1.6）在此工程下新建Maven-Commons模块（与上面一样，选择跳转骨架）

1.7）在此工程下新建Maven-Web模块



第二步：在各个模块之间建立依赖关系：

2.1) 在Maven-Interface模块的pom.xml文件中添加对Maven-Pojo的依赖

```
<!--添加对pojo模块的依赖-->
    <dependencies>
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_POJO</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
        <!--添加对Commons模块的依赖-->
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_Commons</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

2.2) 在Maven-Dao模块的pom.xml文件中添加对Maven-Pojo的依赖

```
<dependency>
     <groupId>com.zelin</groupId>
     <artifactId>Maven_POJO</artifactId>
     <version>0.0.1-SNAPSHOT</version>
</dependency>
```

2.3) 在Maven-Service模块的pom.xml文件中添加对Maven-Interface及Maven-Dao的依赖

```
<!-- 引入服务接口模块的依赖 -->
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_Interface</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
        <!--引入Dao模块的依赖-->
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_DAO</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
```

2.4) 在Maven-Web模块的pom.xml文件中添加对Maven-Service模块的依赖：

```
<dependency>
           <groupId>com.zelin</groupId>
           <artifactId>Maven_Service</artifactId>
           <version>0.0.1-SNAPSHOT</version>
</dependency>
```

第三步：添加各个模块中对外部jar的依赖

3.1) 在Maven-Dao模块的pom.xml文件中依赖：(数据库访问相关)

```
【Maven-Dao的pom.xml】
<?xml version="1.0" encoding="UTF-8"?>
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>Maven03</artifactId>
        <groupId>com.zelin</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Maven_DAO</artifactId>

    <dependencies>
        <!-- 日志处理 -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
        </dependency>
        <!-- Mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
        </dependency>
        <dependency>
            <groupId>com.github.miemiedev</groupId>
            <artifactId>mybatis-paginator</artifactId>
        </dependency>
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
        </dependency>
        <!-- MySql -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- 连接池 -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>

        </dependency>
```

```
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_POJO</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

    </dependencies>
</project>
```

3.2) 在Maven-Serivce模块的pom.xml文件中依赖：(Spring访问相关)

```
【Maven-Service的pom.xml】：
<?xml version="1.0" encoding="UTF-8"?>
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>Maven03</artifactId>
        <groupId>com.zelin</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Maven_Service</artifactId>

    <dependencies>
        <!-- Jackson Json处理工具包 -->
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
        <!-- Spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>

            <artifactId>spring-aspects</artifactId>
```

```
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jms</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
        </dependency>
        <!-- 引入服务接口模块的依赖 -->
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_Interface</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
        <!--引入Dao模块的依赖-->
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_DAO</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

    </dependencies>
</project>
```

3.3) 在Maven-Web模块的pom.xml文件中依赖：

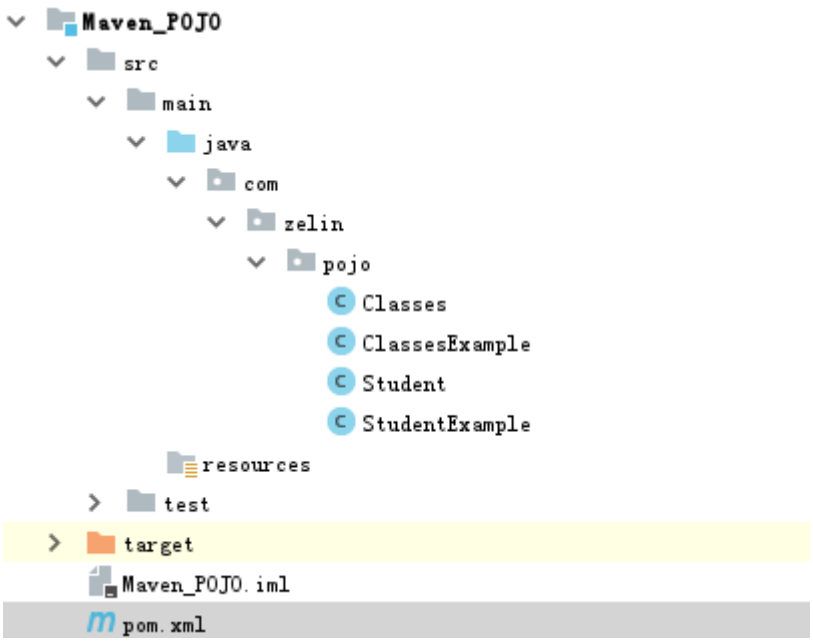
```
<?xml version="1.0" encoding="UTF-8"?>

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>Maven03</artifactId>
        <groupId>com.zelin</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>Maven_Web</artifactId>
    <packaging>war</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.7</maven.compiler.source>
        <maven.compiler.target>1.7</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>

            <groupId>junit</groupId>
```

```
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.zelin</groupId>
            <artifactId>Maven_Service</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>

    <build>
        <finalName>Maven_Web</finalName>
        <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
            <plugins>
                <plugin>
                    <artifactId>maven-clean-plugin</artifactId>
                    <version>3.0.0</version>
                </plugin>
                <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_war_packaging -->
                <plugin>
                    <artifactId>maven-resources-plugin</artifactId>
                    <version>3.0.2</version>
                </plugin>
                <plugin>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.7.0</version>
                </plugin>
                <plugin>
                    <artifactId>maven-surefire-plugin</artifactId>
                    <version>2.20.1</version>
                </plugin>
                <plugin>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>3.2.0</version>
                </plugin>
                <plugin>
                    <artifactId>maven-install-plugin</artifactId>
                    <version>2.5.2</version>
                </plugin>
                <plugin>
                    <artifactId>maven-deploy-plugin</artifactId>
                    <version>2.8.2</version>
                </plugin>

                <!-- 资源文件拷贝插件 -->
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-resources-plugin</artifactId>
                    <version>2.7</version>

                    <configuration>
```

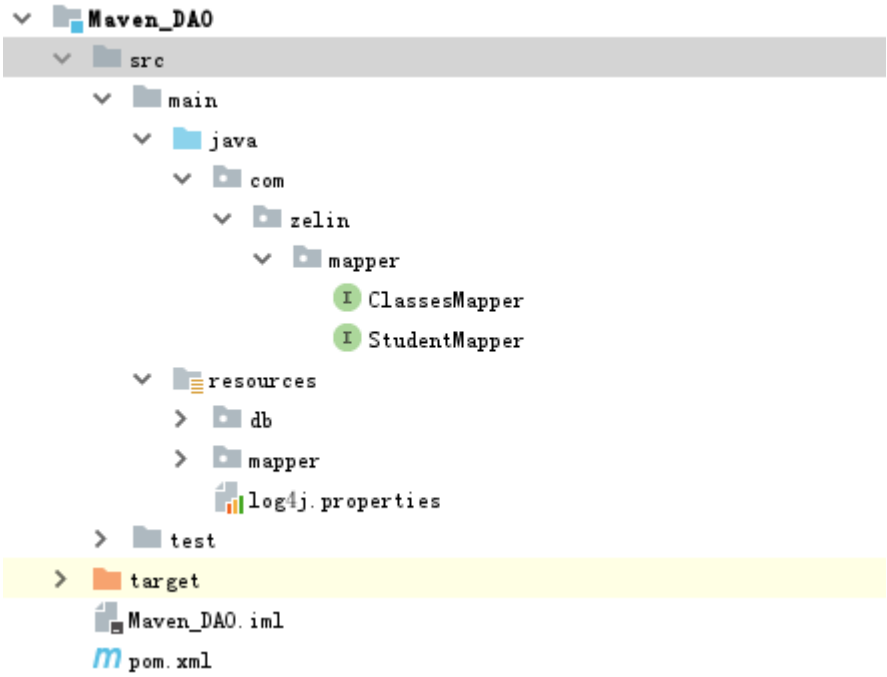
```
                        <encoding>UTF-8</encoding>
                    </configuration>
                </plugin>
                <!-- 配置Tomcat插件 -->
                <plugin>
                    <groupId>org.apache.tomcat.maven</groupId>
                    <artifactId>tomcat7-maven-plugin</artifactId>
                    <version>2.2</version>
                    <configuration>
                        <port>9000</port>
                        <path>/</path>
                        <contextReloadable>true</contextReloadable>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>

```

第四步：复制MyBatis自动生成相关的资源及自定义的配置文件

4.1) 复制MyBatis自动生成的pojo类到Maven-Pojo模块下：



4.2) 复制MyBatis自动生成的interface接口及自定义配置到MavenDao模块下：



> 注意：
>
> 1. 上图中的文件夹db中存放的是数据库访问相关字符串资源。内容大致如下：

```
db.driver=com.mysql.jdbc.Driver
db.url=jdbc:mysql:///java1301?useUnicode=true&amp;characterEncoding=utf-8
db.user=root
db.password=123
```

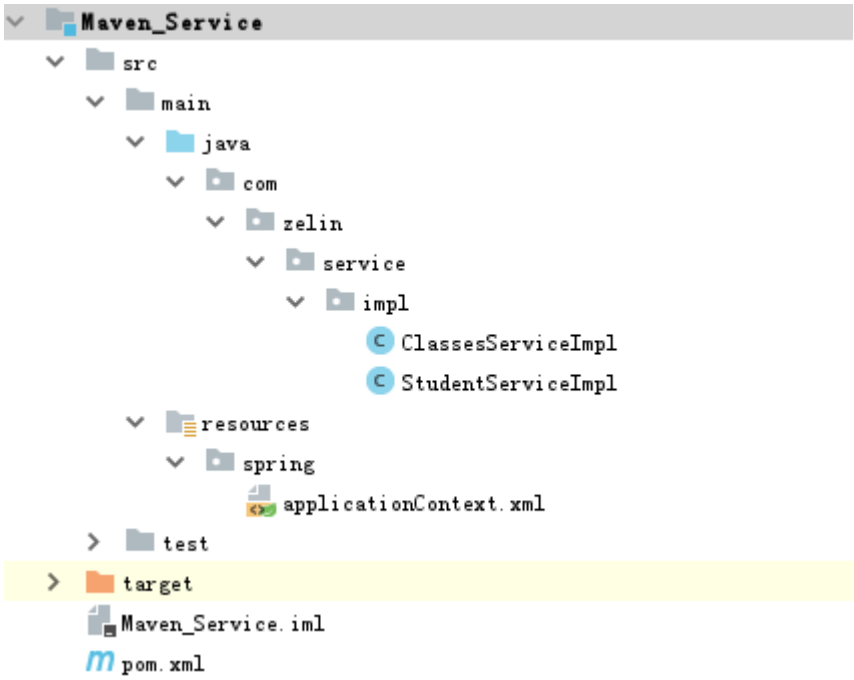
> 2. 生成的Mapper接口如StuentMapper.java如下：

```java
public interface StudentMapper {
    long countByExample(StudentExample example);

    int deleteByExample(StudentExample example);

    int deleteByPrimaryKey(Integer sid);

    int insert(Student record);

    int insertSelective(Student record);

    List<Student> selectByExample(StudentExample example);

    Student selectByPrimaryKey(Integer sid);

    int updateByExampleSelective(@Param("record") Student record, @Param("example") StudentExample example);

    int updateByExample(@Param("record") Student record, @Param("example") StudentExample example);

```

```
    int updateByPrimaryKeySelective(Student record);

    int updateByPrimaryKey(Student record);
}
```

3. 自动生成的Mapper映射文件略过。

4.3) 在Maven-Service模块定义配置文件及Service的实现，效果如下：

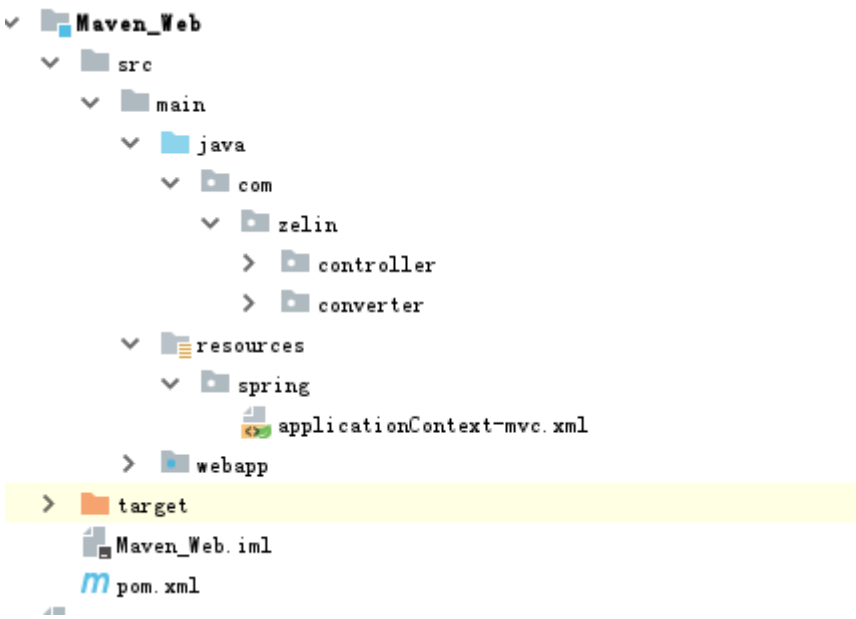


注意：其中的配置文件applictionContext.xml文件的内容大致如下：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd">

    <!-- 读取属性文件 -->
    <context:property-placeholder location="classpath*:db/db.properties"/>
    <!-- 配置包扫描器 -->
    <context:component-scan base-package="com.zelin.service.impl"/>
    <!-- 配置数据源 -->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${db.driver}"/>
        <property name="url" value="${db.url}"/>
        <property name="username" value="${db.user}"/>
        <property name="password" value="${db.password}"/>
    </bean>
    <!-- 配置SqlSessionFactoryBean -->
    <bean id="sqlSessionFactoryBean"
```

```
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="typeAliasesPackage" value="com.zelin.pojo"/>
        <property name="plugins">
            <list>
                <bean class="com.github.pagehelper.PageHelper">
                    <property name="properties">
                        <value>
                        dialect=mysql
                        </value>
                    </property>
                </bean>
            </list>
        </property>
        <property name="mapperLocations" value="classpath*:mapper/*.xml"/>
    </bean>
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.zelin.mapper"/>
    </bean>
</beans>

```

4.4) 在Maven-Web模块定义配置文件及Controller的实现，效果如下：



> **注意：**
>
> 1. 其中的applicationContext-mvc.xml文件的内容大致如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd
        http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd">
    <context:component-scan base-package="com.zelin.controller"/>
    <mvc:annotation-driven/>
</beans>
```

2. 其中的web.xml文件的内容大致如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
  <display-name>Maven_Web</display-name>
  <!-- 1.加载spring所有的配置文件（不包含springmvc的配置文件） -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <!-- 注意：
        classpath与classpath*区别：
        （1）classpath:代表访问指定类路径下的资源文件
        （2）classpath*:代表访问指定类路径及引入的第三方包中的资源文件。
     -->
    <param-value>classpath*:spring/applicationContext*.xml</param-value>
  </context-param>
  <!-- 2.配置spring的监听器 -->
  <listener>    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <!-- 3.配置DispatcherServlet -->
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring/applicationContext-mvc.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
```

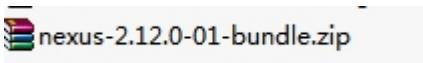
```
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>

  <!-- 4.处理中文乱码的过滤 器 -->
  <filter>
    <filter-name>characterEncoding</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>characterEncoding</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

4.5）最终运行效果如下：

3、 私服（nexus）搭建（了解）

4.1）安装私服nexus：

将



解压到指定的目录中，然后，进入目录中的/bin目录

4.1.1) 开始安装私服：

Nexus.bat install

4.1.2) 启动服务：

Nexus.bat start

4.1.3) 如果想卸载可使用命令：

Nexus.bat uninstall

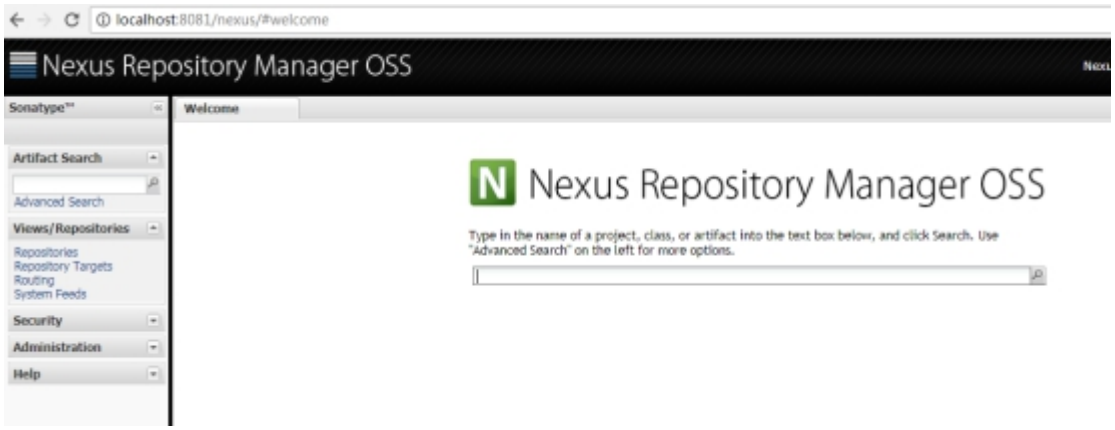
4.1.3) 找到私服安装目录下的conf文件夹，找到nexus.propertis

```
# Jetty section
application-port=8081
application-host=0.0.0.0
nexus-webapp=${bundleBasedir}/nexus
nexus-webapp-context-path=/nexus

# Nexus section
nexus-work=${bundleBasedir}/../sonatype-work/nexus
runtime=${bundleBasedir}/nexus/WEB-INF
```

端口号

上下文路径

4.1.4) 使用username:admin password:admin123这种方式登录私服。



4.2) 在maven的配置文件settings.xml文件中：

【第一部分】

```
<server>
      <id>releases</id>
      <username>admin</username>
      <password>admin123</password>
</server>
<server>
      <id>snapshots</id>
      <username>admin</username>
      <password>admin123</password>
</server>
```

【第二部分】：在profiles标签下配置

```
<profile>
    <!--profile的id-->
    <id>dev</id>
    <repositories>
      <repository>
        <!--仓库id，repositories可以配置多个仓库，保证id不重复-->
        <id>nexus</id>
        <!--仓库地址，即nexus仓库组的地址-->
        <url>http://localhost:8081/nexus/content/groups/public/</url>
        <!--是否下载releases构件-->
        <releases>
          <enabled>true</enabled>
        </releases>
        <!--是否下载snapshots构件-->
        <snapshots>
          <enabled>true</enabled>
        </snapshots>
      </repository>
    </repositories>
     <pluginRepositories>
        <!-- 插件仓库，maven的运行依赖插件，也需要从私服下载插件 -->
        <pluginRepository>
            <!-- 插件仓库的id不允许重复，如果重复后边配置会覆盖前边 -->
            <id>public</id>
            <name>Public Repositories</name>
            <url>http://localhost:8081/nexus/content/groups/public/</url>
        </pluginRepository>
    </pluginRepositories>
  </profile>

```

【第三部分】：在profiles标签外面激活上面的profile配置信息:

```
<activeProfiles>
   <activeProfile>dev</activeProfile>
 </activeProfiles>
```

4.3 ）在Maven-D这个模块的pom.xml文件中添加如下配置：

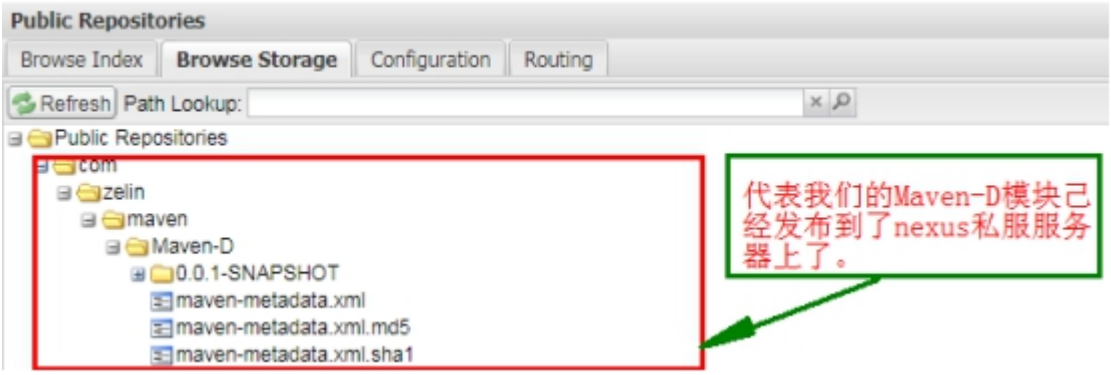
```
<distributionManagement>
   <repository>
       <id>releases</id>
       <url>http://localhost:8081/nexus/content/repositories/releases/</url>
   </repository>
   <snapshotRepository>
       <id>snapshots</id>
       <url>http://localhost:8081/nexus/content/repositories/snapshots/</url>
   </snapshotRepository>
 </distributionManagement>
```

4.4 ）将Maven-D deploy到nexus上，然后，再从本地仓库中删除Maven-D相关内容，最后，关闭Maven-D模块。

运行deploy命令后的效果如下：

```
--- maven-deploy-plugin:2.7:deploy (default-deploy) @ Maven-Dao ---
Downloading: http://localhost:8081/nexus/content/groups/public/org/codehaus/plexus/plexus-utils/1.
Downloaded: http://localhost:8081/nexus/content/groups/public/org/codehaus/plexus/plexus-utils/1.5
Downloading: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-
Uploading: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-SN
Uploaded: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-SNA
Uploading: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-SN
Uploaded: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-SNA
Downloading: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/maven-
Uploading: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-SN
Uploaded: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/0.0.1-SNA
Uploading: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/maven-me
Uploaded: http://localhost:8081/nexus/content/repositories/snapshots/com/zelin/Maven-Dao/maven-met
------------------------------------------------------------------------
```

在nexus服务器中查看效果如下：



4.5 ）运行当前的Maven-Web，效果如下：

```
[INFO] Maven-P
[INFO] Maven-D
[INFO] Maven-S
[INFO] Maven-W
[INFO] Downloading: http://localhost:8081/nexus/content/groups/public/org/codehaus/mojo/maven-metadata.xml
[INFO] Downloading: http://localhost:8081/nexus/content/groups/public/org/apache/maven/plugins/maven-metadata.xml
[INFO] Downloaded: http://localhost:8081/nexus/content/groups/public/org/codehaus/mojo/maven-metadata.xml (20 KB at
[INFO] Downloaded: http://localhost:8081/nexus/content/groups/public/org/apache/maven/plugins/maven-metadata.xml (1
[INFO] Downloading: http://localhost:8081/nexus/content/groups/public/org/codehaus/mojo/tomcat-maven-plugin/maven-m
[INFO] Downloaded: http://localhost:8081/nexus/content/groups/public/org/codehaus/mojo/tomcat-maven-plugin/maven-me
[INFO]
[INFO] ------------------------------------------------------------------------
```

代表正在从私服服务器上下载我们使用的Maven-D模块

4.6 ）程序最终运行效果同上。