[**使用idea2017搭建SSM框架**](http://www.cnblogs.com/hackyo/p/6646051.html)

需要准备的环境：

　　idea 2017.1

　　jdk1.8

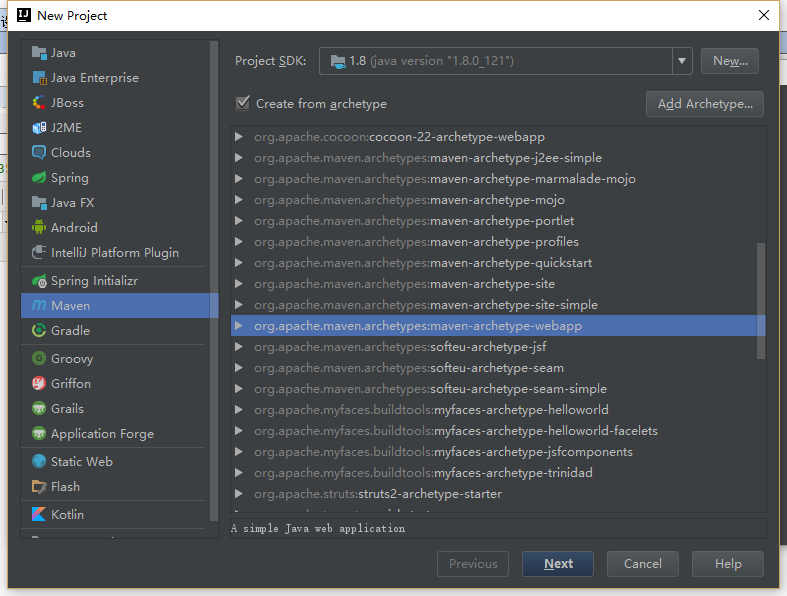
　　Maven 3.3.9

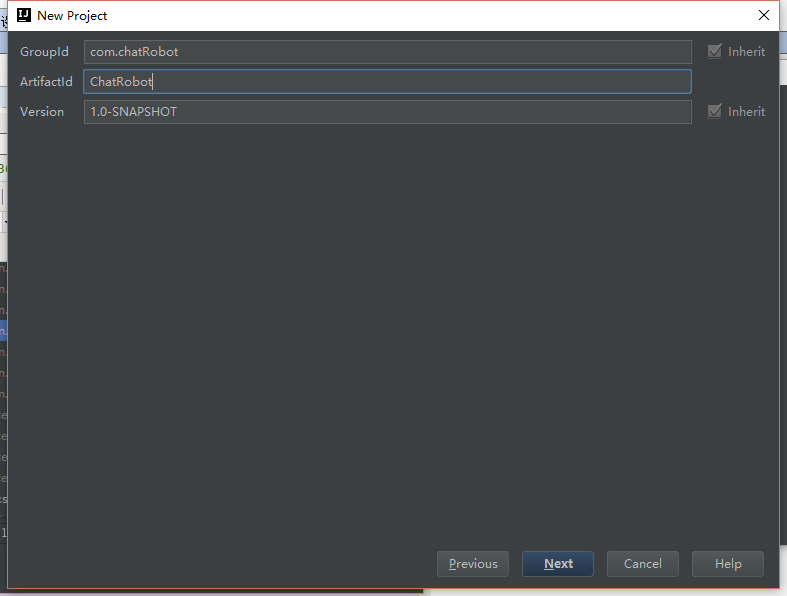
　　  请提前将idea与Maven、jdk配置好，本次项目用的都是比较新的

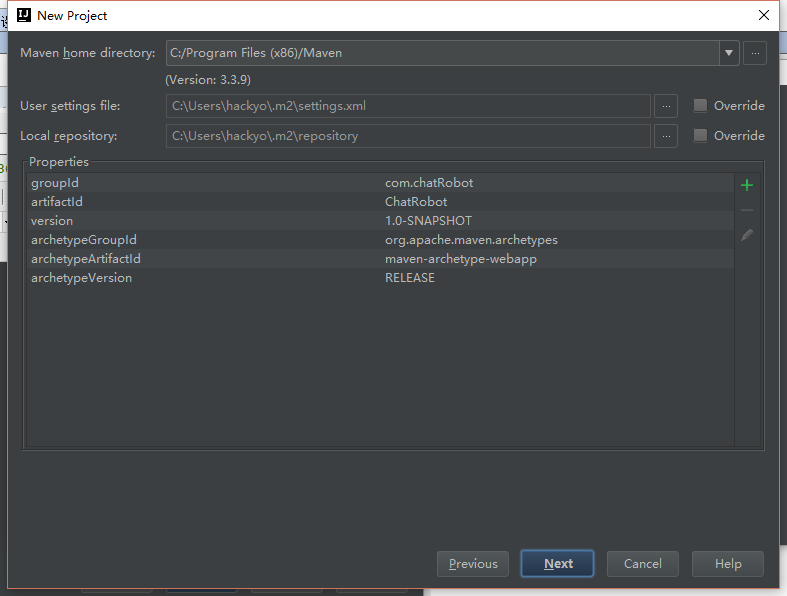
         注：配置完ide红线报错没关系！可以run！

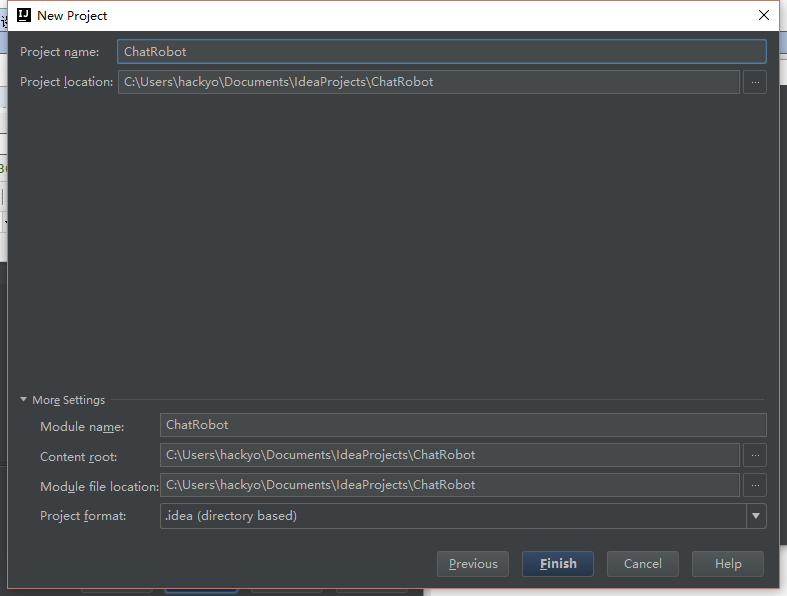
步骤：

一、首先使用idea新建一个Maven webapp项目





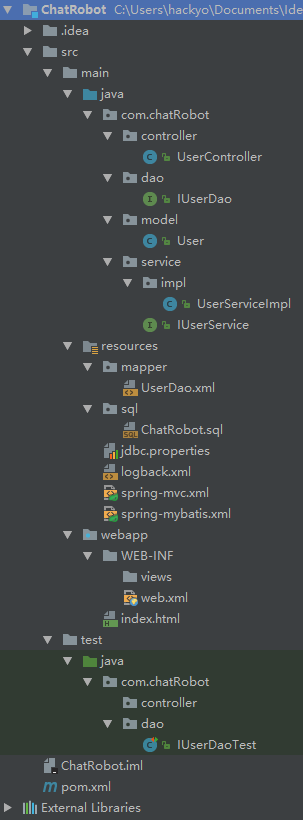




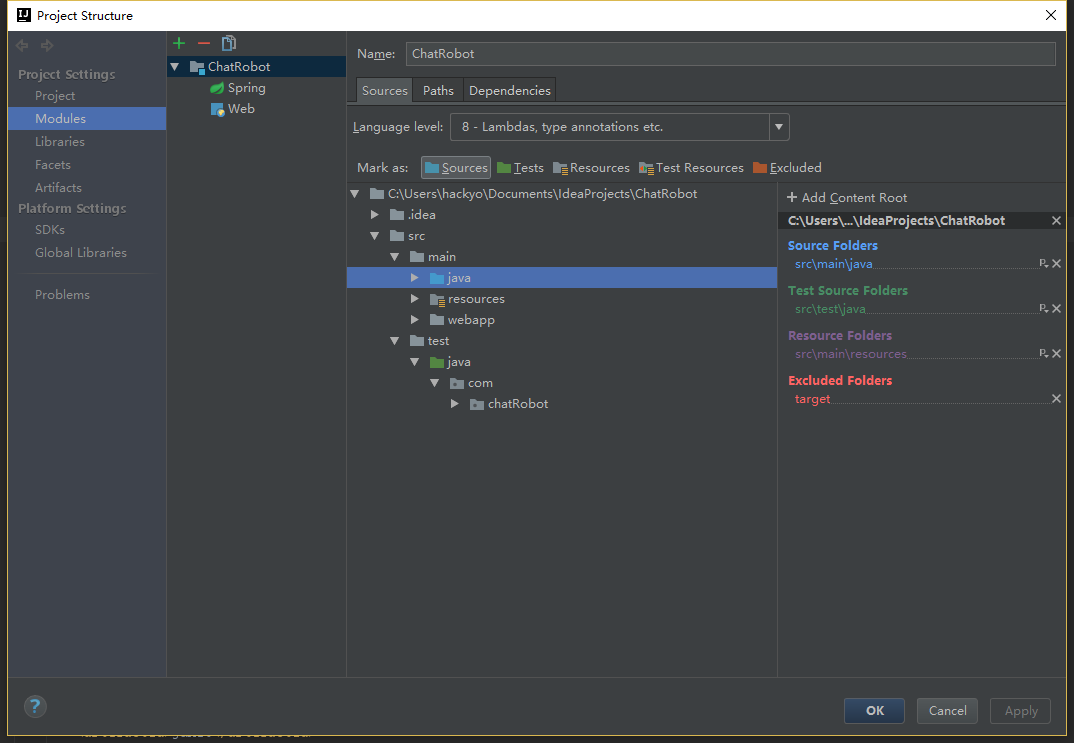
　　点击Finish，第一次搭建可能会很慢，甚至可能需要VPN才能搭建成功

二、搭建目录结构

　　我这里列出的是搭建完了之后所有的目录和文件，诸位先把目录文件建起来，然后我在给出文件内容



　　这里的目录建好之后还需要设置一下，让idea识别目录作用，选择File-Project Structure



　　设置完成后ok

三、配置文件内容

pom.xml

[复制代码](javascript:void(0);)

<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.chatRobot</groupId>

<artifactId>ChatRobot</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>ChatRobot Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<!-- 设置项目编码编码 -->

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<!-- spring版本号 -->

<spring.version>4.3.5.RELEASE</spring.version>

<!-- mybatis版本号 -->

<mybatis.version>3.4.1</mybatis.version>

</properties>

<dependencies>

<!-- java ee -->

<dependency>

<groupId>javax</groupId>

<artifactId>javaee-api</artifactId>

<version>7.0</version>

</dependency>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

<!-- 实现slf4j接口并整合 -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.2</version>

</dependency>

<!-- JSON -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.8.7</version>

</dependency>

<!-- 数据库 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.41</version>

<scope>runtime</scope>

</dependency>

<!-- 数据库连接池 -->

<dependency>

<groupId>com.mchange</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.5.2</version>

</dependency>

<!-- MyBatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<!-- mybatis/spring整合包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.1</version>

</dependency>

<!-- Spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<build>

<finalName>ChatRobot</finalName>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<!-- 设置JDK版本 -->

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

[复制代码](javascript:void(0);)

 注意右下角更新pom

logback.xml

[复制代码](javascript:void(0);)

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

<configuration debug="true">

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="debug">

<appender-ref ref="STDOUT"/>

</root>

</configuration>

[复制代码](javascript:void(0);)

这里可以控制输出格式和内容，有兴趣的可以自己设置

jdbc.properties

[复制代码](javascript:void(0);)

jdbc.driver=com.mysql.jdbc.Driver

#数据库地址

jdbc.url=jdbc:mysql://xxxxxxxxx:3306/ChatRobot?useUnicode=true&characterEncoding=utf8

#用户名

jdbc.username=xxxx

#密码

jdbc.password=xxxxx

#最大连接数

c3p0.maxPoolSize=30

#最小连接数

c3p0.minPoolSize=10

#关闭连接后不自动commit

c3p0.autoCommitOnClose=false

#获取连接超时时间

c3p0.checkoutTimeout=10000

#当获取连接失败重试次数

c3p0.acquireRetryAttempts=2

[复制代码](javascript:void(0);)

spring-mybatis.xml

[复制代码](javascript:void(0);)

<?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"

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.xsd

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

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

<!-- 扫描service包下所有使用注解的类型 -->

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

<!-- 配置数据库相关参数properties的属性：${url} -->

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

<!-- 数据库连接池 -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${jdbc.driver}"/>

<property name="jdbcUrl" value="${jdbc.url}"/>

<property name="user" value="${jdbc.username}"/>

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

<property name="maxPoolSize" value="${c3p0.maxPoolSize}"/>

<property name="minPoolSize" value="${c3p0.minPoolSize}"/>

<property name="autoCommitOnClose" value="${c3p0.autoCommitOnClose}"/>

<property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"/>

<property name="acquireRetryAttempts" value="${c3p0.acquireRetryAttempts}"/>

</bean>

<!-- 配置SqlSessionFactory对象 -->

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

<!-- 注入数据库连接池 -->

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

<!-- 扫描model包 使用别名 -->

<property name="typeAliasesPackage" value="com.chatRobot.model"/>

<!-- 扫描sql配置文件:mapper需要的xml文件 -->

<property name="mapperLocations" value="classpath:mapper/\*.xml"/>

</bean>

<!-- 配置扫描Dao接口包，动态实现Dao接口，注入到spring容器中 -->

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

<!-- 注入sqlSessionFactory -->

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

<!-- 给出需要扫描Dao接口包 -->

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

</bean>

<!-- 配置事务管理器 -->

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<!-- 注入数据库连接池 -->

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

</bean>

<!-- 配置基于注解的声明式事务 -->

<tx:annotation-driven transaction-manager="transactionManager"/>

</beans>

[复制代码](javascript:void(0);)

spring-mvc.xml

[复制代码](javascript:void(0);)

<?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.xsd

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

http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">

<!-- 扫描web相关的bean -->

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

<!-- 开启SpringMVC注解模式 -->

<mvc:annotation-driven/>

<!-- 静态资源默认servlet配置 -->

<mvc:default-servlet-handler/>

<!-- 配置jsp 显示ViewResolver -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>

<property name="prefix" value="/WEB-INF/views/"/>

<property name="suffix" value=".jsp"/>

</bean>

</beans>

[复制代码](javascript:void(0);)

web.xml

[复制代码](javascript:void(0);)

<?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>ChatRobot</display-name>

<description>ChatRobot\_Alpha\_0.0.1</description>

<!-- 编码过滤器 -->

<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>

<!-- 配置DispatcherServlet -->

<servlet>

<servlet-name>SpringMVC</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- 配置springMVC需要加载的配置文件-->

<init-param>

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

<param-value>classpath:spring-\*.xml</param-value>

</init-param>

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

<async-supported>true</async-supported>

</servlet>

<servlet-mapping>

<servlet-name>SpringMVC</servlet-name>

<!-- 匹配所有请求，此处也可以配置成 \*.do 形式 -->

<url-pattern>/</url-pattern>

</servlet-mapping>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

[复制代码](javascript:void(0);)

到这里基本上环境就搭建完成了，下面开始测试

四、测试

先导入一份数据库测试文件

[复制代码](javascript:void(0);)

CREATE TABLE `user` (

`id` int(**11**) NOT NULL AUTO\_INCREMENT COMMENT '用户ID',

`email` varchar(**255**) NOT NULL COMMENT '用户邮箱',

`password` varchar(**255**) NOT NULL COMMENT '用户密码',

`username` varchar(**255**) NOT NULL COMMENT '用户昵称',

`role` varchar(**255**) NOT NULL COMMENT '用户身份',

`status` int(**1**) NOT NULL COMMENT '用户状态',

`regTime` datetime NOT NULL COMMENT '注册时间',

`regIp` varchar(**255**) NOT NULL COMMENT '注册IP',

PRIMARY KEY (`id`),

UNIQUE KEY `email` (`email`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=**2** DEFAULT CHARSET=utf8;

INSERT INTO `user` VALUES ('1', 'xxx', 'xxxxx', 'xxxxx', 'root', '0', '2017-03-28 09:40:31', '127.0.0.1');

SET FOREIGN\_KEY\_CHECKS=**1**;

[复制代码](javascript:void(0);)

接下来配置类

UserController

[复制代码](javascript:void(0);)

package com.chatRobot.controller;

import javax.servlet.http.HttpServletRequest;

import com.chatRobot.model.User;

import com.chatRobot.service.IUserService;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

@Controller

@RequestMapping("/user")

public class UserController {

@Resource

private IUserService userService;

@RequestMapping("/showUser.do")

public void selectUser(HttpServletRequest request, HttpServletResponse response) throws IOException {

request.setCharacterEncoding("UTF-8");

response.setCharacterEncoding("UTF-8");

long userId = Long.parseLong(request.getParameter("id"));

User user = this.userService.selectUser(userId);

ObjectMapper mapper = new ObjectMapper();

response.getWriter().write(mapper.writeValueAsString(user));

response.getWriter().close();

}

}

[复制代码](javascript:void(0);)

IUserDao

[复制代码](javascript:void(0);)

package com.chatRobot.dao;  
  
import com.chatRobot.model.User;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface IUserDao {  
  
 User selectUser(long id);  
  
}

[复制代码](javascript:void(0);)

User

[复制代码](javascript:void(0);)

package com.chatRobot.model;

import java.util.Date;

public class User {

private long id;

private String email;

private String password;

private String username;

private String role;

private int status;

private Date regTime;

private String regIp;

public long getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public Date getRegTime() {

return regTime;

}

public void setRegTime(Date regTime) {

this.regTime = regTime;

}

public String getRegIp() {

return regIp;

}

public void setRegIp(String regIp) {

this.regIp = regIp;

}

}

[复制代码](javascript:void(0);)

IUserService

[复制代码](javascript:void(0);)

package com.chatRobot.service;

import com.chatRobot.model.User;

public interface IUserService {

public User selectUser(long userId);

}

[复制代码](javascript:void(0);)

UserServiceImpl

[复制代码](javascript:void(0);)

package com.chatRobot.service.impl;

import com.chatRobot.dao.IUserDao;

import com.chatRobot.model.User;

import com.chatRobot.service.IUserService;

import org.springframework.stereotype.Service;

import javax.annotation.Resource;

@Service("userService")

public class UserServiceImpl implements IUserService {

@Resource

private IUserDao userDao;

public User selectUser(long userId) {

return this.userDao.selectUser(userId);

}

}

[复制代码](javascript:void(0);)

UserDao.xml

[复制代码](javascript:void(0);)

<?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">

<!-- 设置为IUserDao接口方法提供sql语句配置 -->

<mapper namespace="com.chatRobot.dao.IUserDao">

<select id="selectUser" resultType="User" parameterType="long">

SELECT \* FROM user WHERE id = #{id}

</select>

</mapper>

[复制代码](javascript:void(0);)

然后新建个测试类，来测试mybatis

IUserDaoTest

[复制代码](javascript:void(0);)

package com.chatRobot.dao;

import com.chatRobot.model.User;

import org.junit.Test;

import org.junit.runner.RunWith;

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

import org.springframework.test.context.ContextConfiguration;

import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

// 加载spring配置文件

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration({"classpath:spring-mybatis.xml"})

public class IUserDaoTest {

@Autowired

private IUserDao dao;

@Test

public void testSelectUser() throws Exception {

long id = 1;

User user = dao.selectUser(id);

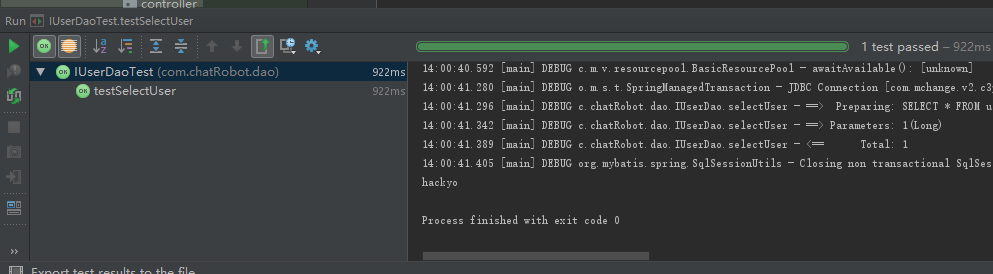
System.out.println(user.getUsername());

}

}

[复制代码](javascript:void(0);)

运行后结果应该是会在控制台输出id为1的用户名，没成功的话就找BUG去吧...



继续新建个页面来测试springmvc和mybatis

index.html

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>test</title>

</head>

<script>

function selectUser() {

var xmlhttp = new XMLHttpRequest();

xmlhttp.onreadystatechange = function () {

if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {

document.getElementById("test").innerHTML = xmlhttp.responseText;

}

}

xmlhttp.open("POST", "user/showUser.do", true);

xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

xmlhttp.send("id=1");

}

</script>

<body>

<p id="test">Hello World!</p>

<button type="button" onclick="selectUser()">onclick test</button>

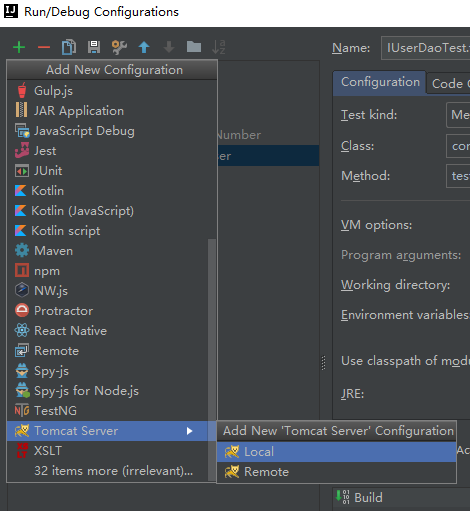
</body>

</html>

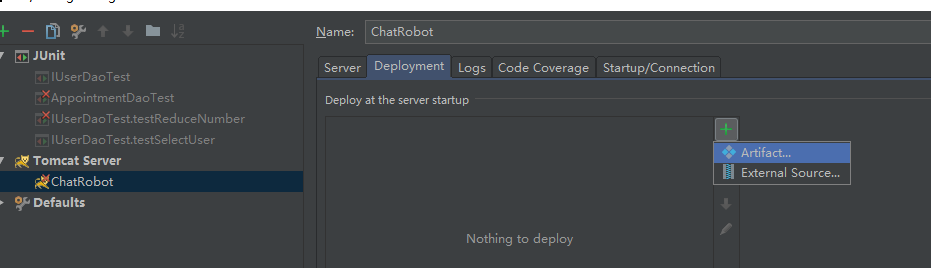
[复制代码](javascript:void(0);)

新建完成后配置项目运行环境，点击Run-Edit Configurations...

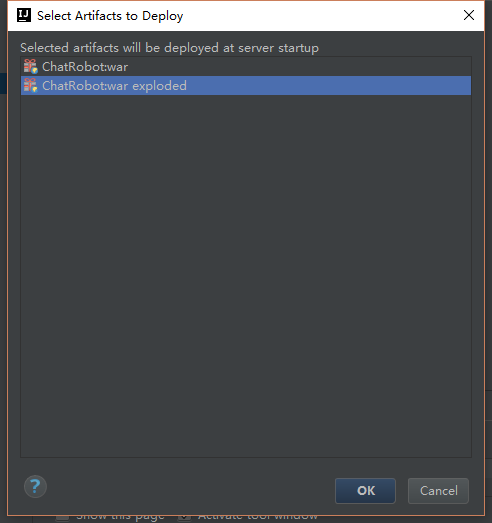
点击加号新建运行环境，选择Tomcat Server-Local



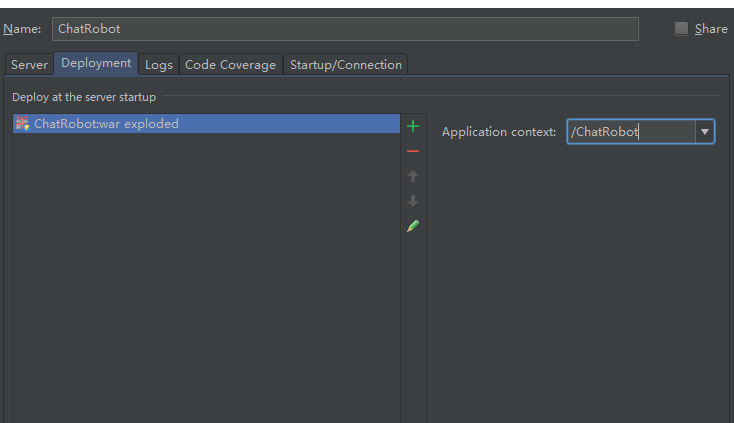
选中新建好的服务器，右边选择Deployment,点击加号-Atifact...



选择第二项

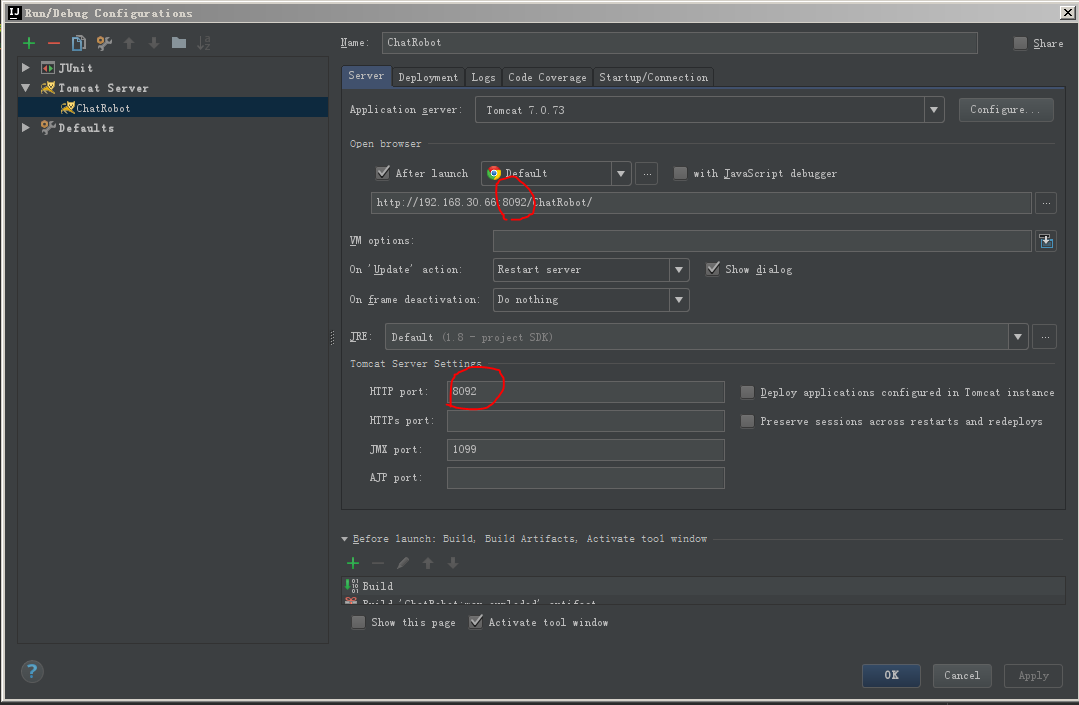


然后在右边Application context配置你的项目名



最后运行项目，在打开的页面中点击按钮测试，成功的话会在页面上显示id为1的用户信息

Open browser 中的端口8092**必须**要和 Tomcat Server Settings --> HTTP port 中的端口一致 8092



CentOS7 jdk1.8 tomcat7 war包部署：

ifconfig

48 java -version

49 env

50 whereis jdk

51 whereis java

52 echo JAVA\_HOME

53 echo $JAVA\_HOME

54 ls

55 cd local

56 dir

57 ll

58 ll -la /

59 cat /etc/profile

60 echo $JAVA\_HOME

61 echo $CLASSPATH

62 echo $PATH

63 whereis java

64 ls

65 tar -zxvf jdk-8u171-linux-x64.tar.gz

66 ls

67 mkdir -p /usr/local/java

68 mv jdk1.8.0\_171/ /usr/local/java/

69 ls

70 vi /etc/profile

71 ll /usr/local/java/jdk1.8.0\_171/

72 vi /etc/profile

#jdk

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_171

export CLASSPATH=.:${JAVA\_HOME}/jre/lib/rt.jar:${JAVA\_HOME}/lib/tools.jar

export PATH=$PATH:${JAVA\_HOME}/bin

73 source /etc/profile

74 java -version

75 rm -f /usr/bin/java

76 rm -f /usr/bin/javac

77 rm -f /etc/alternatives/java

78 rm -f /etc/alternatives/javac

79 cat /usr/bin/ja\*

80 java -version

81 source /etc/profile

82 java -version

83 ls

84 tar -zxvf apache-tomcat-7.0.86.tar.gz

85 ls

86 mv apache-tomcat-7.0.86 /opt/tomcat

87 cd /opt/

88 ls

89 cd tomcat/

90 ls

91 vi conf/server.xml

92 cd bin/

93 ls

94 ./startup.sh

95 ls

96 service iptables stop

97 service ip6tables stop

98 ifconfig

99 cd ..

100 ls

101 cd logs/

102 ls

103 tail -100f catalina.out

104 chkconfig iptables off

105 stop iptables.service

106 systemctl disable firewalld

107 service iptables status

108 systemctl iptables status

109 getenforce

110 /usr/sbin/sestatus -v

111 setenforce 0

112 vi /etc/selinux/config

SELINUX=disabled

113 cd ..

114 ls

115 cd b in

116 cd bin

117 ./shutdown.sh

118 cd ..

119 ls

120 cd webapps/

121 ls

122 cp /var/local/ChatRobot.war .

123 ls

124 jar ChatRobot.war

125 jar -t ChatRobot.war

126 ls

127 reboot

128 cd /opt/tomcat/webapps/

129 ls

130 history

**Mybatis-generator 反向生成代码**

使用这个插件可以快速生成一些代码,包含 实体类/Mapper接口/\*Mapper.xml文件

首先,我们需要搭建一个Maven的项目。

在pom.xml中添加代码

<plugins>

<plugin>

<!--Mybatis-generator插件,用于自动生成Mapper和POJO-->

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-maven-plugin</artifactId>

<version>1.3.2</version>

<configuration>

<!--配置文件的位置-->

<configurationFile>yourLocation/mybatis-generator-config.xml</configurationFile>

<verbose>true</verbose>

<overwrite>true</overwrite>

</configuration>

<executions>

<execution>

<id>Generate MyBatis Artifacts</id>

<goals>

<goal>generate</goal>

</goals>

</execution>

</executions>

<dependencies>

<dependency>

<groupId>org.mybatis.generator</groupId>

<artifactId>mybatis-generator-core</artifactId>

<version>1.3.2</version>

</dependency>

</dependencies>

</plugin>

</plugins>

注意,plugins标签是build标签的子标签

添加好之后,我们就需要配置mybatis-generator-config.xml配置文件中的代码

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

<!DOCTYPE generatorConfiguration PUBLIC

"-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"

"http://mybatis.org/dtd/mybatis-generator-config\_1\_0.dtd" >

<generatorConfiguration>

<!-- 本地数据库驱动程序jar包的全路径 -->

<classPathEntry location=""/>

<context id="context" targetRuntime="MyBatis3">

<commentGenerator>

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

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

</commentGenerator>

<!-- 数据库的相关配置 -->

<jdbcConnection driverClass="" connectionURL="" userId="" password=""/>

<javaTypeResolver>

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

</javaTypeResolver>

<!-- 实体类生成的位置 -->

<javaModelGenerator targetPackage="目标包" targetProject="目标项目classpath">

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

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

</javaModelGenerator>

<!-- \*Mapper.xml 文件的位置 -->

<sqlMapGenerator targetPackage="目标包" targetProject="目标项目classpath">

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

</sqlMapGenerator>

<!-- Mapper 接口文件的位置 -->

<javaClientGenerator targetPackage="目标包" targetProject="目标项目classpath" type="XMLMAPPER">

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

</javaClientGenerator>

<!-- 相关表的配置 -->

<table tableName="表名" enableCountByExample="false" enableDeleteByExample="false" enableSelectByExample="false"

enableUpdateByExample="false"/>

</context>

</generatorConfiguration>

随后,在idea的右侧栏点击Maven,选中添加的Mybatis-generator插件并运行,就可以得到相应的代码啦~

运行插件

