第一章 企业项目开发--maven+springmvc+spring+mybatis+velocity整合

代码的github地址：https://github.com/muyinchen/ssm

一、ssmm简介

- ssmm是当下企业最常用的开发框架架构
- maven：管理项目jar包，构建项目
- spring：IOC容器，事务管理
- springmvc：mvc框架
- myBatis：持久层框架
- velocity：前端视图模板（**相较于jsp，速度非常快，而且不需要占据jvm的永久代内存**）

上述这些框架再加上版本控制工具（git）、自动化部署工具（jenkins），就组成了当下中大型企业最常用的项目开发部署架构；以上各种框架以后我也会依次做笔记去写的，所以在下边不会做详细介绍。还有，在以下的整合过程中会有一些细节方面的内容，我会在后续的本系列博客中仔细去说。

二、下面介绍怎样整合ssmm

环境：

- eclipse-jee-indigo-SR2-win32（新出的一些eclipse不支持jdk1.6）
- jdk1.6.45（当下企业最常用的jdk版本）
- spring3.2.6
- mybatis3.1.1
- mybatis-spring1.1.1（mybatis与spring集成的一个工具jar）
- mysql5.1.27
- maven3.0.5
- velocity1.5

2.1、构建maven项目

步骤：

2.1.1、手工创建maven项目（整个过程就是创建一个符合maven格式的目录结构）

注意：这里使用手工创建，而不是在eclipse中使用maven插件去创建，是因为个人感觉eclipse的maven插件不好用。

首先自己创建一个文件夹ssmm（这个文件夹名称就是我们的项目名称），然后在ssmm下创建src文件夹和pom.xml文件。

其中，pom.xml文件中的内容如下：

```
<?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>ssmm</artifactId>
    <version>1.0-SNAPSHOT</version>

    <name>ssm</name>
    <packaging>war</packaging>

</project>
```

以上内容，会在之后的maven模块中讲解。

然后，在src下创建main、test两个文件夹；在main文件夹下创建java、resources、webapp三个文件夹，在test文件夹下建立java、resources两个文件夹；之后在webapp下创建META-INF和WEB-INF两个文件夹，在META-INF文件夹下创建MANIFEST.MF文件，该文件内容如下：

```
Manifest-Version:1.0
```

在WEB-INF文件夹下创建templates（该文件夹将来用于存放模板文件）和web.xml文件，web.xml文件内容如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee
/web-app_2_5.xsd">
5
6 </web-app>
```

当构建好以上目录结构后，在src/main/java下创建一个TestMaven.java文件，文件内容如下：

```
public class TestMaven {
    public static void main(String[] args) {
        System.out.println("hello zhaojigang!!!");
    }
}
```

2.1.2、使用命令窗口编译该项目

在ssmm文件夹下（即pom.xml所在的目录）使用打开命令窗口（shift+鼠标右键），使用"mvn compile"编译该项目（前提：电脑安装了maven）。编译成功出现"BUILD SUCCESS"。

2.1.3、引入eclipse

在eclipse中"import"-->"Existing Maven Projects"即可（执行此步之前，需要安装eclipse的maven插件）。

引入项目后，在ssmm项目上右击-->"Properties"-->Text file encoding改为UTF-8

然后对项目进行测试（在webapp目录下建立index.jsp，然后运行项目，在我这里，是使用jetty来运行的，然后通过访问浏览器查看项目是否建立成功）。

2.2、引入spring与springmvc

2.2.1、引入jar包

在这里为了方便，我直接给出这些框架整合的完整的pom.xml（具体引入的每个包做什么用，看pom.xml的注释），之后其他框架的整合就不再讨论jar包问题了。

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
 4
 5     <modelVersion>4.0.0</modelVersion>
 6
 7     <groupId>com.xxx</groupId>
 8     <artifactId>ssmm</artifactId>
 9     <version>1.0-SNAPSHOT</version>
10
11     <name>ssmm</name>
12     <packaging>war</packaging>
13
14     <properties>
15         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16         <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
```

```
    </properties>

    <!-- 引入实际依赖 -->
    <dependencies>
        <!-- json -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.1.39</version>
        </dependency>
        <!-- servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</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>
            <scope>compile</scope>
        </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>
    </dependencies>
</project>
```

2.2.2、配置spring.xml文件

这里直接列出spring.xml的完整版，包括读取属性文件、配置数据源、配置fastjson转换器、配置mybatis、配置velocity。

```
<?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-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" />

    <!-- 引入属性文件 -->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!-- 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 -->
    <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">
        <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>
```

在读取数据源的时候，用一个jdbc.properties文件来存放所有的信息。

```
1 jdbc.driverClassName = com.mysql.jdbc.Driver
2 jdbc.url = jdbc:mysql://localhost:3306/blog?zeroDateTimeBehavior=convertToNull&amp;useUnicode=true&
amp;characterEncoding=utf-8
3 jdbc.username = root
4 jdbc.password = 123456
```

注意：上述属性文件中有一些细节，之后会讲。

2.2.3、配置web.xml文件

这里直接给出web.xml的完整版，包括配置DispatcherServlet、CharacterEncodingFilter与欢迎页面。

注意：不需要配置spring的ContextLoaderListener了。

```
 1 <?xml version="1.0" encoding="utf-8"?>
 2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
 3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee
/web-app_2_5.xsd">
 5
 6     <servlet>
 7         <servlet-name>dispatcherServlet</servlet-name>
 8         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
 9         <init-param>
10             <param-name>contextConfigLocation</param-name>
11             <param-value>classpath*:spring*.xml</param-value>
12         </init-param>
13         <load-on-startup>1</load-on-startup>
14     </servlet>
15     <servlet-mapping>
16         <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>
```


2.3、引入mybatis

引入jar在pom.xml中搞定了；配置mybatis在spring.xml中搞定了。

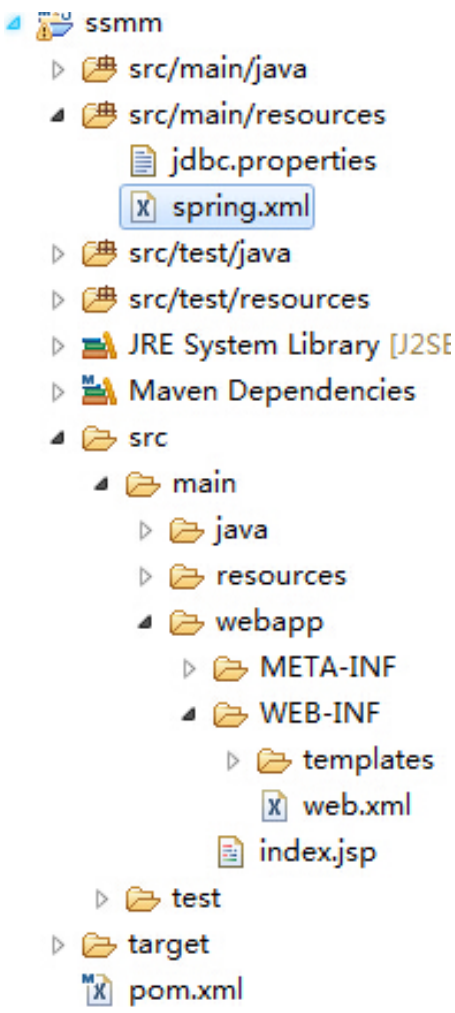
注意：这里对于mybatis的配置并不完全，因为这里我只用到了mybatis的注解，并没有用到xml形式，关于注解与xml的使用场景以及mybatis的一些其他方面的知识，以后再做记录。

2.4、引入velocity

引入jar在pom.xml中搞定了；配置mybatis在spring.xml中搞定了。具体的配置信息在以后会做记录。

2.5、测试

在以上程序搭建完成后，形成下边的结构（自己注意看上述所述的各个文件的位置所在）

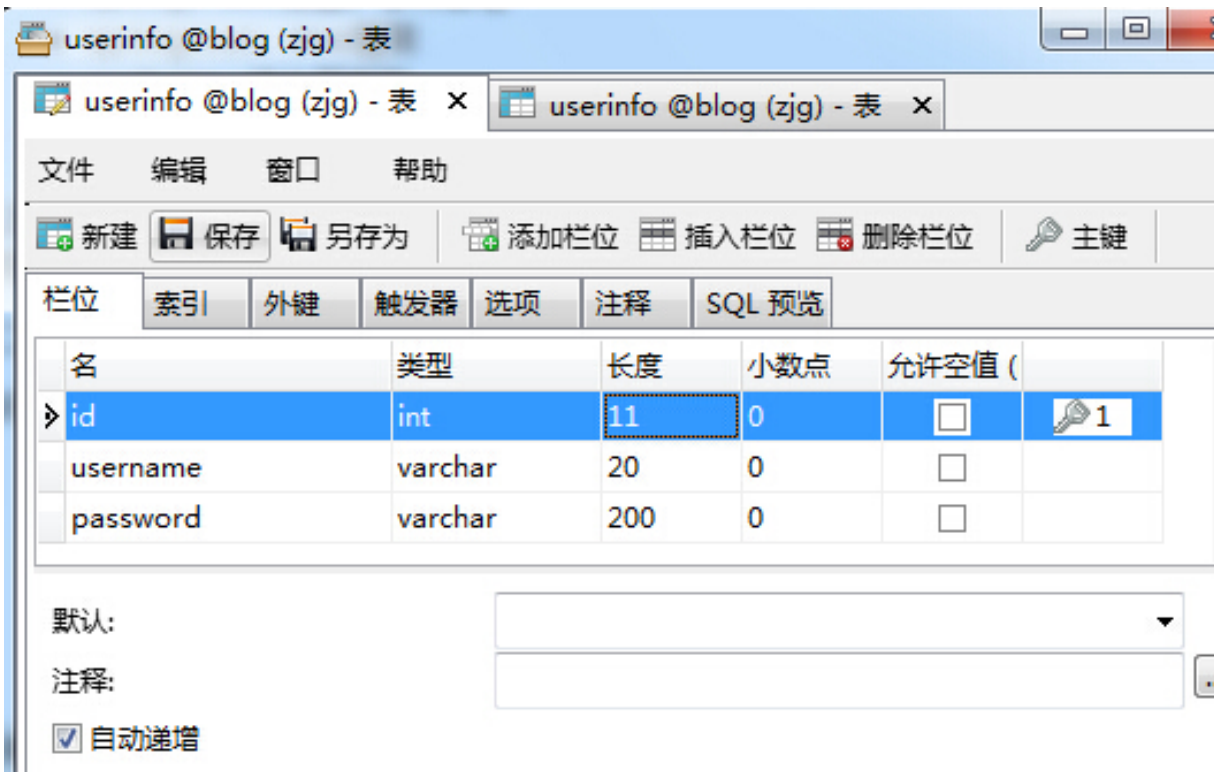


然后进行测试：

测试的内容包含两部分逻辑

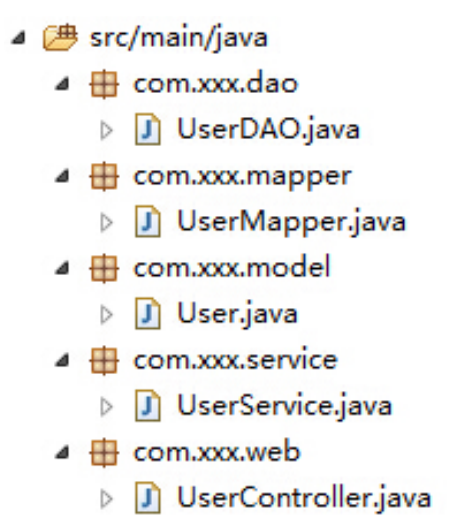
- 注册：注册成功后返回true，注册失败后返回false（注意：在下边的controller中，这一块json的返回会在之后详细讲解）
- 登录：登录成功后跳到userinfo.vm文件，并输出登录的用户信息；登录失败后跳到error.vm文件，并输出相关的错误提示信息。

数据库表：



1）编写后台

后台代码结构如图



UserController：

```java
package com.xxx.web;

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.User;
import com.xxx.service.UserService;

@Controller
@RequestMapping("user")
public class UserController {

    @Autowired
    private UserService userService;

    @ResponseBody
    @RequestMapping("register")
    public boolean register(@RequestParam("username") String username,
                            @RequestParam("password") String password){
        User user = new User();
        user.setUsername(username);
        user.setPassword(password);

        boolean isRegisterSuccess = userService.register(user);

        return isRegisterSuccess;
    }

    @RequestMapping("login")
    public ModelAndView login(@RequestParam("username") String username,
                              @RequestParam("password") String password){
        User user = userService.login(username, password);

        ModelAndView modelAndView = new ModelAndView();
        if(user == 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("user", user);
            modelAndView.setViewName("userinfo");
        }

        return modelAndView;
    }
}
```

UserService :

```java
package com.xxx.service;

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

import com.xxx.dao.UserDAO;
```

```java
import com.xxx.model.User;

@Service
public class UserService {

    @Autowired
    private UserDAO userDao;

    public boolean register(User user){
        return userDao.register(user);
    }

    public User login(String username, String password) {
        return userDao.login(username, password);
    }
}
```

UserDAO：

```java
package com.xxx.dao;

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

import com.xxx.mapper.UserMapper;
import com.xxx.model.User;

@Repository
public class UserDAO {

    @Autowired
    private UserMapper userMapper;

    public boolean register(User user){
        return userMapper.insertUser(user)==1?true:false;
    }

    public User login(String username ,String password){
        return userMapper.selectByUsernameAndPwd(username, password);
    }
}
```

UserMapper：

```java
package com.xxx.mapper;

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.User;

public interface UserMapper {

    @Insert("INSERT INTO userinfo(username, password) VALUES(#{username},#{password})")
```

```
14     public int insertUser(User user);
15
16     @Select("SELECT * FROM userinfo WHERE username = #{username} AND password = #{password}")
17     @Results(value = { @Result(id = true, column = "id", property = "id"),
18                        @Result(column = "username", property = "username"),
19                        @Result(column = "password", property = "password")})
20     public User selectByUsernameAndPwd(@Param("username")String username ,@Param("password")String
password);
21 }
```

具体逻辑自己看吧，很简单的。

2）编写前台

error.vm

```
<!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:$user.id
        username:$user.username
        password:$user.password
    </div>
</body>
</html>
```

velocity的语法自己查看相关的文档吧。

最后，通过浏览器输入带参数URL的形式对程序进行测试。

整个整合过程就是这样的。