**作业一 自己总结的SSM知识点**，github仓库地址：https://github.com/Grxhyyt/qinjiu.git

# Spring

1.控制反转(IoC)

所谓控制反转，就是一个对象的使用不再是手动创建，而是被动的被传递 进来。这么做的好处就在于–解耦，尤其是面对接口编程时。看起来相差不大，但IoC更加的有利于代码的解耦。使用IoC后就由老的面向对象编程变成了面向接口编程，我在使用MsgService的时候，他的实际实现是对我屏蔽的。这对解耦、拓展、维护是一个很好的基础。我要更改sendMsg的逻辑，只需再提供一个新的该接口的实现替换上就好，其他调用的地方都无需改动。

2.面向切面(AOP)

AOP使程序猿更加关注主体业务的开发，一些公共的可抽象的业务比如日志、事务等，在编码时都无须额外关注。这样能更好的实现单一原则、高内聚。

## Spring框架的七大模块

Spring Core：框架的最基础部分，提供 IoC 容器，对 bean 进行管理。

Spring Context：继承BeanFactory，提供上下文信息，扩展出JNDI、EJB、电子邮件、国际化等功能。

Spring DAO：提供了JDBC的抽象层，还提供了声明性事务管理方法。

Spring ORM：提供了JPA、JDO、Hibernate、MyBatis 等ORM映射层.

Spring AOP：集成了所有AOP功能

Spring Web：提供了基础的 Web 开发的上下文信息，现有的Web框架，如JSF、Tapestry、Structs等，提供了集成

Spring Web MVC：提供了 Web 应用的 Model-View-Controller 全功能实现。

# SpringMVC

SpringMVC是在Spring的一部分，是一个WEB开发框架。

MVC指的就是MVC设计模式，Model、View、Controller。具体是什么我也不赘述了。

SpringMVC核心处理流程：

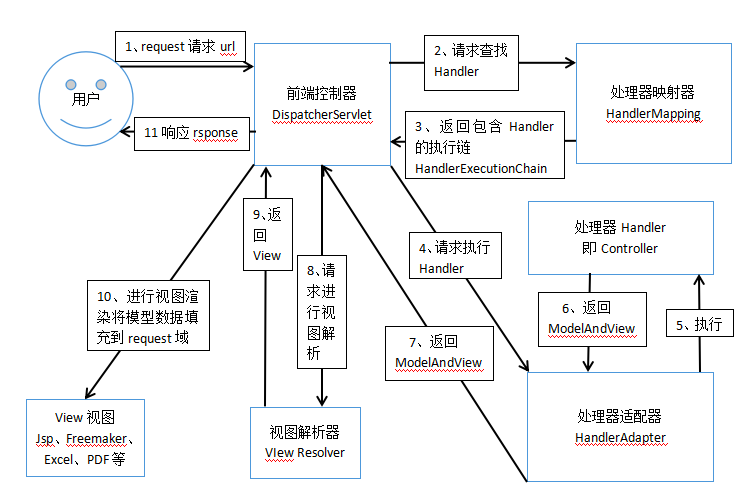
1、DispatcherServlet前端控制器接收发过来的请求，交给HandlerMapping处理器映射器

2、HandlerMapping处理器映射器，根据请求路径找到相应的HandlerAdapter处理器适配器（处理器适配器就是那些拦截器或Controller）

3、HandlerAdapter处理器适配器，处理一些功能请求，返回一个ModelAndView对象（包括模型数据、逻辑视图名）

4、根据返回的ModelAndView，选择一个适合的ViewResolver（必须是已经注册到Spring容器中的ViewResolver–视图解析器)返回给DispatcherServlet ；

5、DispatcherServlet根据ViewResolver视图解析器,找到ModelAndView对象指定的视图对象



# Mybatis

通过MyBatis提供的缓存机制来实现缓存：

一级缓存：

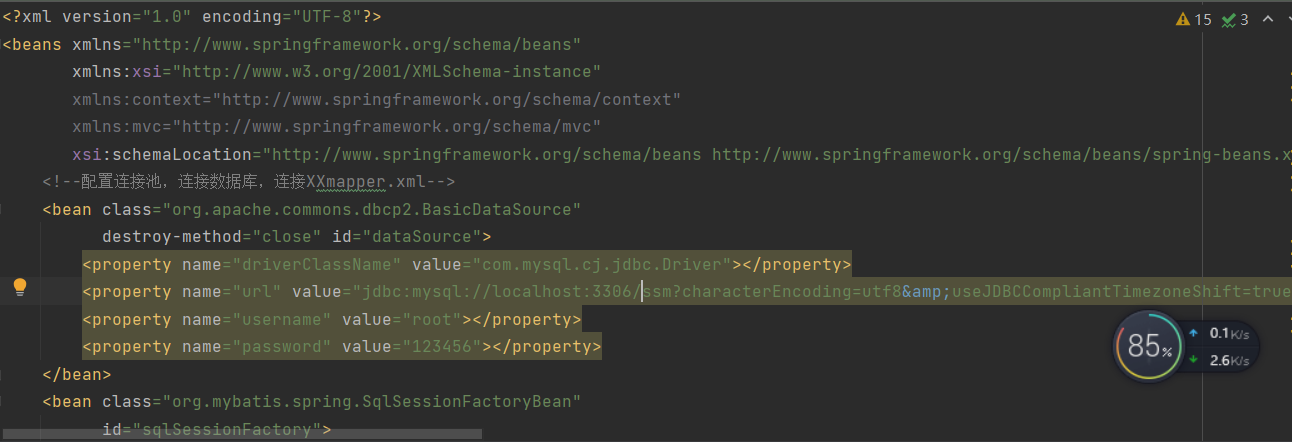
缓存只在一个事务中有效，即同一个事务中先后执行多次同一个查询，只在第一次真正去查库，并将结果缓存，之后的查询都直接获取缓存中的中数据。如果是不同的事务，则缓存无效。

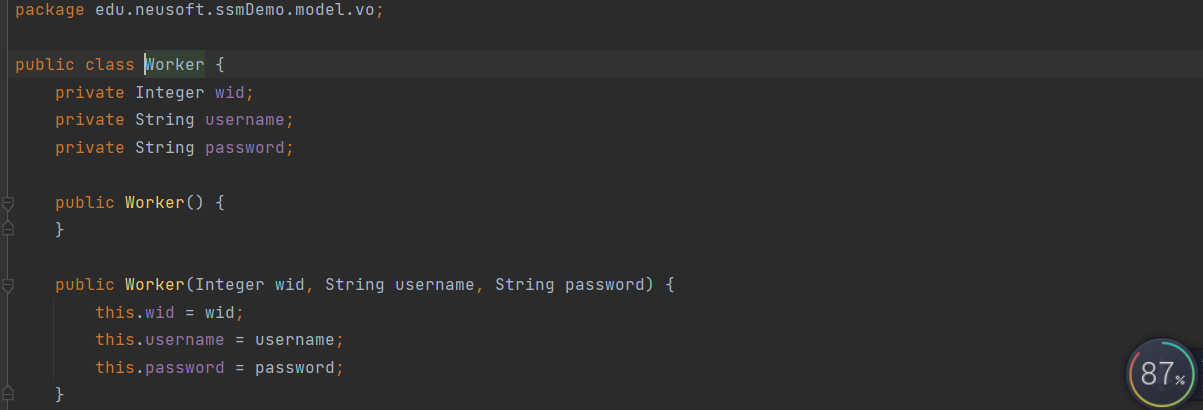
二级缓存：

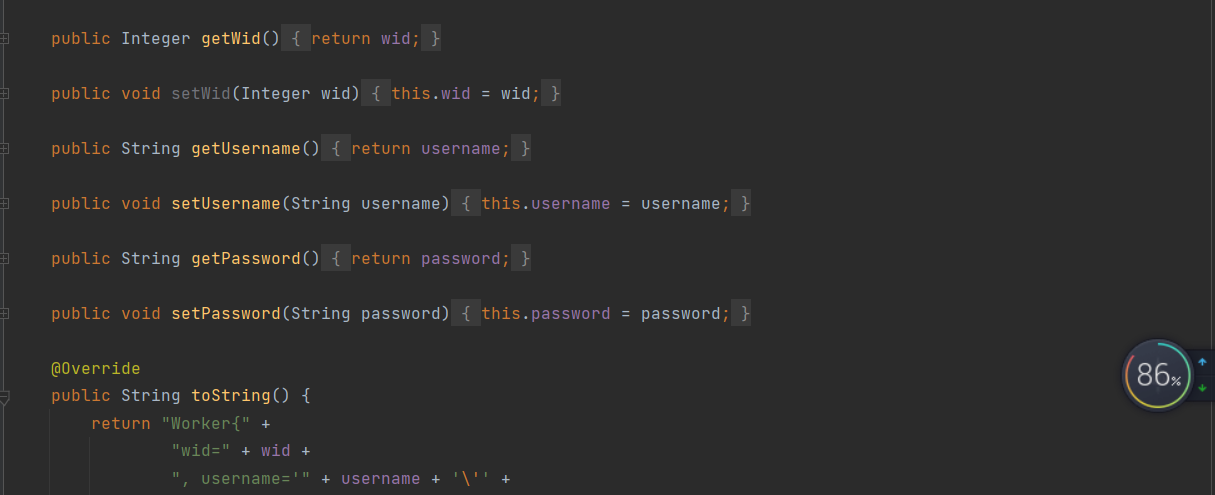
缓存在全局有效，一个事务查询一个sql得到结果，会被缓存起来，之后只要缓存未被清楚，则其他事务如果查询同一个sql，得到的将会是之前缓存的结果。二级缓存作用范围大，作用时间长，可能造成的危害也更大，所以在开发中一般很少启用Mybatis的二级缓存。

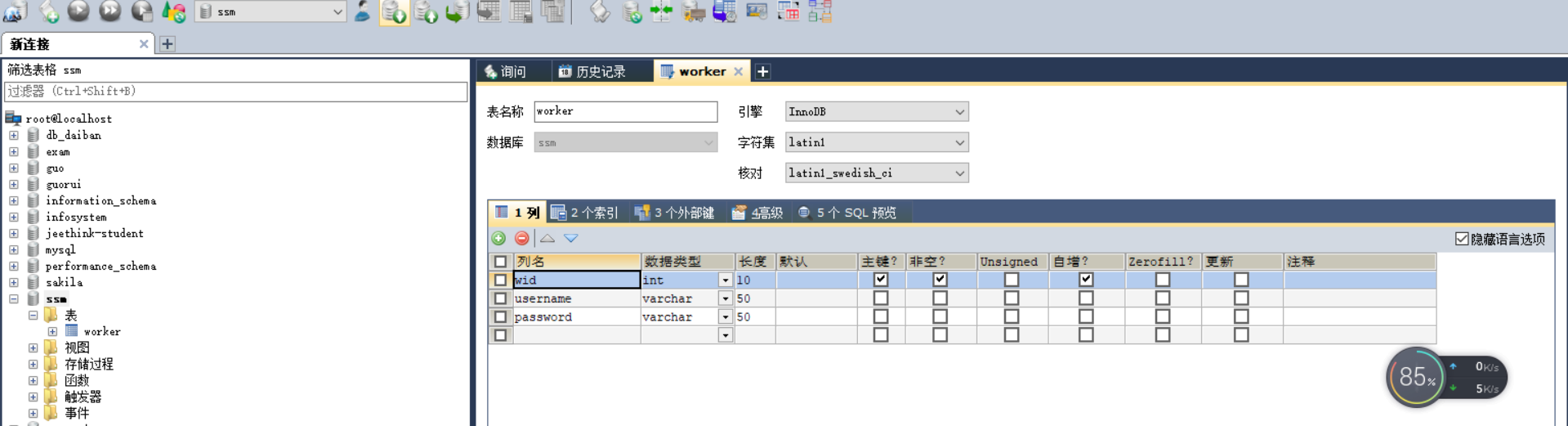
**作业二 报告**

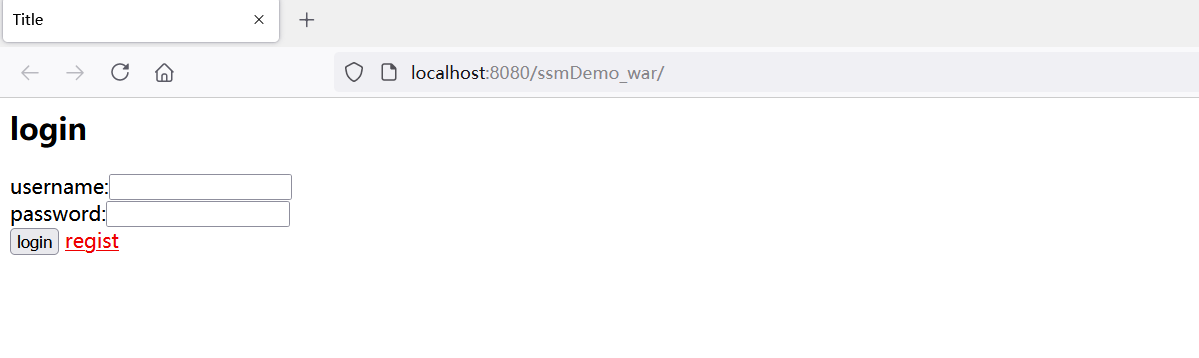
**SSM用户登录报告**

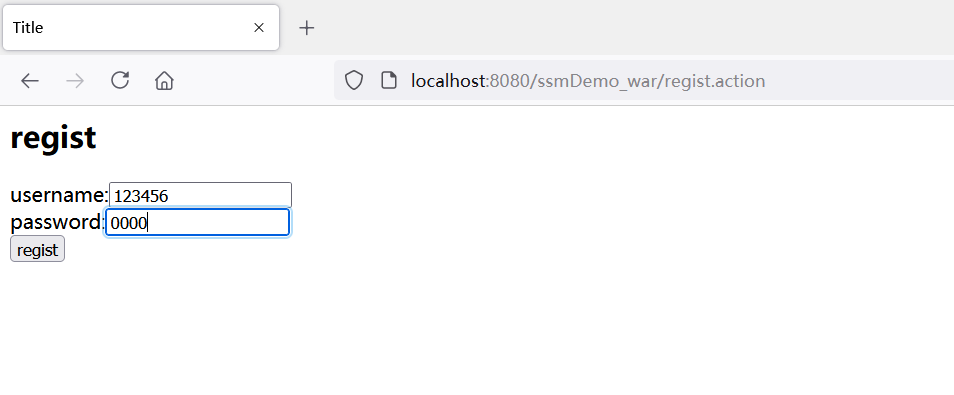


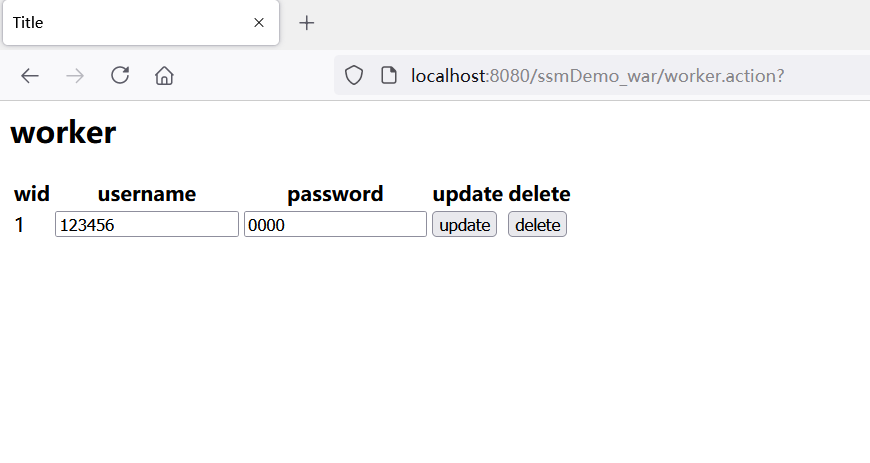
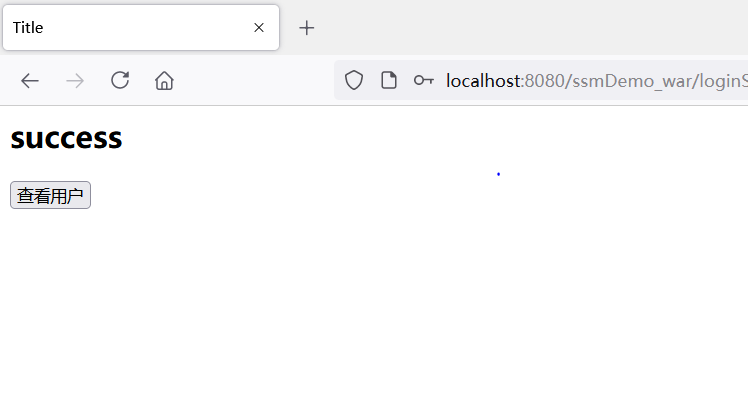












### 1.课题的内容

构建ssm环境，并且实现与mysql数据交互，实现用户表的增删改查，实现登陆，注册功能，登录对数据库进行查找，注册功能对数据库进行增加。

### 需求分析

登录

注册

### 系统数据需求

定义数据表，user

该表有个四段

ID用户id，主键，自增，不能为空，自动生成

设置了Username用户名

设置Password密码

### 详细

I bean: user

Private integer id;

Private string username

Private String password

Private Integer age

提供getter和setter

Controller: usercontroller

@controller

@ruquestmapping(“user”)

Public class UserAction

private IUserSV userService =(IUserSV)ApplicationContextUtil.getBeam("IUserSV");;//处理展示请求的路径

@RequestMapping("/showUser")public String toIndex(Model model){

int userld = Integer.parseInt(request.getParameter("id"));int userId= 1;

model.addAttribute( "userList" , userList.get(O));return "/user/main";

public class Worker {  
 private Integer wid;  
 private String username;  
 private String password;public Worker() {  
 }  
  
 public Worker(Integer wid, String username, String password) {  
 this.wid = wid;  
 this.username = username;  
 this.password = password;  
 }  
  
 public Integer getWid() {  
 return wid;  
 }  
  
 public void setWid(Integer wid) {  
 this.wid = wid;  
 }  
  
 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;  
 }  
  
 @Override  
 public String toString() {  
 return "Worker{" +  
 "wid=" + wid +  
 ", username='" + username + '\'' +  
 ", password='" + password + '\'' +  
 '}';  
 }  
}