web实验

servlet层

register函数

```java
//1.获取数据
Map<String, String[]> map = request.getParameterMap();
//2.封装对象
User user = new User();
try {
    BeanUtils.populate(user,map);
} catch (IllegalAccessException e) {
    e.printStackTrace();
} catch (InvocationTargetException e) {
    e.printStackTrace();
}
//3.调用 service 完成注册
//UserService service = new UserServiceImpl();
boolean flag = service.regist(user);
ResultInfo info = new ResultInfo();
//4.响应结果
if(flag){
    //注册成功
    info.setFlag(true);
}else{
    //注册失败
    info.setFlag(false);
    info.setErrorMsg("注册失败!");
}
//将 info 对象序列化为 json
ObjectMapper mapper = new ObjectMapper();
String json = mapper.writeValueAsString(info);
//将 json 数据写回客户端
//设置 content-type
response.setContentType("application/json;charset=utf-8");
response.getWriter().write(json);
```

service层

regist函数（注册）

```java
public boolean regist(User user) {
    //1.根据用户名查询用户对象
    User u = userDao.findByUsername(user.getUsername());
    //判断 u 是否为 null
    if(u != null){
        //用户名存在，注册失败
        return false;
    }
    //2.保存用户信息
    userDao.save(user);
    return true;
}
```

Dao层（数据库）

findByUsername函数（查找用户是否存在）

```java
private JdbcTemplate template = new JdbcTemplate(JDBCUtils.getDataSource());
1 usage
@Override
public User findByUsername(String username) {
    User user = null;
    try {
        //1.定义 sql
        String sql = "select * from tab_user where username = ?";
        //2.执行 sql BeanPropertyRowMapper将数据库查询结果转换为Java类对象，参数是java类对象
        //这里使用queryForObject是因为queryForObject的返回值是一个对象，而query的返回值是一个List
        user = template.queryForObject(sql, new BeanPropertyRowMapper<User>(User.class),
                username);
    } catch (Exception e) {
    }
    return user;
}
```

save（保存用户信息）

```java
public void save(User user) {
    //如果数据库中没有该条数据，则进行数据保存
    //1.定义 sql
    String sql = "insert into tab_user(username,password,name,birthday,sex,telephone,email) values(?,?,?,?,?,?,?)";
    //2.执行 sql
    template.update(sql,user.getUsername(),
            user.getPassword(),
            user.getName(),
            user.getBirthday(),
            user.getSex(),
            user.getTelephone(),
            user.getEmail());
}
```

findByUsernameAndPassword（根据用户名和密码查询）

```java
public User findByUsernameAndPassword(String username, String password) {
    User user = null;
    try {
        //1.定义 sql
        String sql = "select * from tab_user where username = ? and password = ?";
        //2.执行 sql
        user = template.queryForObject(sql, new BeanPropertyRowMapper<User>(User.class),
                username,password);
    } catch (Exception e) {
    }
    return user;
}
```

（后加的，了解一下怎么处理多个参数的情况）

前端

校验函数

```
//校验用户名
//单词字符，长度 8 到 20 位
function checkUsername() {
    //1.获取用户名值
    var username = $("#username").val();
    //2.定义正则
    var reg_username = /^\w{8,20}$/;
    //3.判断，给出提示信息
    var flag = reg_username.test(username);
    if(flag){
        //用户名合法
        $("#username").css("border","");
    }else{
        //用户名非法,加一个红色边框
        $("#username").css("border","1px solid red");
    }
    return flag;
}
```

ajax请求

```
$(function () {
    //当表单提交时，调用所有的校验方法
    $("#registerForm").submit(function(){
        if(checkUsername() && checkPassword() && checkEmail())
        {
            //serialize:序列化表单值 格式如-FirstName=Bill&LastName=Gates
            //function是请求成功后的回调函数
            $.post("user/register",$(this).serialize(),function (data)
            {
                if(data.flag)
                {
                    location.href="register_ok.html";
                }
                else {
                    $("#errorMsg").html(data.errorMsg);
                }
            });
        }
        //如果校验未通过，则不允许提交
        return false;

    });
    //当某一个组件失去焦点是，调用对应的校验方法
    $("#username").blur(checkUsername);
    $("#password").blur(checkPassword);
    $("#email").blur(checkEmail);
});
```