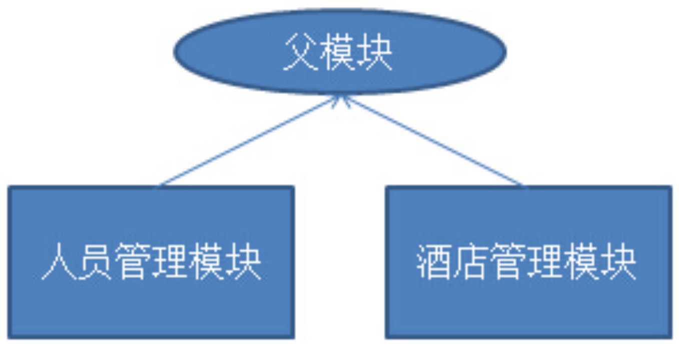
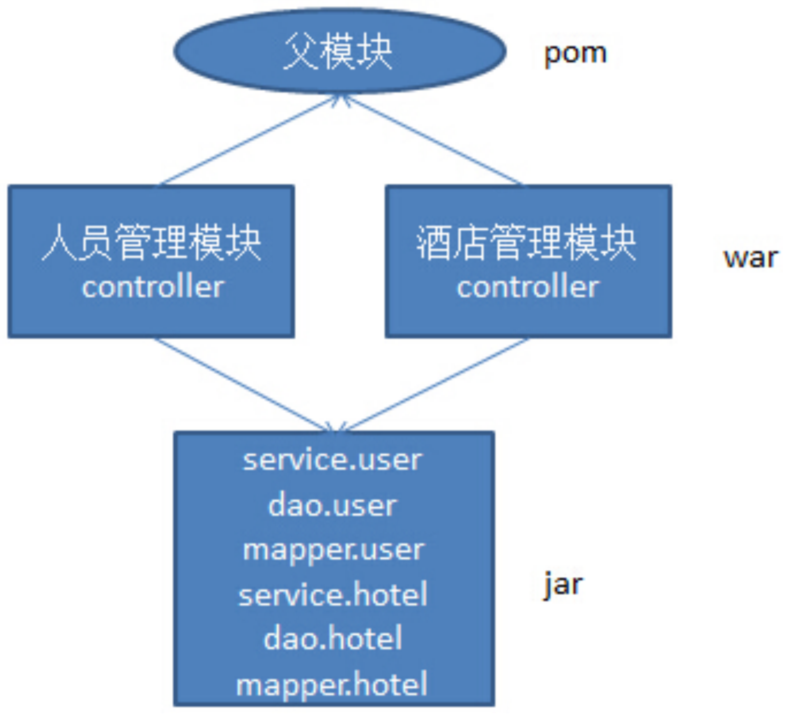
第三章 企业项目开发--企业中的项目架构以及多环境分配

1、业务模块与数据模块分离

在实际开发中，我们项目的架构业务模块和数据模块是分离的，举个例子，假设我们的项目有"人员管理模块"和"酒店管理模块"两个模块，按照上一章的介绍，我们会建立下图所示的项目结构：



其中，人员管理模块的controller、service、dao、mapper都在一个项目中，而在实际使用中，我们会将数据模块分离出来，即将以上两个子模块的service、dao、mapper拿出来，放在一个子项目中，形成如下的项目结构：



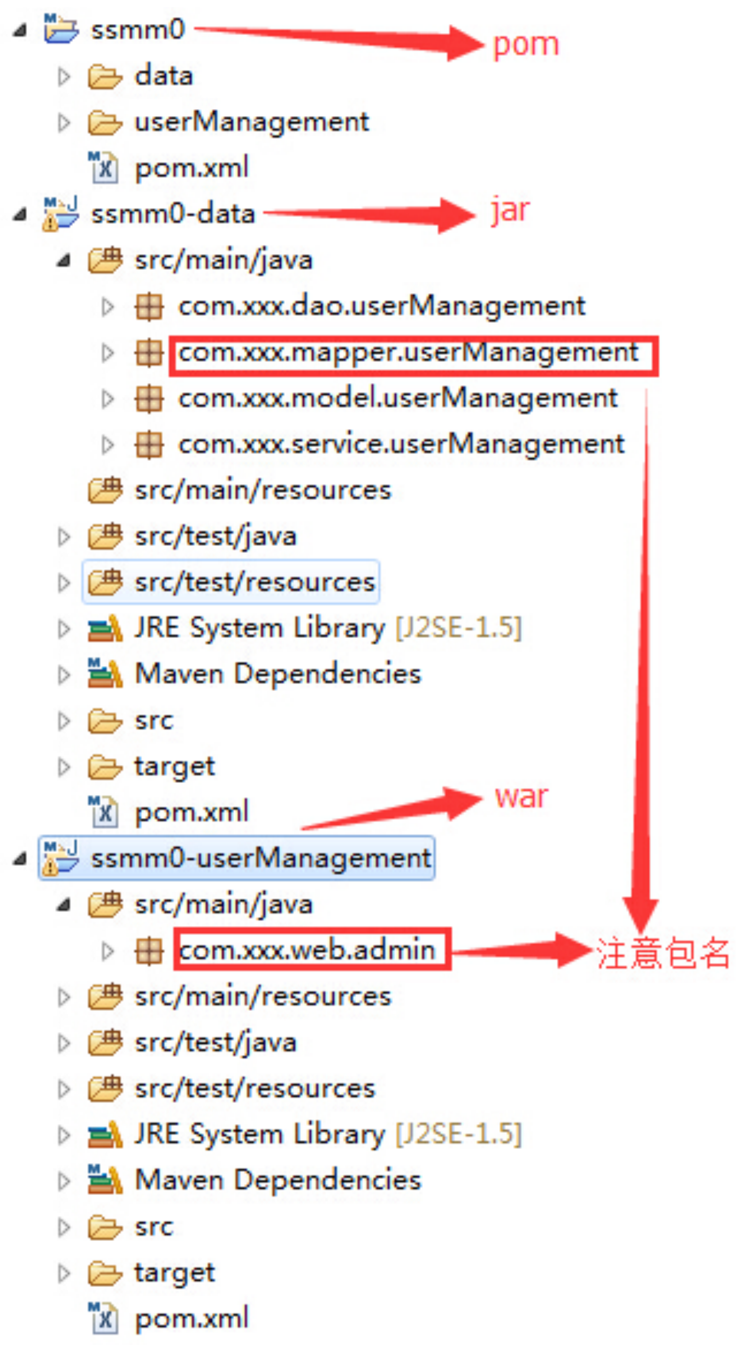
注意以下几点：

- **包的命名最好是com.xxx.mapper.user和不是com.xxx.user.mapper**，前者在spring.xml中配置mybatis时更方便，具体见spring.xml的中的注释
- 在controller那一层的项目是需要部署的，即是war，而下边的数据模块是作为war的一个jar，所以在war层的pom.xml需要将下边的数据模块作为一个jar来引入到项目中
- **service层**到底是放在业务模块处还是放在数据模块处，这个根据需求而定，**一般而言，都放在数据模块处**，方便彼此service的调用，如userService调用hotelService，如果这个时候把两个service分别放在各自的业务模块层中，相互的调用就要通过RPC了，当然，有的时候可能有些与其他模块都不调用的service放在war层可能会好一些。
- 将来编写的缓存模块类、通用模块类、RPC工具类等都会作为jar被war层调用。

2、实现

我将上一章的项目做了修改，将ssmm项目改成了userManagement项目，并将userManagement项目实现了业务模块和数据模块的分离，具体的操作参照第一章和第二章的相关内容，这里直接给出项目结构和各个文件。

2.1、项目结构


```
ssmm0
    data
    userManagement
    pom.xml
ssmm0-data
    src/main/java
        com.xxx.dao.userManagement
        com.xxx.mapper.userManagement
        com.xxx.model.userManagement
        com.xxx.service.userManagement
    src/main/resources
    src/test/java
    src/test/resources
    JRE System Library [J2SE-1.5]
    Maven Dependencies
    src
    target
    pom.xml
ssmm0-userManagement
    src/main/java
        com.xxx.web.admin
    src/main/resources
    src/test/java
    src/test/resources
    JRE System Library [J2SE-1.5]
    Maven Dependencies
    src
    target
    pom.xml
```

2.2、代码实现

2.2.1、ssmm0

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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.xxx</groupId>
    <artifactId>ssmm0</artifactId>
    <version>1.0-SNAPSHOT</version>

    <name>ssmm0</name>
    <packaging>pom</packaging><!-- 父模块 -->

    <!-- 管理子模块 -->
    <modules>
        <module>userManagement</module><!-- 具体业务1-人员管理系统 -->
        <module>data</module><!-- 封装数据操作 -->
    </modules>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    </properties>

    <!-- dependencyManagement不会引入实际的依赖，只是作为一个依赖池，供其和其子类使用 -->
```

```xml
    <dependencyManagement>
        <dependencies>
            <!-- json -->
            <dependency>
                <groupId>com.alibaba</groupId>
                <artifactId>fastjson</artifactId>
                <version>1.1.39</version>
            </dependency>
            <!-- servlet -->
            <dependency>
                <groupId>javax.servlet</groupId>
                <artifactId>javax.servlet-api</artifactId>
                <version>3.0.1</version>
                <scope>provided</scope>
            </dependency>
            <!-- spring -->
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-core</artifactId>
                <version>3.2.6.RELEASE</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-beans</artifactId>
                <version>3.2.6.RELEASE</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>3.2.6.RELEASE</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-web</artifactId>
                <version>3.2.6.RELEASE</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-webmvc</artifactId>
                <version>3.2.6.RELEASE</version>
            </dependency>
            <!-- 这个是使用velocity的必备包 -->
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context-support</artifactId>
                <version>3.2.6.RELEASE</version>
            </dependency>
            <!-- mysql -->
            <dependency>
                <groupId>mysql</groupId>
                <artifactId>mysql-connector-java</artifactId>
                <version>5.1.27</version>
                <scope>runtime</scope>
            </dependency>
            <!-- 数据源 -->
            <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat-jdbc</artifactId>
                <version>7.0.47</version>
            </dependency>
            <!-- mybatis -->
            <dependency>
                <groupId>org.mybatis</groupId>
                <artifactId>mybatis</artifactId>
                <version>3.1.1</version>
            </dependency>
            <dependency>
                <groupId>org.mybatis</groupId>
                <artifactId>mybatis-spring</artifactId>
                <version>1.1.1</version>
            </dependency>
            <!-- velocity -->
            <dependency>
                <groupId>org.apache.velocity</groupId>
                <artifactId>velocity</artifactId>
                <version>1.5</version>
            </dependency>
            <dependency>
                <groupId>velocity-tools</groupId>
                <artifactId>velocity-tools-generic</artifactId>
                <version>1.2</version>
            </dependency>
            <!-- 用于加解密 -->
            <dependency>
                <groupId>commons-codec</groupId>
                <artifactId>commons-codec</artifactId>
                <version>1.7</version>
            </dependency>
            <dependency>
                <groupId>org.bouncycastle</groupId>
                <artifactId>bcprov-jdk15on</artifactId>
                <version>1.47</version>
            </dependency>
            <!-- 集合工具类 -->
            <dependency>
                <groupId>org.apache.commons</groupId>
                <artifactId>commons-collections4</artifactId>
                <version>4.0</version>
            </dependency>
            <!-- http -->
            <dependency>
                <groupId>org.apache.httpcomponents</groupId>
                <artifactId>httpclient</artifactId>
                <version>4.2.6</version>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <!-- 引入实际依赖 -->
    <dependencies>
        <!-- json -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
        </dependency>
        <!-- spring -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
        </dependency>
        <!-- 集合工具类 -->
        <dependency>
```

```
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-collections4</artifactId>
        </dependency>
    </dependencies>

    <build>
        <resources>
            <!--
                这里配置了这一块儿true，才可以让指定文件（这里是src/main/resources/*.xml）读到pom.xml中的配置信息 ，
                值得注意的是，如果src/main/resources下还有其他文件，而你不想让其读pom.xml，
                你还必须得把src/main/resources下的其余文件再配置一遍，配置为false（不可读pom.xml），
                如下边的注释那样，否则，会报这些文件（在这里，就是*.properties）找不到的错误
            -->
            <resource>
                <directory>src/main/resources</directory>
                <filtering>true</filtering>
                <includes>
                    <include>*.xml</include>
                </includes>
            </resource>
            <!-- <resource>
                <directory>src/main/resources</directory>
                <filtering>false</filtering>可以读 ，若改为false就是不可读
                <includes>
                    <include>*.properties</include>
                </includes>
            </resource> -->
        </resources>

    </build>

    <!--
        profiles可以定义多个profile，然后每个profile对应不同的激活条件和配置信息，从而达到不同环境使用不同配置信息的效
果
        注意两点：
        1）<activeByDefault>true</activeByDefault>这种情况表示服务器启动的时候就采用这一套env（在这里，就是prod）
        2）当我们启动服务器后，想采用开发模式，需切换maven的env为dev，如果env的配置本身就是dev，需要将env换成rc或prod，
点击apply，然后再将env切换成dev，点击apply才行
    -->
    <profiles>
        <!-- 开发env -->
        <profile>
            <id>dev</id>
            <activation>
                <activeByDefault>false</activeByDefault>
                <property>
                    <name>env</name>
                    <value>dev</value>
                </property>
            </activation>
            <properties>
                <env>dev</env>

                <jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>
                <!--
                    对于jdbc.url中内容的配置，如果需要配置 &amp;时，有两种方法：
                    1）如下边这样，使用<![CDATA[XXX]]>包起来
                    2）使用jdbc.properties文件来读取此pom.xml，然后spring.xml再读取jdbc.properties文件
                    显然，前者更方便，而且还省了一个jdbc.properties的文件，但是，有的时候，还是会用后者的；
                    在使用后者的时候，注意三点：
                    1）需要修改上边的build中的内容
                    2）需要在spring.xml中配置<context:property-placeholder
location="classpath:jdbc.properties"/>
                    3）将jdbc.properties放在ssmm0-data项目中，之后需要将ssmm0-data项目的env配置为dev
                -->
                <jdbc.url><![CDATA[jdbc:mysql://127.0.0.1:3306/blog?zeroDateTimeBehavior=convertToNull&
amp;useUnicode=true&amp;characterEncoding=utf-8]]></jdbc.url>
                <jdbc.username>root</jdbc.username>
                <jdbc.password>123456</jdbc.password>
            </properties>
        </profile>
        <!-- 预上线env -->
        <profile>
            <id>rc</id>
            <activation>
                <activeByDefault>false</activeByDefault>
                <property>
                    <name>env</name>
                    <value>rc</value>
                </property>
            </activation>
            <properties>
                <env>rc</env>

                <jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>
                <!-- 假设的一个地址 -->
                <jdbc.url><![CDATA[jdbc:mysql://10.10.10.100:3306/blog?zeroDateTimeBehavior=convertToNull&
amp;useUnicode=true&amp;characterEncoding=utf-8]]></jdbc.url>
                <jdbc.username>root2</jdbc.username>
                <jdbc.password>1234562</jdbc.password>
            </properties>
        </profile>
        <!-- 线上env -->
        <profile>
            <id>prod</id>
            <activation>
                <activeByDefault>true</activeByDefault>
                <property>
                    <name>env</name>
                    <value>prod</value>
                </property>
            </activation>
            <properties>
                <env>prod</env>

                <jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>
                <!-- 假设的一个地址 -->
                <jdbc.url><![CDATA[jdbc:mysql://99.99.99.999:3307/blog?zeroDateTimeBehavior=convertToNull&
amp;useUnicode=true&amp;characterEncoding=utf-8]]></jdbc.url>
                <jdbc.username>sadhijhqwui</jdbc.username>
                <jdbc.password>zxczkchwihcznk=</jdbc.password>
            </properties>
        </profile>
    </profiles>
</project>
```

注意：

- **所有的注意点：都在注释中**
- 上述<build>中的resource的配置是为了是spring.xml可以读取pom.xml文件的内容，具体的注意点，查看注释
- profiles的配置是为了配置多套环境（在这里配置了三套env，开发，预上线和线上环境），具体的注意点，查看注释

2.2.2、ssmm0-data

pom.xml

```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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <!-- 指定父模块 -->
    <parent>
        <groupId>com.xxx</groupId>
        <artifactId>ssmm0</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <groupId>com.xxx.ssmm0</groupId>
    <artifactId>ssmm0-data</artifactId>

    <name>ssmm0-data</name>
    <packaging>jar</packaging><!-- 只是作为其他模块使用的工具 -->

    <!-- 引入实际依赖 -->
    <dependencies>
        <!-- mysql -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- 数据源 -->
        <dependency>
            <groupId>org.apache.tomcat</groupId>
            <artifactId>tomcat-jdbc</artifactId>
        </dependency>
        <!-- mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
        </dependency>
    </dependencies>
</project>
```

注意：<package>为jar

com.xxx.model.userManagement.Admin

```java
package com.xxx.model.userManagement;

/**
 * 管理员
 */
public class Admin {
    private int id;
    private String username;
    private String password;

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

com.xxx.mapper.userManagement.AdminMapper

```
package com.xxx.mapper.userManagement;

import org.apache.ibatis.annotations.Insert;
import org.apache.ibatis.annotations.Param;
import org.apache.ibatis.annotations.Result;
import org.apache.ibatis.annotations.Results;
import org.apache.ibatis.annotations.Select;

import com.xxx.model.userManagement.Admin;

/**
 * 管理员Mapper
 */
public interface AdminMapper {
    @Insert("INSERT INTO userinfo(username, password) VALUES(#{username},#{password})")
    public int insertAdmin(Admin admin);

    @Select("SELECT * FROM userinfo WHERE username = #{username} AND password = #{password}")
    @Results(value = {
            @Result(id = true, column = "id", property = "id"),
            @Result(column = "username", property = "username"),
            @Result(column = "password", property = "password") })
    public Admin selectAdmin(@Param("username") String username,
                             @Param("password") String password);
}
```

com.xxx.dao.userManagement.AdminDao

```
package com.xxx.dao.userManagement;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
```

```java
import com.xxx.mapper.userManagement.AdminMapper;
import com.xxx.model.userManagement.Admin;

/**
 * 管理员DAO
 */
@Repository
public class AdminDao {
    @Autowired
    private AdminMapper adminMapper;

    public boolean register(Admin admin){
        return adminMapper.insertAdmin(admin)==1?true:false;
    }

    public Admin login(String username ,String password){
        return adminMapper.selectAdmin(username, password);
    }
}
```

com.xxx.service.userManagement.AdminService

```java
package com.xxx.service.userManagement;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.xxx.dao.userManagement.AdminDao;
import com.xxx.model.userManagement.Admin;

/**
 * 管理员service
 */
@Service
public class AdminService {
    @Autowired
    private AdminDao adminDao;

    public boolean register(Admin admin){
        return adminDao.register(admin);
    }

    public Admin login(String username, String password) {
        return adminDao.login(username, password);
    }
}
```

代码很简单，与之前的基本一样，只是名字换了而已。

值得注意的是包名：com.xxx.mapper.userManagement而非com.xxx.userManagement.mapper。

2.2.3、ssmm0-userManagement

pom.xml

```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/maven-v4_0_0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <!-- 指定父模块 -->
    <parent>
        <groupId>com.xxx</groupId>
        <artifactId>ssmm0</artifactId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <groupId>com.xxx.ssmm0</groupId>
    <artifactId>ssmm0-userManagement</artifactId>
    <!--<version>1.0-SNAPSHOT</version>--><!-- 父模块已经指定了版本号，这里就不用了-->

    <name>ssmm0-userManagement</name>
    <packaging>war</packaging><!-- 需要部署的模块 -->

    <!-- 引入实际依赖 -->
    <dependencies>
        <!-- 将ssmm0-data项目作为一个jar引入项目中 -->
        <dependency>
            <groupId>com.xxx.ssmm0</groupId>
            <artifactId>ssmm0-data</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
        <!-- servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
        </dependency>
        <!-- spring mvc -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
        </dependency>
        <!-- 这个是使用velocity的必备包 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
        </dependency>
        <!-- velocity -->
        <dependency>
            <groupId>org.apache.velocity</groupId>
            <artifactId>velocity</artifactId>
        </dependency>
        <dependency>
            <groupId>velocity-tools</groupId>
            <artifactId>velocity-tools-generic</artifactId>
        </dependency>
    </dependencies>
</project>
```

注意：将ssmm0-data作为普通的jar引入即可。

spring.xml

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

```xml
<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-3.2.xsd
                           http://www.springframework.org/schema/context
                           http://www.springframework.org/schema/context/spring-context-3.2.xsd
                           http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

    <!-- 注解扫描 -->
    <context:component-scan base-package="com.xxx" />

    <!-- 配置fastjson转换器 -->
    <mvc:annotation-driven>
        <mvc:message-converters register-defaults="true">
            <bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter"></bean>
        </mvc:message-converters>
    </mvc:annotation-driven>

    <!-- 引入数据源，这里变量的读取都是从ssmm0的pom.xml中读取的 -->
    <bean id="xxxDataSource" class="org.apache.tomcat.jdbc.pool.DataSource" destroy-method="close">
        <property name="driverClassName" value="${jdbc.driverClassName}" />
        <property name="url" value="${jdbc.url}" />
        <property name="username" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
    </bean>

    <!-- 引入mybatis -->
    <bean id="xxxSqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="xxxDataSource" />
    </bean>
    <bean id="xxxMapperScannerConfigurer" class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!--
            这里就是包名为什么就做com.xxx.mapper.user而非com.xxx.user.mapper,
            这样的话，比如说有两个项目com.xxx.mapper.user和com.xxx.mapper.hotel，value只需写作com.xxx.mapper即可
            否则，value就要写作com.xxx.user.mapper,com.xxx.hotel.mapper
         -->
        <property name="basePackage" value="com.xxx.mapper" />
        <property name="sqlSessionFactoryBeanName" value="xxxSqlSessionFactory" />
    </bean>

    <!-- 配置velocity -->
    <bean id="velocityConfigurer" class="org.springframework.web.servlet.view.velocity.VelocityConfigurer">
        <property name="resourceLoaderPath">
            <value>WEB-INF/templates/</value>
        </property>
        <property name="velocityProperties">
            <props>
                <prop key="input.encoding">UTF-8</prop>
                <prop key="output.encoding">UTF-8</prop>
            </props>
        </property>
    </bean>
    <bean id="viewResolver" class="org.springframework.web.servlet.view.velocity.VelocityViewResolver">
        <property name="suffix" value=".vm" />
        <property name="contentType" value="text/html;charset=utf-8" />
        <property name="dateToolAttribute" value="date"/>
        <property name="numberToolAttribute" value="number"/>
    </bean>
</beans>
```

注意：这里对包名的体现，由于直接使用spring.xml去读ssmm0的 pom.xml了，所以jdbc.properties文件就不要了，在spring.xml中，指定文件位置的<context:property-placeholder>标签就删掉了

web.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee
/web-app_2_5.xsd">

    <servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>encodingFilter</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>
        <init-param>
            <param-name>forceEncoding</param-name>
            <param-value>true</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <welcome-file-list>
        <welcome-file>/index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

com.xxx.web.admin.AdminController

```java
package com.xxx.web.admin;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.servlet.ModelAndView;

import com.xxx.model.userManagement.Admin;
import com.xxx.service.userManagement.AdminService;

/**
 * adminController
 */
@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private AdminService adminService;

    /**
     * 管理员注册
     */
    @ResponseBody
    @RequestMapping("/register")
    public boolean register(@RequestParam("username") String username,
                            @RequestParam("password") String password){
        Admin admin = new Admin();
        admin.setUsername(username);
        admin.setPassword(password);

        boolean isRegisterSuccess = adminService.register(admin);

        return isRegisterSuccess;
    }

    /**
     * 管理员登录
     */
    @RequestMapping("/login")
    public ModelAndView login(@RequestParam("username") String username,
                              @RequestParam("password") String password){
        Admin admin = adminService.login(username, password);

        ModelAndView modelAndView = new ModelAndView();
        if(admin == null){
            modelAndView.addObject("message", "用户不存在或者密码错误！请重新输入");
            modelAndView.setViewName("error");
        }else{
            modelAndView.addObject("admin", admin);
            modelAndView.setViewName("userinfo");
        }

        return modelAndView;
    }
}
```

error.vm

```html
<!DOCTYPE html>
<html lang="zh-cn">
<head>
    <meta charset="UTF-8">
    <title>登录失败</title>
</head>
<body>
    <div>
        $message
    </div>
```

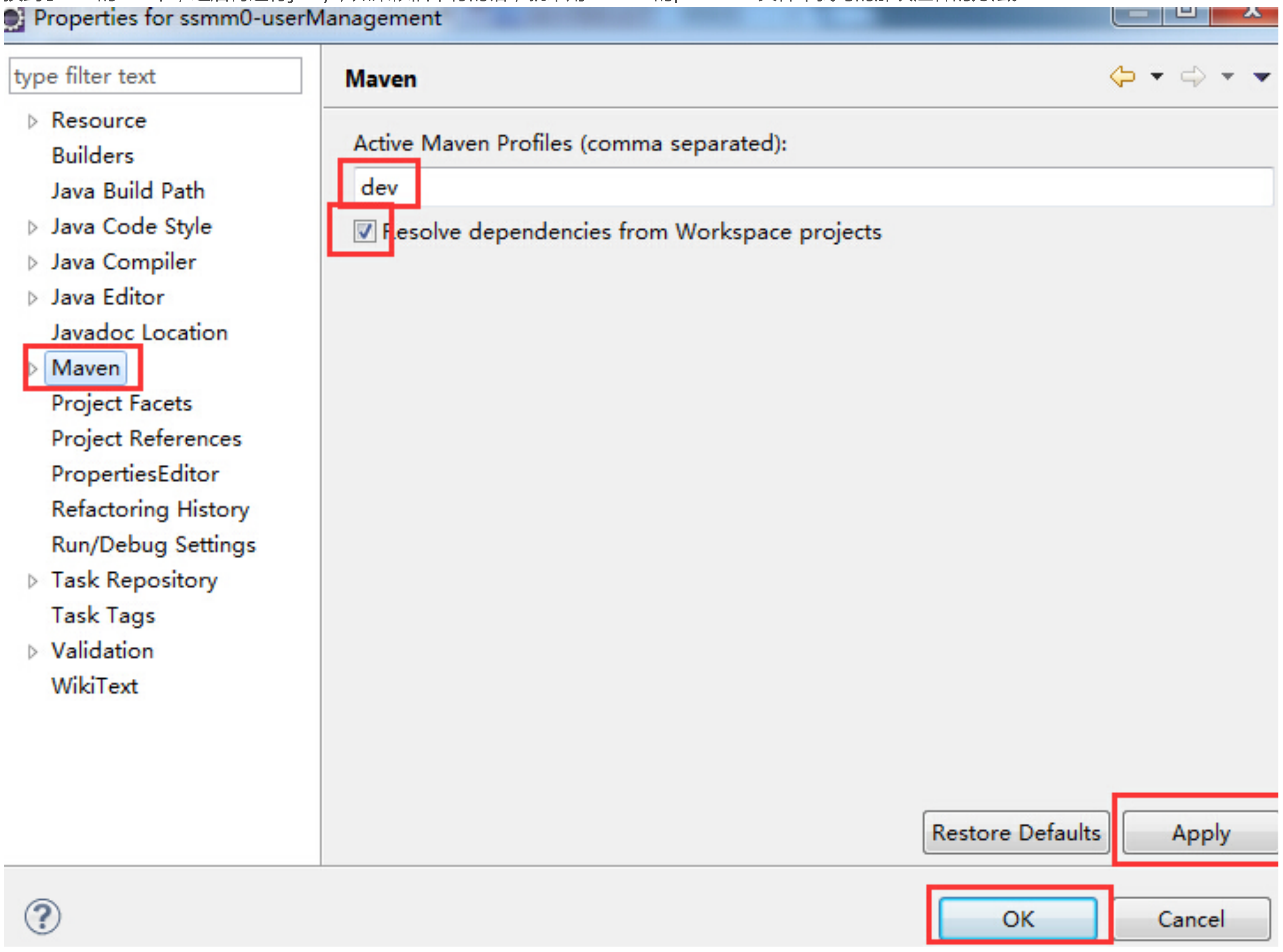
```
</body>
</html>
```

userinfo.vm

```
<!DOCTYPE html>
<html lang="zh-cn">
<head>
    <meta charset="UTF-8">
    <title>登录成功</title>
</head>
<body>
    <div>
        id:$admin.id
        username:$admin.username
        password:$admin.password
    </div>
</body>
</html>
```

具体测试过程：

- 直接运行jetty的话，会发现连不上数据库服务器99.99.99.999，这是因为默认启动服务器之后，我们使用的是prod的一套env，这套env中的数据库服务器是我自己乱写的：99.99.99.999
- 这时候，在ssmm0-userManagement项目上右击-->"Build Path"-->"Configure Build Path"-->"Maven"-->修改env，如下图所示，这样就切换到了dev的env下，之后再运行jetty，如果依旧不行的话，就采用ssmm0的pom.xml文件中我写的那块注释的方法。


Properties for ssmm0-userManagement

Maven

Active Maven Profiles (comma separated): dev

[✓] Resolve dependencies from Workspace projects

这样，一个基本上就是**企业中开发常用的结构**的项目就完成了，**这个项目非常重要，一定自己试着去写一个，一定要仔细看其中的每一条注释。**