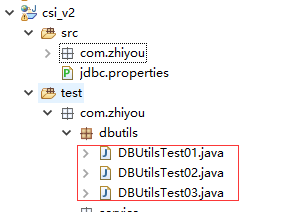
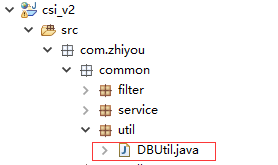
# DBUtils工具类使用





# 重构DBUtil工具类



增加以下泛型方法：

/\*\*

\* 根据条件查询列表

\* **@param** sql

\* **@param** clazz

\* **@param** param

\* **@return**

\*/

**public** **static** <T> List<T> query(String sql,Class<T> clazz, Object... param) {

**try** {

QueryRunner qr = **new** QueryRunner(DBUtil.*getDateSource*());

**return** qr.query(sql,**new** BeanListHandler<T>(clazz), param);

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** **new** ArrayList<>();

}

/\*\*

\* 根据条件返回单个对象

\* **@param** sql

\* **@param** clazz

\* **@param** param

\* **@return**

\*/

**public** **static** <T> T queryOne(String sql,Class<T> clazz, Object... param) {

**try** {

QueryRunner qr = **new** QueryRunner(DBUtil.*getDateSource*());

**return** qr.query(sql,**new** BeanHandler<T>(clazz), param);

} **catch** (SQLException e) {

e.printStackTrace();

}

**return** **null**;

}

/\*\*

\* 新增 修改 删除

\* **@param** sql

\* **@param** param

\* **@throws** Exception

\*/

**public** **static** **void** update(String sql, Object... param) **throws** Exception {

QueryRunner qr = **new** QueryRunner(DBUtil.*getDateSource*());

qr.update(sql, param);

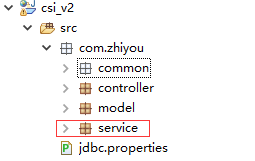
}

# 重构项目

CSI\_v1版本中Servlet负担太重，该版本中servelt除了负责请求转发，还负责了数据的获取和处理工作

重构后让servlet只负责 将模型数据请求转发到视图页面

新增加service层 这一层负责数据库的访问和模型数据的填充



在service包中增加UserService类

## 新增UserService

使用DBUtils工具和我们封装的DBUtil工具类完成对用户信息的增删改查：

**public** **class** UserService {

/\*\*

\* 查询所有

\*

\* **@return**

\*/

**public** List<User> findList() {

**return** DBUtil.*query*("select \* from sys\_user", User.**class**);

}

/\*\*

\* 查询单条记录

\*

\* **@param** id

\* **@return**

\*/

**public** User findOne(Map<String, String[]> param) {

**return** DBUtil.*queryOne*("select \* from sys\_user where username=? and password=?", User.**class**,

param.get("username")[0], param.get("password")[0]);

}

/\*\*

\* 根据id查询

\*

\* **@param** id

\* **@return**

\*/

**public** User getById(Object id) {

**return** DBUtil.*queryOne*("select \* from sys\_user where id=?", User.**class**, id);

}

/\*\*

\* 删除

\*

\* **@param** id

\* **@throws** Exception

\*/

**public** **void** deleteById(Object id) **throws** Exception {

DBUtil.*update*("delete from sys\_user where id=?", id);

}

/\*\*

\* 新增

\*

\* **@param** param

\* **@throws** Exception

\*/

**public** **void** add(Map<String, String[]> param) **throws** Exception {

String sql = "insert into sys\_user(id,username,password,name,email,mobile) values"

+ "(s\_user\_id.nextval,?,?,?,?,?)";

DBUtil.*update*(sql, param.get("username")[0], param.get("password")[0], param.get("name")[0],

param.get("email")[0], param.get("mobile")[0]);

}

/\*\*

\* 更新

\*

\* **@param** param

\* **@throws** Exception

\*/

**public** **void** update(Map<String, String[]> param) **throws** Exception {

String sql = "update sys\_user set email=? ,mobile=? where id=? ";

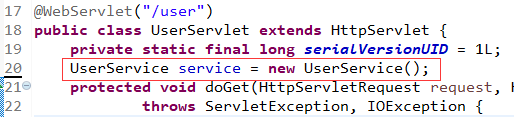
DBUtil.*update*(sql, param.get("email")[0], param.get("mobile")[0], param.get("id")[0]);

}

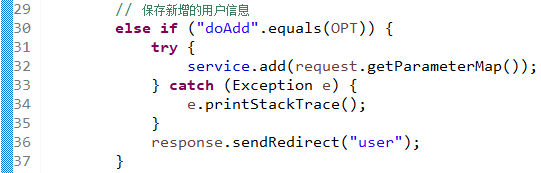
}

## 重构UserServlet

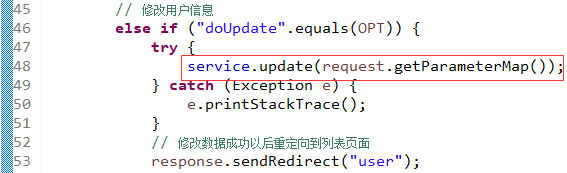
**定义全局UserService变量**



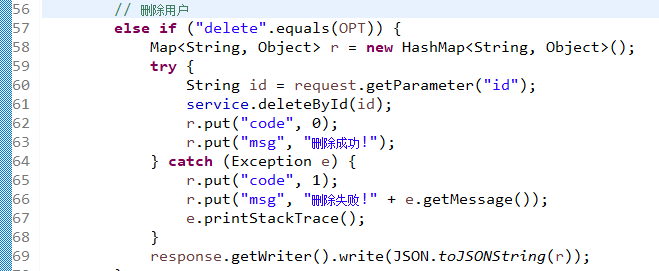
### 新增用户



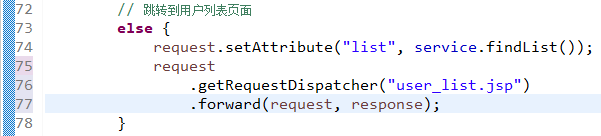
### 修改用户



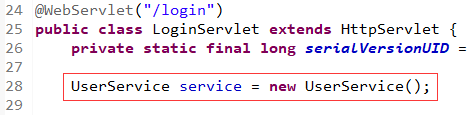
### 删除用户



### 查询用户



## 重构LoginServlet



**完整登录代码**

@Override

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

String username = req.getParameter("username");

String password = req.getParameter("password");

String rememberMe = req.getParameter("rememberMe");

**if** (rememberMe != **null**) {

// 将用户名和密码写入cookie中

Cookie ck1 = **new** Cookie("username", username);

ck1.setMaxAge(60 \* 60 \* 24 \* 365);// 设置cookie的生命周期

resp.addCookie(ck1);// 将cookie信息写入浏览器

Cookie ck2 = **new** Cookie("password", password);

ck2.setMaxAge(60 \* 60 \* 24 \* 365);// 设置cookie的生命周期

resp.addCookie(ck2);// 将cookie信息写入浏览器

}

// 1.校验用户名和密码是否正确

**try** {

Map<String,Object> r = **new** HashMap<String,Object>();

User user = service.findOne(req.getParameterMap());

// 2.如果用户名和密码都正确 跳转到系统主页

**if** (user != **null**) {

req.getSession().setAttribute("sessionUser", user);

r.put("code", 0);

r.put("msg", "登录成功");

}

// 3.如果用户名或密码错误 跳回到登录页面 并且给出提示

**else** {

r.put("code", 1);

r.put("msg", "用户名或密码错误");

}

// map对象转换成json字符串

resp.getWriter().write(JSON.*toJSONString*(r));

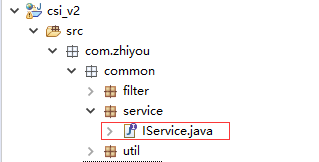
} **catch** (Exception e) {

e.printStackTrace();

}

}

# 抽取IServcie接口



**Iservice**

**public** **interface** IService<T> {

/\*\*

\* 查询所有

\*

\* **@return**

\*/

**public** List<T> findList();

/\*\*

\* 查询单条记录

\*

\* **@param** id

\* **@return**

\*/

**public** T findOne(Map<String, String[]> param);

/\*\*

\* 根据id查询

\*

\* **@param** id

\* **@return**

\*/

**public** T getById(Object id);

/\*\*

\* 删除

\*

\* **@param** id

\* **@throws** Exception

\*/

**public** **void** deleteById(Object id) **throws** Exception;

/\*\*

\* 新增

\*

\* **@param** param

\* **@throws** Exception

\*/

**public** **void** add(Map<String, String[]> param) **throws** Exception;

/\*\*

\* 更新

\*

\* **@param** param

\* **@throws** Exception

\*/

**public** **void** update(Map<String, String[]> param) **throws** Exception;

}

**后续所有的XXXService类都实现Iservcie接口**

