SSI框架搭建

SpringMVC3.1.2+Spring3.1.2+Mybatis3.2.6

编号：SSI-SMVC3-S3-I3

版本：V1.0

级别：公开

编写时间：2016-02-17

目 录

[1 导言 1](#_Toc443662053)

[1.1 目的 1](#_Toc443662054)

[1.2 范围 1](#_Toc443662055)

[1.3 说明 1](#_Toc443662056)

[2 搭建SpringMVC 2](#_Toc443662057)

[2.1 搭建所需jar包 2](#_Toc443662058)

[2.2 其他依赖包 3](#_Toc443662059)

[2.3 搭建步骤 4](#_Toc443662060)

[2.3.1 创建项目 4](#_Toc443662061)

[2.3.2 导入jar包 6](#_Toc443662062)

[2.3.3 配置web.xml 7](#_Toc443662063)

[2.3.4 配置spring-servlet.xml 9](#_Toc443662064)

[2.3.5 配置applicationContext.xml 10](#_Toc443662065)

[2.3.6 配置log4j.properties 10](#_Toc443662066)

[3 整合mybatis 11](#_Toc443662067)

[3.1 整合所需jar包 11](#_Toc443662068)

[3.2 其他依赖包 11](#_Toc443662069)

[3.3 整合步骤 11](#_Toc443662070)

[3.3.1 导入jar包 11](#_Toc443662071)

[3.3.2 配置config.properties 12](#_Toc443662072)

[3.3.3 配置spring-dataSource.xml 12](#_Toc443662073)

[3.3.4 配置applicationContext.xml 15](#_Toc443662074)

[3.3.5 配置mybatis-config.xml 16](#_Toc443662075)

[3.3.6 创建实体model 17](#_Toc443662076)

[3.3.7 创建实例化dao 19](#_Toc443662077)

[3.3.8 创建业务服务service 21](#_Toc443662078)

[3.3.9 创建控制层controller 23](#_Toc443662079)

[3.3.10 页面代码 28](#_Toc443662080)

[3.3.11 启动项目 37](#_Toc443662081)

1. 导言
   1. 目的

本文档是根据个人的工作经验搭建的轻量级SSI框架，也是实际应用中比较全面的基础框架，用于指导SSI框架初学者学习搭建SSI框架，希望能给各位使用者提供帮助，同时也希望朋友们尽量去帮助其他人。

* 1. 范围

本次框架搭建的版本是SpringMVC3.1.2+Spring3.1.2+Mybatis3.2.6，数据库采用的是mysql，在eclipse开发工具下搭建直接搭建的web项目，页面采用的是h5，ajax实现数据访问，如果页面为jsp等，则修改controller的返回类型即可。

* 1. 说明

本文档是个人总结的经验，仅供参考学习，同时欢迎朋友们一起沟通学习，文档中会对使用的jar做详细解说，对配置的过程使用图文说明，因为SSI博大精深，很多知识在搭建过程中没有使用到，只是一个基础框架的搭建，能测试通过就可以，所以对于高要求的开发，需要自己添加jar、增加方法和配置，同时文档中出现的纰漏希望朋友们等指正。

开发环境eclipse-jee-indigo-SR1-win32-dic，Tomcat7.0，JDK1.7，mysql。

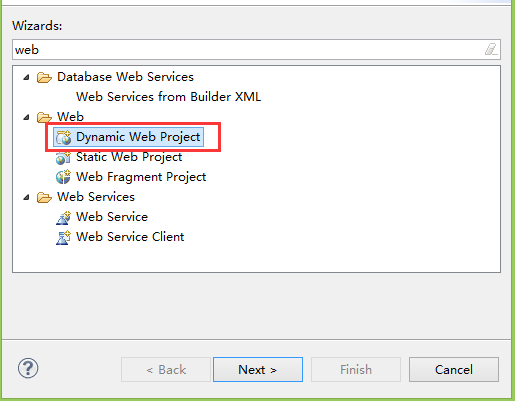
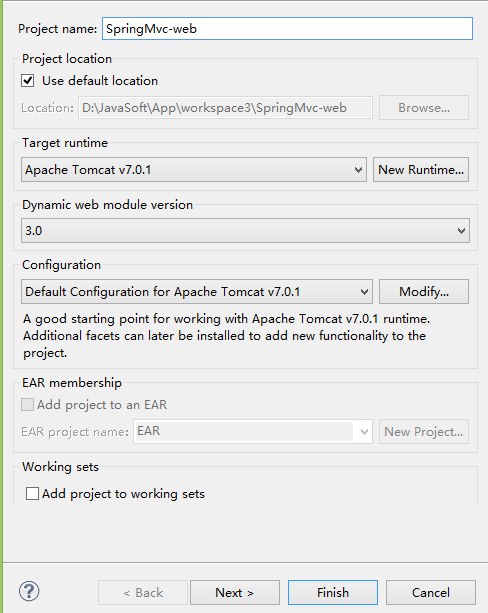
1. 搭建SpringMVC
   1. 搭建所需jar包

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 描述 |
| 1 | spring-aop-3.1.2.RELEASE.jar | spring面向切面编程，提供AOP（面向切面编程） |
| 2 | spring-asm-3.1.2.RELEASE.jar | spring独立的asm程序  [spring2.5.6的时候需要asmJar 包,3.0开始提供他自己独立的asm.jar .ASM是小巧轻便的 Java 字节码操控框架 ，它能方便地生成和改造 Java 代码。 Hibernate 和 Spring 在底层都用到了 ASM(CgLib 底层用的 asm)] |
| 3 | spring-beans-3.1.2.RELEASE.jar | springIoC（依赖注入）的基础实现，所有应用都要用到的，它包含访问配置文件、创建和管理bean以及进行Inversion of Control / Dependency Injection（IoC/DI）操作相关的所有类 |
| 4 | spring-context-3.1.2.RELEASE.jar | spring 提供在基础 IoC 功能上的扩展服务，此外还提供许多企业级服务的支持，如 邮件服务、任务调度、JNDI定位、EJB 集成、远程访问、 缓存以及各种视图层框架的封装等 |
| 5 | spring-core-3.1.2.RELEASE.jar | 包含Spring框架基本的核心工具类，Spring其它组件要都要使用到这个包里的类，是其它组件的基本核心 |
| 6 | spring-expression-3.1.2.RELEASE.jar | spring 表达式语言 |
| 7 | spring-jdbc-3.1.2.RELEASE.jar | 包含对Spring对JDBC数据访问进行封装的所有类 |
| 8 | spring-tx-3.1.2.RELEASE.jar | 封装了spring对于事物的控制 |
| 9 | spring-web-3.1.2.RELEASE.jar | 包含Web应用开发时，用到Spring框架时所需的核心类，包括自动载入WebApplicationContext 特性的类、Struts与JSF集成类、文件上传的支持类、Filter类和大量工具辅助类 |
| 10 | spring-webmvc-3.1.2.RELEASE.jar | Spring Web MVC是一种基于Java的实现了Web MVC设计模式的请求驱动类型的轻量级Web框架，即使用了MVC架构模式的思想，将web层进行职责解耦，基于请求驱动指的就是使用请求-响应模型 |

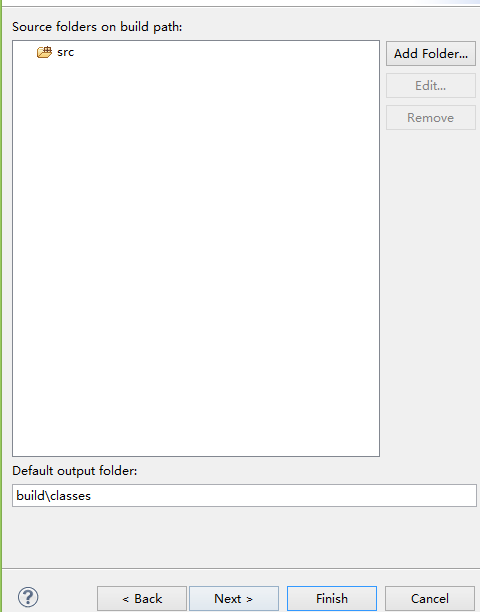
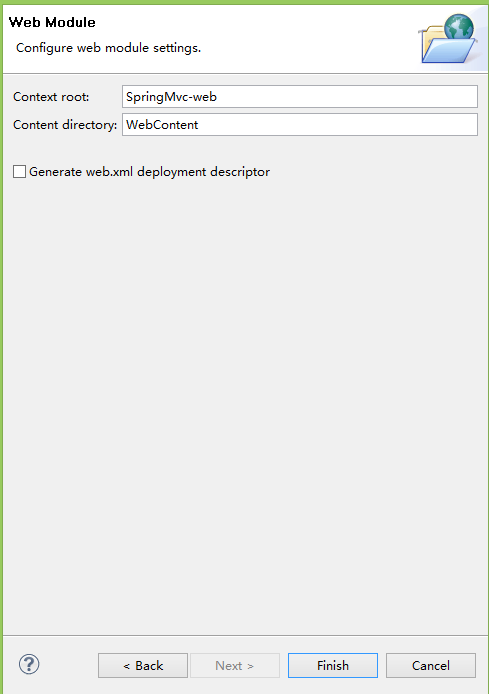
* 1. 其他依赖包

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 描述 |
| 1 | aopalliance-1.0.jar | 这个包是AOP联盟的API包，里面包含了针对面向切面的接口。  通常Spring等其它具备动态织入功能的框架依赖此包 |
| 2 | aspectjweaver-1.8.5.jar | Spring AOP所要用到的包 |
| 3 | commons-lang3-3.3.1.jar | 跟java.lang这个包的作用类似，Commons Lang这一组API也是提供一些基础的、通用的操作和处理，如自动生成toString()的结果、自动实现hashCode()和equals()方法、数组操作、枚举、日期和时间的处理等等 |
| 4 | commons-logging-1.1.1.jar | 在 sun 开发 logger 前，apache 项目已经开发了功能强大的 log4j 日志工具，并向 sun 推荐将其纳入到 jdk 的一部分，可是 sun 拒绝了 apache 的提议，sun 后来自己开发了一套记录日志的工具。可是现在的开源项目都使用的是 log4j，log4j 已经成了事实上的标准，但由于又有一部分开发者在使用 sun logger，因此 apache 才推出 commons-logging，使得我们不必关注我们正在使用何种日志工具 |
| 5 | log4j-1.2.17.jar | 日志组件 |

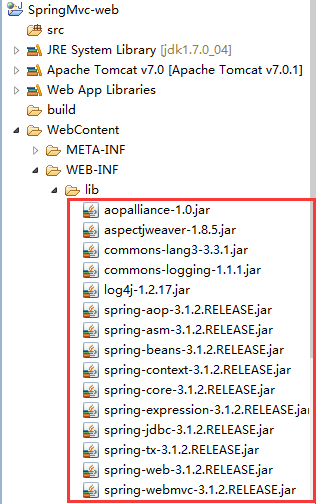
* 1. 搭建步骤
     1. 创建项目

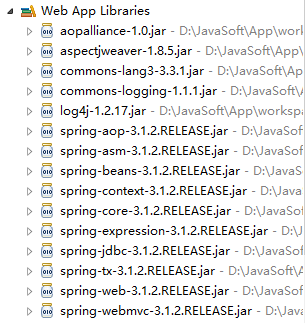
创建一个web项目，如下所示：

需要配置tomcat，用于web启动容器。

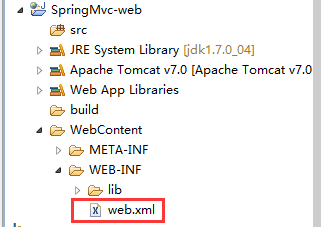


* + 1. 导入jar包

项目中点击lib，复制需要的jar到lib中即完成导入，如下所示：

如果没有导入完成，右键点击项目名（SSH），选择properties-Java Build Path，点击Add Extenrnal JARs，选择项目lib中的jar包进行导入，如下所示：

* + 1. 配置web.xml

在WEB-INF下配置web.xml，若没有该文件，则自己创建一个xml文件，主要配置spring容器，内容如下所示：

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

id="WebApp\_ID" version="3.0">

<!-- 定义LOG4J监听器 -->

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>

<!-- log4jConfigLocation：log4j配置文件存放路径 -->

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>classpath:config/log4j.properties</param-value>

</context-param>

<!-- springMVC Servlet配置 -->

<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:config/spring-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springMvc</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

<!-- spring监听器 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 防止spring内存溢出监听器 -->

<listener>

<listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>

</listener>

<!-- 装载spring配置文件-->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:config/applicationContext.xml</param-value>

</context-param>

<!-- 字符集过滤器 -->

<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>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 会话有效期(单位：分钟) -->

<session-config>

<session-timeout>15</session-timeout>

</session-config>

<!-- 系统首页 -->

<welcome-file-list>

<welcome-file>index.html</welcome-file>

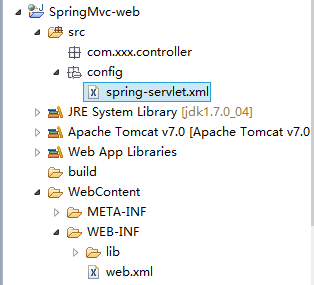
<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

备注：web.xml中，引入了spring-servlet.xml和applicationContext.xml文件，需要把log4j配置在前面，否则会报警告，因为后续加载的程序有打印日志的需求，不过不影响系统运行。

* + 1. 配置spring-servlet.xml

该文件主要进行springmvc的相关配置，如注解、映射、解析等。

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p" xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.2.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.2.xsd ">

<!-- 启用spring mvc 注解 -->

<mvc:annotation-driven />

<!-- 设置使用注解的类所在的jar包 -->

<context:component-scan base-package="com.xxx.controller" />

<!-- 完成请求和注解POJO的映射 -->

<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />

<!-- 对转向页面的路径解析。prefix：前缀， suffix：后缀 -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver" p:prefix="/jsp/" p:suffix=".jsp" />

</beans>

* + 1. 配置applicationContext.xml

配置mybatis的时候再一起介绍。

* + 1. 配置log4j.properties

log4j.rootLogger=info,Console,File

log4j.appender.Console=org.apache.log4j.ConsoleAppender

log4j.appender.Console.encoding=utf-8

log4j.appender.Console.layout=org.apache.log4j.PatternLayout

log4j.appender.Console.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss} %p (%F:%L) - %m%n

#log4j.appender.File=org.apache.log4j.RollingFileAppender

log4j.appender.File=org.apache.log4j.DailyRollingFileAppender

log4j.appender.File.DatePattern = '.'yyyy-MM-dd

log4j.appender.File.File=../logs/SpringMvc.log

#log4j.appender.File.MaxFileSize=10MB

log4j.appender.File.Threshold=ALL

log4j.appender.File.layout=org.apache.log4j.PatternLayout

log4j.appender.File.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss} %p (%F:%L) - %m%n

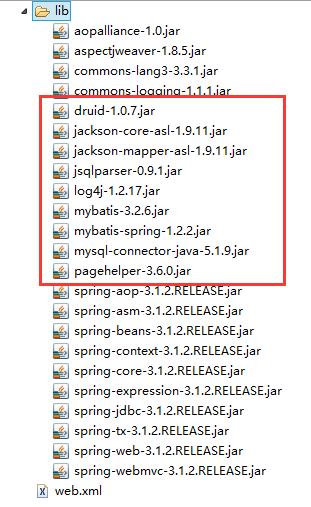
1. 整合mybatis
   1. 整合所需jar包

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 描述 |
| 1 | mybatis-3.2.6.jar | mybatis核心包 |
| 2 | mybatis-spring-1.2.2.jar | mybatis和spring整合包 |
| 3 | mysql-connector-java-5.1.9.jar | mysql驱动包 |

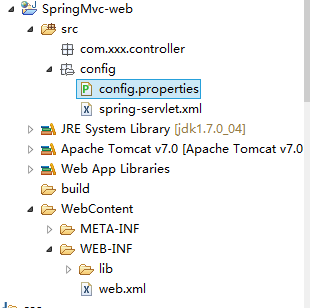
* 1. 其他依赖包

|  |  |  |
| --- | --- | --- |
| 序号 | 名称 | 描述 |
| 1 | druid-1.0.7.jar | Druid是目前最好的数据库连接池，在功能、性能、扩展性方面，都超过其他数据库连接池 |
| 2 | jackson-core-asl-1.9.11.jar | 设置@ResponseBody，把对象转换成json格式传输，Json序列化和反序列化的包 |
| 3 | jackson-mapper-asl-1.9.11.jar | 同上 |
| 4 | jsqlparser-0.9.1.jar | sql解析工具 |
| 5 | pagehelper-3.6.0.jar | mybatis分页插件 |

* 1. 整合步骤
     1. 导入jar包

项目中点击lib，复制需要的jar到lib中即完成导入，如下所示：

* + 1. 配置config.properties

该文件为数据库配置文件，内容如下：

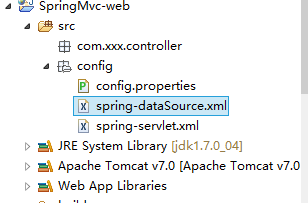
#数据库连接信息配置

jdbc\_username=root

jdbc\_password=root

jdbc\_url=jdbc:mysql://127.0.0.1:3306/springmvc\_test?useUnicode=true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull

* + 1. 配置spring-dataSource.xml

该文件主要进行数据库相关的配置，如下所示：

<?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:tx="http://www.springframework.org/schema/tx" xmlns:aop="http://www.springframework.org/schema/aop"

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/tx

http://www.springframework.org/schema/tx/spring-tx-3.2.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop-3.2.xsd">

<!-- 配置数据源，使用阿里巴巴的druid数据源(也可以使用c3p0或者dbcp，druid的性能更好点 ) -->

<bean id="dataSource" name="dataSource" class="com.alibaba.druid.pool.DruidDataSource"

init-method="init" destroy-method="close">

<!-- 配置数据库连接参数,druid会根据url自动识别dialect和driverClass无须配置 -->

<property name="url" value="${jdbc\_url}" />

<property name="username" value="${jdbc\_username}" />

<property name="password" value="${jdbc\_password}" />

<!-- 初始化来连接大小 -->

<property name="initialSize" value="10" />

<!-- 连接池中使用的最大连接数量，在druid中相当于maxPoolSize -->

<property name="maxActive" value="100" />

<!-- 连接池中最大的空闲连接数 -->

<property name="maxIdle" value="10" />

<!-- 连接池中最小的空闲连接数，在druid中相当于minPoolSize -->

<property name="minIdle" value="5" />

<!-- 获取连接的最大等待时间，单位毫秒 -->

<property name="maxWait" value="60000" />

<!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->

<property name="timeBetweenEvictionRunsMillis" value="60000" />

<!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->

<property name="minEvictableIdleTimeMillis" value="300000" />

<property name="validationQuery" value="SELECT 'x'" />

<property name="testWhileIdle" value="true" />

<property name="testOnBorrow" value="false" />

<property name="testOnReturn" value="false" />

<!-- 打开removeAbandoned功能 -->

<property name="removeAbandoned" value="true" />

<!-- 1800秒，也就是30分钟 -->

<property name="removeAbandonedTimeout" value="1800" />

<!-- 关闭abanded连接时输出错误日志 -->

<property name="logAbandoned" value="true" />

<!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->

<property name="poolPreparedStatements" value="false" />

<property name="maxPoolPreparedStatementPerConnectionSize" value="20" />

<!-- 打开druid的监控和统计功能 并通过日志文件记录sql -->

<property name="filters" value="mergeStat,log4j" />

</bean>

<!-- mybatis相关配置 -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource" />

<!-- 加载mybatis的配置文件 -->

<property name="configLocation" value="classpath:config/mybatis-config.xml" />

<!-- 自动扫描model目录中的映射xml文件，省去了在config中手工配置 -->

<property name="mapperLocations"

value="classpath:com/xxx/dao/mapping/\*\*/\*.xml" />

<!-- 为指定包下的类指定别名，省去了编写全限定名 ，默认使用类名 -->

<property name="typeAliasesPackage" value="com.xxx" />

</bean>

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<property name="basePackage" value="com.xxx.dao" />

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />

</bean>

<!-- 配置事物 -->

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource" />

</bean>

<!-- 拦截器方式配置事物 -->

<tx:advice id="transactionAdvice" transaction-manager="transactionManager">

<tx:attributes>

<tx:method name="save\*" propagation="REQUIRED" />

<tx:method name="update\*" propagation="REQUIRED" />

<tx:method name="delete\*" propagation="REQUIRED" />

<tx:method name="load\*" propagation="SUPPORTS" />

<tx:method name="find\*" propagation="SUPPORTS" />

<tx:method name="list\*" propagation="SUPPORTS" />

<tx:method name="check\*" propagation="SUPPORTS" />

<tx:method name="\*" propagation="SUPPORTS" />

</tx:attributes>

</tx:advice>

<aop:config>

<aop:pointcut id="transactionPointcut"

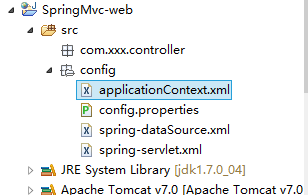
expression="execution(\* com.xxx.service..\*Impl.\*(..))" />

<aop:advisor pointcut-ref="transactionPointcut"

advice-ref="transactionAdvice" />

</aop:config>

</beans>

* + 1. 配置applicationContext.xml

该文件主要进行自动注入，配置文件引入等配置，如下所示：

<?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-3.2.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.2.xsd ">

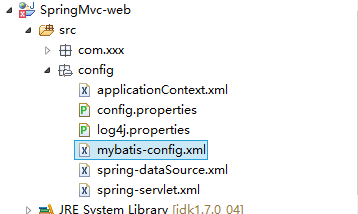
<!-- 自动扫描包(自动注入) -->

<context:component-scan base-package="com.xxx.service" />

<!-- 引入配置文件 -->

<context:property-placeholder location="classpath:config/config.properties" />

<!-- 引入数据源文件 -->

 <import resource="spring-dataSource.xml"/>

</beans>

* + 1. 配置mybatis-config.xml

该文件主要进行mybatis配置，如下所示：

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

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<settings>

<!-- 开启缓存 -->

<setting name="cacheEnabled" value="true" />

<!-- 关闭延迟加载，所有的对象都即时加载 -->

<setting name="lazyLoadingEnabled" value="true" />

<!-- 如果存在延时加载，则对象属性按需加载 -->

<setting name="aggressiveLazyLoading" value="true" />

<setting name="logImpl" value="LOG4J" />

</settings>

<plugins>

<!-- com.github.pagehelper为PageHelper类所在包名 -->

<plugin interceptor="com.github.pagehelper.PageHelper">

<property name="dialect" value="mysql" />

<!-- 该参数默认为false -->

<!-- 设置为true时，会将RowBounds第一个参数offset当成pageNum页码使用 -->

<!-- 和startPage中的pageNum效果一样 -->

<property name="offsetAsPageNum" value="true" />

<!-- 该参数默认为false -->

<!-- 设置为true时，使用RowBounds分页会进行count查询 -->

<property name="rowBoundsWithCount" value="true" />

<!-- 设置为true时，如果pageSize=0或者RowBounds.limit = 0就会查询出全部的结果 -->

<!-- （相当于没有执行分页查询，但是返回结果仍然是Page类型） -->

<property name="pageSizeZero" value="true" />

<!-- 分页参数合理化，默认false禁用 -->

<!-- 启用合理化时，如果pageNum<1会查询第一页，如果pageNum>pages会查询最后一页 -->

<!-- 禁用合理化时，如果pageNum<1或pageNum>pages会返回空数据 -->

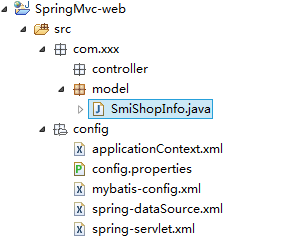
<property name="reasonable" value="true" />

</plugin>

</plugins>

</configuration>

* + 1. 创建实体model

创建一个用户实体，通过用户名和密码登录系统。

package com.xxx.model;

import java.io.Serializable;

/\*\*

\* 影院店铺信息表

\*

\* @author author

\* @date 2016/02/16

\*/

public class SmiShopInfo implements Serializable{

private static final long serialVersionUID = -205189814706545519L;

private Integer id;

private String shopNo;

private String shopName;

private String shopType;

private String shopPassword;

private String note;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getShopNo() {

return shopNo;

}

public void setShopNo(String shopNo) {

this.shopNo = shopNo;

}

public String getShopName() {

return shopName;

}

public void setShopName(String shopName) {

this.shopName = shopName;

}

public String getShopType() {

return shopType;

}

public void setShopType(String shopType) {

this.shopType = shopType;

}

public String getShopPassword() {

return shopPassword;

}

public void setShopPassword(String shopPassword) {

this.shopPassword = shopPassword;

}

public String getNote() {

return note;

}

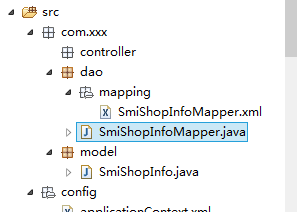
public void setNote(String note) {

this.note = note;

}

}

* + 1. 创建实例化dao

创建实例化dao接口，及映射关系以及实现，如下所示：

SmiShopInfoMapper.java

package com.xxx.dao;

import org.apache.ibatis.annotations.Param;

import com.xxx.model.SmiShopInfo;

/\*\*

\* 影院店铺信息映射关系

\* 数据实例化接口

\*

\* @author author

\* @date 2016/02/16

\*/

public interface SmiShopInfoMapper {

/\*\*

\* 根据门店编码和密码获取门店信息

\*

\* @param password

\* 门店密码

\* @param shopNo

\* 门店编码

\*/

public SmiShopInfo getByNoAndPwd(@Param("password") String password, @Param("shopNo") String shopNo);

}

SmiShopInfoMapper.xml

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

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace="com.smi170.dao.SmiShopInfoMapper" >

<resultMap id="BaseResultMap" type="com.smi170.model.SmiShopInfo" >

<id column="id" property="id" jdbcType="INTEGER" />

<result column="shop\_no" property="shopNo" jdbcType="VARCHAR" />

<result column="shop\_name" property="shopName" jdbcType="VARCHAR" />

<result column="shop\_type" property="shopType" jdbcType="VARCHAR" />

<result column="shop\_password" property="shopPassword" jdbcType="VARCHAR" />

<result column="note" property="note" jdbcType="VARCHAR" />

</resultMap>

<sql id="Base\_Column\_List" >

id, shop\_no, shop\_name, shop\_type, shop\_password, note

</sql>

<!-- 根据门店编码和密码获取门店信息 -->

<select id="getByNoAndPwd" resultMap="BaseResultMap" >

select

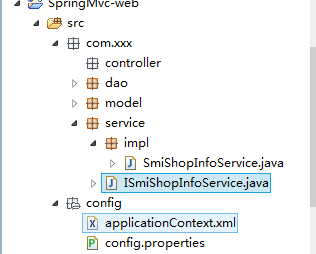
<include refid="Base\_Column\_List" />

from smi\_shop\_info

where shop\_no = #{shopNo,jdbcType=VARCHAR} and shop\_password = #{password,jdbcType=VARCHAR}

</select>

</mapper>

* + 1. 创建业务服务service

创建业务服务接口，及实现，如下所示：

ISmiShopInfoService.java

package com.xxx.service;

import com.xxx.model.SmiShopInfo;

/\*\*

\* 影院店铺信息接口服务

\*

\* @author author

\* @date 2016/02/16

\*/

public interface ISmiShopInfoService {

/\*\*

\* 根据店铺的编码和密码查询店铺信息

\* @param shopNo 店铺编码

\* @param password 密码

\* @return 店铺信息

\* @throws Exception

\*/

public SmiShopInfo getShopByNoAndPwd(String shopNo, String password) throws Exception;

}

SmiShopInfoService.java

package com.xxx.service.impl;

import org.apache.log4j.Logger;

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

import org.springframework.stereotype.Service;

import com.xxx.dao.SmiShopInfoMapper;

import com.xxx.model.SmiShopInfo;

import com.xxx.service.ISmiShopInfoService;

/\*\*

\* 影院店铺信息接口服务实现

\*

\* @author author

\* @date 2016/02/16

\*/

@Service("smiShopInfoService")

public class SmiShopInfoService implements ISmiShopInfoService {

private Logger logger = Logger.getLogger(this.getClass());

@Autowired

public SmiShopInfoMapper shopInfoMapper;

@Override

public SmiShopInfo getShopByNoAndPwd(String shopNo, String password) throws Exception {

if (null==shopNo||"".equals(shopNo)) {

logger.warn("门店编码为null或空，请检查！");

throw new Exception("门店编码为空");

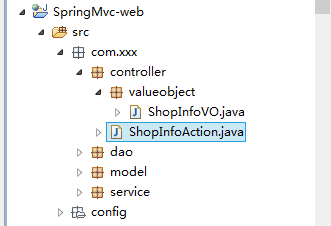
}

return shopInfoMapper.getByNoAndPwd(password, shopNo);

}

}

* + 1. 创建控制层controller

创建控制层，以及vo，提供给页面异步调用，前端采用ajax调用，如下所示：

ShopInfoAction.java

package com.xxx.controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.log4j.Logger;

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.ResponseBody;

import com.xxx.controller.valueobject.ShopInfoVO;

import com.xxx.model.SmiShopInfo;

import com.xxx.service.ISmiShopInfoService;

/\*\*

\* 影院店铺信息action

\*

\* @author author

\* @date 2016/02/16

\*/

@Controller

public class ShopInfoAction {

private Logger logger = Logger.getLogger(this.getClass());

@Autowired

private ISmiShopInfoService smiShopInfoService;

/\*\*

\* 用户登录

\*

\* @ResponseBody 设置该参数，指定response的type为比如json或xml，本文采用json序列化方式传输数据

\*

\* @param request

\* @param response

\* @param shopNo

\* @param password

\* @return

\* @throws Exception

\*/

@RequestMapping(value = "/login.do")

public @ResponseBody ShopInfoVO login(HttpServletRequest request,HttpServletResponse response,String shopNo,String password) throws Exception {

SmiShopInfo shopInfo = smiShopInfoService.getShopByNoAndPwd(shopNo, password);

if(null==shopInfo||"".equals(shopInfo)){

logger.warn(shopNo+"：登录失败，用户名或密码错误");

return null;

}else{

ShopInfoVO vo = new ShopInfoVO();

vo.setId(shopInfo.getId());

vo.setShopNo(shopInfo.getShopNo());

vo.setShopName(shopInfo.getShopName());

vo.setShopPassword("");//密码不传输到页面

vo.setShopType(shopInfo.getShopType());

vo.setNote(shopInfo.getNote());

logger.info(shopInfo.getShopName()+"["+shopNo+"]：登录成功");

return vo;

}

}

}

ShopInfoVO.java 序列化成json文件，提供给页面使用

package com.xxx.controller.valueobject;

import org.codehaus.jackson.map.annotate.JsonSerialize;

import org.springframework.stereotype.Component;

/\*\*

\* 影院店铺信息vo

\*

\* @author author

\* @date 2016/02/16

\*/

@Component

@JsonSerialize

public class ShopInfoVO {

private Integer id;

private String shopNo;

private String shopName;

private String shopType;

private String shopPassword;

private String note;

/\*\*

\* @return the id

\*/

public Integer getId() {

return id;

}

/\*\*

\* @param id the id to set

\*/

public void setId(Integer id) {

this.id = id;

}

/\*\*

\* @return the shopNo

\*/

public String getShopNo() {

return shopNo;

}

/\*\*

\* @param shopNo the shopNo to set

\*/

public void setShopNo(String shopNo) {

this.shopNo = shopNo;

}

/\*\*

\* @return the shopName

\*/

public String getShopName() {

return shopName;

}

/\*\*

\* @param shopName the shopName to set

\*/

public void setShopName(String shopName) {

this.shopName = shopName;

}

/\*\*

\* @return the shopType

\*/

public String getShopType() {

return shopType;

}

/\*\*

\* @param shopType the shopType to set

\*/

public void setShopType(String shopType) {

this.shopType = shopType;

}

/\*\*

\* @return the shopPassword

\*/

public String getShopPassword() {

return shopPassword;

}

/\*\*

\* @param shopPassword the shopPassword to set

\*/

public void setShopPassword(String shopPassword) {

this.shopPassword = shopPassword;

}

/\*\*

\* @return the note

\*/

public String getNote() {

return note;

}

/\*\*

\* @param note the note to set

\*/

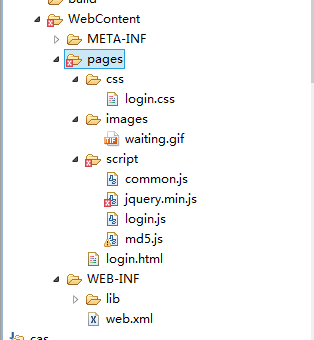
public void setNote(String note) {

this.note = note;

}

}

* + 1. 页面代码

页面采用ajax调用后台服务，如下所示：

* + - 1. login.css

@charset "utf-8";

html {font-size:62.5%;}

body {font-size:1.6rem; font-family:"microsoft yahei" ,"arial, helvetica, sans-serif";}

body, div, ul, li, h1, h2, h3, h4, h5, h6, pre, form, fieldset, input, textarea, p, blockquote, th, td { padding:0;margin:0;}

ul,li{ list-style: none;list-style-type:none;}

.login .hd {background:-webkit-gradient(linear,0 0,0 100%,from(#f8f8f8),to(#e6e6e6)); height:45px; -webkit-box-shadow:0 1px 1px 0 #c4c4c4; text-align:center; line-height:45px; font-weight:bold; position:relative;}

.login .hd .back\_shop{ display:block; position:absolute; top:14px; left:12px; border:8px solid transparent; border-right:8px solid #ef2185; width:0; height:0; content: ' '}.login .bd {margin:12px 20px;}

.login .bd .item {height:40px;}

.login .bd label {display:inline-block; width:80px;}

.login .bd input {background:-webkit-gradient(linear,0 0,0 100%,from(#f5f5f5),to(#fdfdfd)); border:solid 1px #DDD;-webkit-appearance:none;border-top-color:#bbb; height:33px;-webkit-box-sizing:border-box;box-sizing:border-box; width:100%; font-size:1.3rem; padding:0 2%;}

.login .ft {margin:0 20px;}

.login .ft .btnLogin {font-size:20px; background:#ef2185; height:40px; text-align:center; border:0; color:#fff; text-shadow:0 -1px 0 #8E5244;-webkit-box-shadow:0 -1px 0 #bf3210 inset; width:100%;}

#msg {width:300px; background:#000; border-radius:0 0 5px 5px; line-height:30px; color:#fff; text-align:center; position:fixed; left:50%; margin:0 0 0 -150px; top:0; z-index:100; opacity:0.8}

.slidedown {

animation:fromtop 0.5s ease;

-webkit-animation:fromtop 0.5s ease;

}

@keyframes fromtop {

from {transform:translateY(-100%)}

}

@-webkit-keyframes fromtop {

from {-webkit-transform:translateY(-100%)}

}

.slideup {

animation:totop 0.5s ease;

-webkit-animation:totop 0.5s ease;

}

@keyframes totop {

to {transform:translateY(-100%)}

}

@-webkit-keyframes totop {

to {-webkit-transform:translateY(-100%)}

}

* + - 1. login.html

<!DOCTYPE HTML>

<html>

<head>

<title>xxx系统</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">

<link rel="stylesheet" type="text/css" href="css/login.css">

</head>

<body>

<div class="login">

<div class="hd">xxx系统</div>

<div class="bd">

<div class="item"><input type="text" name="username" id="username" placeholder="请输入门店编号" /></div>

<div class="item"><input type="password" name="password" id="password" placeholder="请输入密码" /></div>

</div>

<div class="ft">

<input type="submit" value="登 录" id="btnLogin" class="btnLogin" />

</div>

</div>

<script src="script/jquery.min.js"></script>

<script src="script/md5.js"></script>

<script src="script/common.js"></script>

<script src="script/login.js"></script>

<script>

var G = {

timer: null,

timer2: null

};

function showmsg(str) {

if (!$("#msg").length) {

G.msg = $("<div id='msg' />").prependTo(document.body);

}

if (G.timer) {

clearTimeout(G.timer);

G.msg.hide();

G.msg.removeClass("slidedown").removeClass("slideup");

}

if (G.timer2) {

clearTimeout(G.timer2);

}

G.msg.html(str).show().addClass("slidedown");

G.timer = setTimeout(function () {

G.msg.removeClass("slidedown").addClass("slideup");

G.timer2 = setTimeout(function () {

G.msg.hide();

}, 500);

}, 3000);

}

</script>

</body>

</html>

* + - 1. login.js

//初始化数据

$(document).ready(function(){

var username = getLocalStorage("username");

if(!isNull(username)){

$('#username').val(username);

}

});

//登录

$("#btnLogin").click(function(){

var username = $('#username').val();

var password = $('#password').val();

if (isNull(username)) {

showmsg("请输入门店编号");

return false;

}

if (isNull(password)) {

showmsg("请输入您的密码");

return false;

}

//加密

password = $.md5(password);

showLoadingDialog("登录中，请稍后...");

$.ajax({

type: "POST",

async: true,

url: ctx + "/login.do",

data: {"shopNo":username,"password":password},

error: function(){

hideLoadingDialog();

showmsg("登录异常");

},

success: function(res){

hideLoadingDialog();

if(!isNull(res)){

//console.log(res.shopNo);

//console.log(res);

clearLocalStorage();

setLocalStorage("username",res.shopNo);//用于记住用户名，不清除

setLocalStorage("shopNo",res.shopNo);

setLocalStorage("shopId",res.id);

setLocalStorage("shopInfo",JSON.stringify(res));

showmsg("恭喜你，登录成功！");

} else {

showmsg("用户名或密码错误");

}

}

});

});

* + - 1. common.js

var ctx = '/SpringMvc-web';

var LOCAL\_STORAGE\_KEY\_PREFIX = "smi\_local\_storage";

var storage = window.localStorage;

function getLocalStorage(key) {

var value = storage.getItem(LOCAL\_STORAGE\_KEY\_PREFIX + key);

if (value == "" || value == null || value == undefined) {

return "";

}

return value;

}

function setLocalStorage(key, value) {

storage.setItem(LOCAL\_STORAGE\_KEY\_PREFIX + key, value);

}

function clearLocalStorage(){

//保留登录用户名

var username = getLocalStorage("username");

storage.clear();

setLocalStorage("username",username);

}

function removeStorage(key) {

storage.removeItem(LOCAL\_STORAGE\_KEY\_PREFIX + key);

}

function removesStorage(key) {

window.sessionStorage.removeItem(LOCAL\_STORAGE\_KEY\_PREFIX + key);

}

var dic = new Object();

dic.getJsonData = function (serviceCode, methodName) {

return dic.getFormData(serviceCode, methodName);

};

dic.getFormData = function (serviceCode, methodName, formId) {

var obj = new Object();

if (formId) {

var formData = $('#' + formId).serializeArray();

for (var index = 0; index < formData.length; index++) {

var p = formData[index];

if (obj[p["name"]] == undefined) {

obj[p["name"]] = p["value"];

}else {

obj[p["name"]] += ',' + p["value"];

}

}

}

obj['serviceCode'] = serviceCode;

obj['method'] = methodName;

return obj;

};

//ajax 方法封装

dic.getAjaxData = function (type, param, success, error, beforesend, complete) {

$.ajax({

type: type,

url: ctx\_url,

data: param,

dataType: 'json',

success: (success == "" || success == undefined) ? (function () {

}) : success,

beforeSend: (beforesend == "" || beforesend == undefined) ? (function () {

}) : success,

complete: (complete == "" || complete == undefined) ? (function () {

}) : complete,

error: (error == "" || error == undefined) ? (function () {

}) : error

});

};

//显示加载提示框

function showLoadingDialog(value){

if ($("#mask-loading").size()==0){

var style="border-radius: 5px; color: #FFF; font-size: 16px;width: 60%;background: rgba(98, 98, 98, 0.8) none repeat scroll 0 0 !important;background: #000;position: fixed;left: 20%;top: 45%;z-index: 9999;padding: 10px;box-sizing: border-box;text-align: center;";

var content;

if(isNull(value)){

content = "正在加载中...";

}else{

content = value;

}

$(document.body).append("<div style='" + style + "' id='mask-loading'><img style='vertical-align:middle;' src='" + ctx + "/pages/images/waiting.gif'/>" + content + "<br></div>");

}

}

//提示框

function showToast(text) {

if ($("#mask-toast").size()==0){

var style="border-radius: 8px; color: #FFF; font-size: 16px;width: 50%;background: rgba(98, 98, 98, 0.8) none repeat scroll 0 0 !important;background: #000;position: fixed;left: 30%;top: 45%;z-index: 19999;padding: 10px;box-sizing: border-box;text-align: center;";

$(document.body).append("<div style='" + style + "' id='mask-toast'>" + text + "</div>");

}

setTimeout(function(){

$("#mask-toast").remove();

},3000);

}

//隐藏加载提示框

function hideLoadingDialog() {

$("#mask-loading").remove();

}

//校验是否为空

function isNull(value) {

if (value == undefined || value == null || value == "") {

return true;

} else {

return false;

}

}

//电话号码校验

function isMobile(value) {

var mobile = /^1\d{10}$|^0\d{2,3}-?\d{7,8}(-?\d{1,4})?$/;

return mobile.test(value);

}

//解析序列化时间

function formatTime(val) {

var re = /-?\d+/;

var m = re.exec(val);

var d = new Date(parseInt(m[0]));

return d.format("yyyy-MM-dd hh:mm");

}

//格式化时间

Date.prototype.format = function (format){

var o = {

"M+": this.getMonth() + 1, //month

"d+": this.getDate(), //day

"h+": this.getHours(), //hour

"m+": this.getMinutes(), //minute

"s+": this.getSeconds(), //second

"q+": Math.floor((this.getMonth() + 3) / 3), //quarter

"S": this.getMilliseconds() //millisecond

};

if (/(y+)/.test(format)){

format = format.replace(RegExp.$1,(this.getFullYear() + "").substr(4 - RegExp.$1.length));

}

for (var k in o){

if (new RegExp("(" + k + ")").test(format)){

format = format.replace(RegExp.$1,RegExp.$1.length == 1 ? o[k] :("00" + o[k]).substr(("" + o[k]).length));

}

}

return format;

};

//跳转链接地址

function goToPage(url){

window.location.href = url;

}

//跳转登录

function goToLogin(){

window.location.href = ctx+"/pages/login.html";

}

//退出系统

function loginout(){

clearLocalStorage();

window.location.href = ctx+"/pages/login.html";

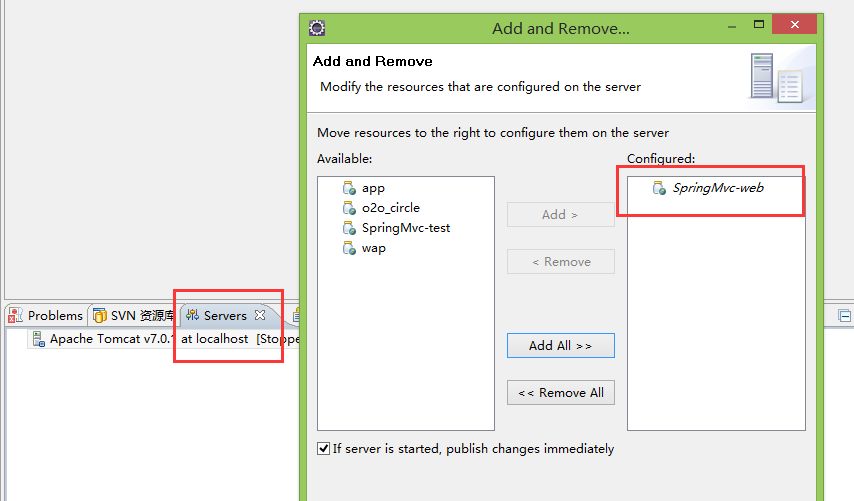
}

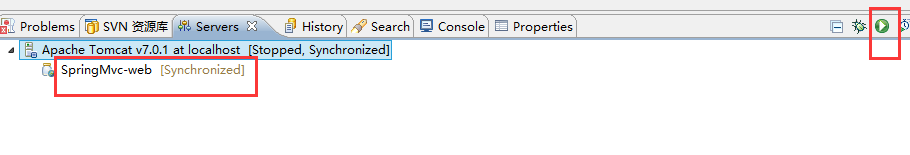
* + 1. 启动项目
       1. 编译项目

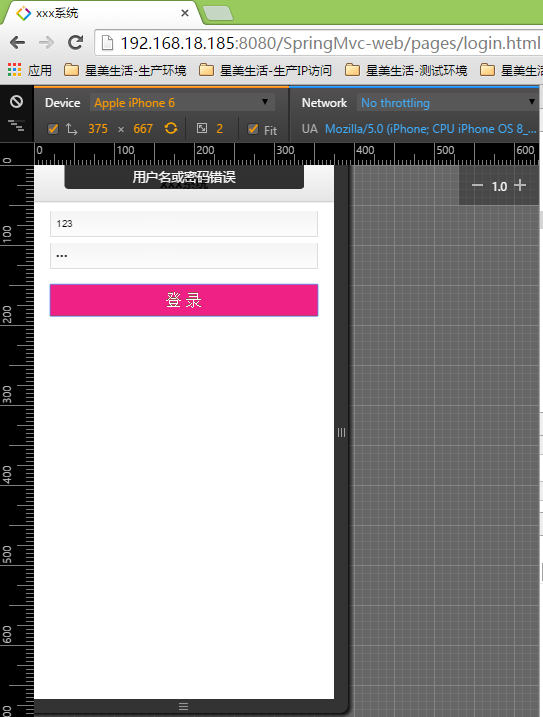
左键选中项目，点击菜单的project，然后点击clean，编译项目

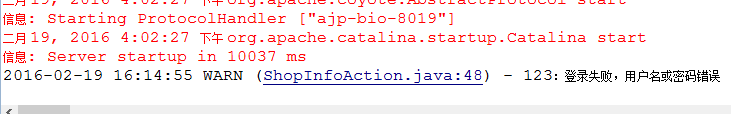
* + - 1. 部署项目

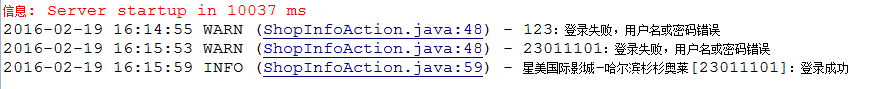
右键eclipse的server下的tomcat，（需要先配置好tomcat）

添加项目到server

完成后，右键点击server下的tomcat，clean以及publish

看到状态为synchronized时，启动项目。

查看后台日志，如下所示：

用户名密码正确，查看后台日志：

通过测试，SSI框架搭建成功！

备注：调试过程中，若有异常，可把错误信息复制给度娘找解决方案，多数情况是缺包，或者配置文件导致，恭祝大家学习愉快！