*2018/07/16*

*chenPeng版权所有，未经本人允许不得以任何方式转发或传播*

*违者必究*

# 基本思想

1. 四层架构

为了提高软件开发的效率，从设计模式的角度出发，在三层架构的基础上提把业务逻辑层进一步分出一层，单独形成一个数据接口层，数据接口层屏蔽各种底层数据库之间的差异，负责与底层数据库之间的连接。形成四层软件体系结构的框架。从上到下依次是：表示层、业务逻辑层、数据接口层、数据层。表示层（WEB）是应用软件进行人机交互的接口；业务逻辑层（service）负责处理用户的业务请求；数据接口层（Dao）负责与底层数据库之间的交互；数据层（bean）则负责存储数据。

分层设计的软件系统，由于把相似功能的类或组件放在同一层里。所以好的分层系统应该具备：层的内部“高内聚”，层与层之间是“松耦合”的。

* 1. Bean数据层

相当于一个物体，他有各种属性；实际应用中根据前台需要什么数据定义一个类或是前台提交的[表单](http://www.w3school.com.cn/html/html_forms.asp)前台传回来后台的数据格式定义一个相对于的数据类型

* 1. Dao数据接口层/持久层

与数据库进行交互；数据持久层用于访问数据库，数据持久层中一般是通过DAO（数据访问对象设计模式）访问数据库的，也是为了降低耦合度，DAO被设计为接口，DAO实现可以是JDBC，也可以使用一些ORM框架如Hibernate，mybatis等。

* 1. Service业务逻辑层

服务层是系统的核心业务处理层，负责接收表示层的指令和数据，根据业务逻辑的需要调用相应的持久层，并将结果返回给表示层。为了降低表示层和服务层的耦合问题，我们会在两层之间引入接口，然后在运行期注入依赖关系。

* 1. Web表示层

表示层是用户与系统交互的组件集合，用户通过这一层向系统提交请求或发出指令，系统通过这一层接收用户请求或指令，然后，根据请求指令调用服务层，再根据调用的结果，将相应的内容展现到表示层。表示层应该是轻薄的，不应该具有业务逻辑。如果我们的系统是一个Web系统的话，我们采用的技术有HTML、JSP和Servlet等，也可以使用MVC框架，例如Struts等框架技术。这时我们将表示层称为Web层更准确一点。无论采用什么技术表示层都应该是轻薄的，不应该具有业务逻辑。也就是说无论我们使用JSP、Servlet作为表示层还是使用Swing作为表示层都应该是轻薄的。

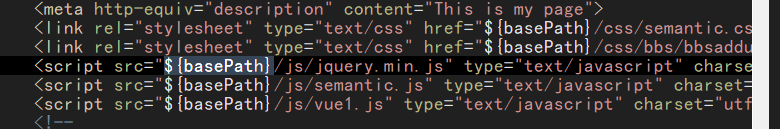
1. SSM（Spring，SpringMVC，Mybatis）简易教程

再此篇建议使用教程中凡是涉及到不做阐述的都可以不去看，等用熟了之后再去了解为什么。文档本着通俗易懂的目的从而文档中有部分说法不严谨敬请读者多多谅解

* 1. 将html页面转换为jsp

当我们写完前台网页之后

将原来的HTML的内容复复制到JSP中去（注意路径问题在我搭好的框架中只需要${basePath}）如图一，原因在此不做阐述。Js，jq，css，img等文件放在webapp（Maven项目）/WebRoot（普通web项目）下。



图一:jsp中的路径问题

* 1. 打通前后台（前台页面到WEB层）

——这一步解释得有点多 只需要做到url进去web层 并且能够syso打印出传入的值就行 多余的东西是在最后一步要用的

在浏览器中可以直接访问webapp（Maven项目）/WebRoot（普通web项目）下除了WEB-INF目录的全部文件

使用SpringMVC中的ModelAndView的setViewName("???/???");可以访问到WEB-INF目录中去（注意：不要将资源文件放入WEB-INF）

此框架中是使用SpringMVC，在SpringMVC中有很多种前后台交互的方法，在此我只举例几种常用的方法。

* + 1. 第一种
* 通过form表单进来，ModelAndView出去（注意高亮部分就OJBK）

Form表单中 action=”addUserDo”就可以进入这个方法。之前提到了要注意路径问题所以action=“${basePath} /addUserDo”

* return andView；代码1中的65行andView.setViewName("bbs/bbsindex"); 的意思是转发到bbs/bbsindex中去；

那么这句话是什么意思呢?

说来话长简单的来说在springMVC配置文件中有这么一段：代码2

其中有 p:prefix="/WEB-INF/pages/" 属性和p:suffix=".jsp"属性

在你andView.setViewName("bbs/bbsindex");然后return andView；之后

SpringMVC会自动拼装组成“/WEB-INF/pages/ bbs/bbsindex.jsp”这样就可以转发到你想去的路径了

显而言之，之所以用[转发而不用重定向](https://blog.csdn.net/meiyalei/article/details/2129120)是因为我们要传值在ModelAndView中还有个方法andView.addObject("???1", ???2);这个方法就是自动的给数据放入四大域中去 名字叫???1 值为???2 这个???2可以传任意一个对象List Map 或者是String Integer等等 只要你想得到的都可以

* 前台中如何取值如果是单独的一个数据就可以用${???1}来直接获取，如果是List简易采用foreach
  + 如果要在前台用foreach则需要使用标准c标签库
  + 第一步：在jsp文件第二行写入

**<**%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %**>**

* + 第二步：在需要的地方写

**<c:forEach** items="${topicList}" var="topic"**>**

**<p>**

**<a** href="${basePath}/bbs/note/${topic.getTid()}"**>**

           ${topic.getTtopic()}

**<samp** style="float: right;"**>**

            ${topic.getDate()}

**</samp>**

**</a>**

**</p>**

**</c:forEach>**

1. /\*\*
2. \* chenPeng
3. \* com.rwybbs.web
4. \* AddUserWeb.java
5. \* 创建人:chenpeng
6. \* 时间：2018年7月13日-下午5:15:38
7. \* 2018陈鹏-版权所有
8. \*/
9. **package** com.rwybbs.web;
11. **import** org.springframework.beans.factory.annotation.Autowired;
12. **import** org.springframework.stereotype.Controller;
13. **import** org.springframework.web.bind.annotation.RequestMapping;
14. **import** org.springframework.web.servlet.ModelAndView;
16. **import** com.rwybbs.bean.BbsUser;
17. **import** com.rwybbs.service.AddUserService;
19. /\*\*
20. \*
21. \* AddUserWeb
22. \* 创建人:chenPeng
23. \* 时间：2018年7月13日-下午5:15:38
24. \* @version 1.0.0
25. \*
26. \*/
27. @Controller
28. **public** **class** AddUserWeb {
30. @Autowired
31. **private** AddUserService addUserService;
33. @RequestMapping("/addUserDo")
34. **public** ModelAndView addUser(
35. Integer uid,
36. String uname,
37. String password,
38. String umailbox,
39. Integer upoint,
40. String uregdate,
41. String uhead,
42. Integer sex,
43. String birth,
44. String satement
45. ){
46. ModelAndView andView = **new** ModelAndView();

49. BbsUser bbsUser = **new** BbsUser();
50. bbsUser.setUid(uid);
51. bbsUser.setUname(uname);
52. bbsUser.setUpassword(password);
53. bbsUser.setUemail(umailbox);
54. bbsUser.setUpoint(upoint);
55. bbsUser.setUregDate(uregdate);
56. bbsUser.setUhead(uhead);
57. bbsUser.setUsex(sex);
58. bbsUser.setUbirthday(birth);
59. bbsUser.setUsatement(satement);
60. bbsUser.setUisSectioner(0);
61. bbsUser.setUstate(1);
63. addUserService.addUser(bbsUser);
65. andView.setViewName("bbs/bbsindex");
66. **return** andView;
67. }


71. }

代码1

1. <!-- 配置图示解释器，将ModelAndView及字符串解析为具体的页面 -->
2. **<bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver"
3. p:viewClass="org.springframework.web.servlet.view.JstlView"
4. p:prefix="/WEB-INF/pages/"
5. p:suffix=".jsp"**/>**

代码2

* + 1. 第二种
* 只转发不从前台得到值（只注意高亮）

1. /\*\*
2. \* chenPeng
3. \* com.rwybbs.web
4. \* BbsIndex.java
5. \* 创建人:chenpeng
6. \* 时间：2018年7月13日-下午11:56:17
7. \* 2018陈鹏-版权所有
8. \*/
9. **package** com.rwybbs.web;
11. **import** java.util.List;
13. **import** org.springframework.beans.factory.annotation.Autowired;
14. **import** org.springframework.stereotype.Controller;
15. **import** org.springframework.web.bind.annotation.RequestMapping;
16. **import** org.springframework.web.servlet.ModelAndView;
18. **import** com.rwybbs.bean.IndexFourm;
19. **import** com.rwybbs.bean.IndexRplay;
20. **import** com.rwybbs.bean.IndexTopic;
21. **import** com.rwybbs.service.IndexService;
23. /\*\*
24. \*
25. \* BbsIndex
26. \* 创建人:chenPeng
27. \* 时间：2018年7月13日-下午11:56:17
28. \* @version 1.0.0
29. \*
30. \*/
31. @Controller
32. **public** **class** IndexWeb {
33. @Autowired
34. **private** IndexService indexService;

37. @RequestMapping("bbs/index")
38. **public** ModelAndView indexInit(){
39. ModelAndView andView = **new** ModelAndView();
40. /\*最新回复\*/
41. List<IndexRplay> rplayList = indexService.getIndexRpaly();
42. /\*热帖\*/
43. List<IndexTopic> topicList = indexService.getIndexTopic();
44. /\*全景业界资讯\*/
45. List<IndexFourm> fourmList1 =
46. indexService.getPanoramicIndustryInformation();
47. /\*全景作品\*/
48. List<IndexFourm> fourmList2 =
49. indexService.getPanoramicWork();
50. /\*生活纪实\*/
51. List<IndexFourm> fourmList3 =
52. indexService.getLifeDocumentary();
53. /\*全景学苑\*/
54. List<IndexFourm> fourmList4 =
55. indexService.getPanoramaAcademy();
56. /\*资源下载\*/
57. List<IndexFourm> fourmList5 =
58. indexService.getDownload();
59. /\*交流分享\*/
60. List<IndexFourm> fourmList6 =
61. indexService.getExchangeSharing();
63. andView.addObject("rplayList", rplayList);
64. andView.addObject("topicList", topicList);
65. andView.addObject("fourmList1", fourmList1);
66. andView.addObject("fourmList2", fourmList2);
67. andView.addObject("fourmList3", fourmList3);
68. andView.addObject("fourmList4", fourmList4);
69. andView.addObject("fourmList5", fourmList5);
70. andView.addObject("fourmList6", fourmList6);
72. andView.setViewName("bbs/bbsindex");
73. **return** andView;
74. }
76. }

代码3

* + 1. 第三种
* 通过url直接传值

1. @RequestMapping("bbs/note/{no}")
2. **public** ModelAndView getNote(@PathVariable Integer no){
3. ModelAndView andView = **new** ModelAndView();
4. /\*增加一次文章浏览量\*/
5. noteService.upDateTClickCoount(no);
6. /\*增加一次板块浏览量\*/
7. noteService.upDateSClickCoount(no);

10. /\*得到主体部分数据\*/
11. Note note = noteService.getNoteMain(no);
12. /\*得到评论部分数据\*/
13. List<NoteRplay> noteRplayList = noteService.getNoteRplay(no);
14. /\*得到热帖\*/
15. List<IndexTopic> topicList = indexService.getIndexTopic();

18. andView.addObject("note", note);
19. andView.addObject("noteRplayList", noteRplayList);
20. andView.addObject("topicList", topicList);
21. andView.setViewName("bbs/bbsnote");
22. **return** andView;
23. }

代码4

* + 1. 第四种
* 通过url传参数的方式传值
  + 这时候 前台的请求为 项目/pay?username=12345

1. @RequestMapping("/pay")
2. **public** ModelAndView intoPay(
3. @RequestParam("username") String username){
4. ModelAndView andView = **new** ModelAndView();
6. List<MoneyUser> user = rwyMoneyService.MoneyService(username);
7. **if** (user==**null**) {
8. andView.setViewName("404");
9. **return** andView;
10. }
11. andView.setViewName("pay");
12. andView.addObject("username", user.get(0));
14. **return** andView;
15. }

代码5

* + 1. 第五种
* Ajax请求 js代码
  + 前台通过点击事件onclick=“函数名（）”随便加在任意的标签里面
  + 在代码5中第2行通过jq取得需要取的值
  + Ajax请求格式固定如3~15行
    - url为请求地址
    - data为数据
    - success为java 中web层返回的值

1. **function** wr(){
2. **var** text = $("#texts").val();
3. $.ajax({
4. type:"post",
5. url:"${basePath}/bbs/note/write",
6. data:{"text":text,"no":${note.getTid()}},
7. success:**function**(data){
8. **if**(data=="success"){
9. location.reload();
10. }**else** **if**(data=="isNull"){
11. alert("评论出错");
12. $("#texts").val("");
13. }
14. }
15. });
16. };

代码6

* Java代码

1. @RequestMapping(method=RequestMethod.POST,value="bbs/note/write")
2. @ResponseBody
3. **public** String setRplay(HttpServletRequest re){
4. String text = re.getParameter("text");
5. String no = re.getParameter("no");
6. String username = (String) re.getSession().getAttribute("username");
7. IsNull isNull = **new** IsNull();
8. **if** (isNull.isNull(text)||
9. isNull.isNull(no)||
10. isNull.isNull(text)) {
11. **return** "isNull";
12. }
14. noteService.setNoteRplay(text, no, username);
15. Return “success” ;
16. }

代码7

自此几种常用的方式介绍完毕

* 1. 测试web

运行tomcat测试看是否进去相对于的web层以及是否得到应该得到的参数

* 1. 测试sql语句

在2.3中测试响应出来的网页对应着在数据库管理软件写sql语句看着是否能得到想要的数据/插入/修改/删除的数据

* 1. 写bean层

在2.4中写的sql语句中 需要增删查改的语句

* + Select中得到的数据列名称写入bean（注意数据格式）
  + Update需要修改的数据以及判断条件的数据列名写入bean
  + Insert需要写入的值的数据列名写入bean
  + Delet传入什么数据列名就写什么如bean
  1. 写dao层

Dao层全部写接口如：代码8

1. /\*\*
2. \* chenPeng
3. \* com.rwybbs.dao
4. \* GetRwyUserDao.java
5. \* 创建人:chenpeng
6. \* 时间：2018年7月13日-下午4:22:54
7. \* 2018陈鹏-版权所有
8. \*/
9. **package** com.rwybbs.dao;
11. **import** com.rwybbs.bean.BbsUser;
12. **import** com.rwybbs.bean.RwyUser;
14. /\*\*
15. \*
16. \* GetRwyUserDao
17. \* 创建人:chenPeng
18. \* 时间：2018年7月13日-下午4:22:54
19. \* @version 1.0.0
20. \*
21. \*/
22. **public** **interface** AddUserDao {
23. /\*\*
24. \* 通过用户名取得users表里面的对应用户数据
25. \* 方法名：getRwyUser
26. \* 创建人：chenPeng
27. \* 时间：2018年7月13日-下午4:25:51
28. \* 手机:17673111810
29. \* @param username
30. \* @return RwyUser
31. \* @exception
32. \* @since  1.0.0
33. \*/
34. **public** RwyUser getRwyUser(String username);
36. /\*\*
37. \* 写入用户
38. \* 方法名：addUser
39. \* 创建人：chenPeng
40. \* 时间：2018年7月13日-下午9:16:40
41. \* 手机:17673111810
42. \* @param user void
43. \* @exception
44. \* @since  1.0.0
45. \*/
46. **public** **void** addUser(BbsUser user);
47. }

代码8

函数说明 ：

public 返回值数据格式 函数名（**传入的数据格式** 传入的数据名）；

* 1. 写sql的xml文件
* 如：代码9
  + 1~3行为固定写法
  + 第4行中namespace的值为对应的dao层（按住ctrl鼠标可以滑过去）
  + 第7行中的id为dao层的函数名
  + 第7行中的parameterType为传入值数据类型（无传入值可不写）
  + 第7行中的resultType为返回出去的值的类型（无返回值课不写）

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
3. "http://mybatis.org/dtd/mybatis-3-mapper.dtd"**>**
4. **<mapper** namespace="com.rwybbs.dao.AddUserDao"**>**
6. <!-- 原来的取值 -->
7. **<select** id="getRwyUser"  parameterType="String"  resultType="com.rwybbs.bean.RwyUser"**>**
8. SELECT
9. uid,
10. uname,
11. upassword,
12. umailbox,
13. phone\_number,
14. takegold,
15. createdate,
16. upicture
17. FROM
18. users
19. WHERE
20. uname = #{username}
21. **</select>**

24. <!-- 第一次进博客录入 -->
25. **<insert** id="addUser" parameterType="com.rwybbs.bean.BbsUser" flushCache="true"**>**
26. INSERT INTO
27. bbsuser (
28. UID,
29. UName,
30. UPassword,
31. UEmail,
32. UBirthady,
33. USex,
34. UHead,
35. USatement,
36. URegDate,
37. UState,
38. UPoint,
39. UIsSectioner
40. )
41. VALUES(
42. #{uid },
43. #{uname },
44. #{ upassword},
45. #{uemail },
46. #{ubirthday },
47. #{usex },
48. #{ uhead},
49. #{ usatement},
50. #{ uregDate},
51. #{ ustate},
52. #{ upoint},
53. #{ uisSectioner}
54. );
55. **</insert>**
57. **</mapper>**

代码：9

自此数据库层与java交互完毕

* 1. 写service层
* 如：代码10
  + 第27行Public class ????前面要加上注解信息 @Service 让spring可以扫到
  + 第29~30行

@Autowired

private AddUserDao addUserDao;

为自动注入 注入的就是之前写好的dao

这个东西约等于AddUserDao addUserDao = new AddUserDao（）；

为什么这么做 在此不做阐述 详情可以参考spring 的ioc与aop

1. /\*\*
2. \* chenPeng
3. \* com.rwybbs.service
4. \* AddUserService.java
5. \* 创建人:chenpeng
6. \* 时间：2018年7月13日-下午4:34:03
7. \* 2018陈鹏-版权所有
8. \*/
9. **package** com.rwybbs.service;
11. **import** org.springframework.beans.factory.annotation.Autowired;
12. **import** org.springframework.stereotype.Service;
14. **import** com.rwybbs.bean.BbsUser;
15. **import** com.rwybbs.bean.RwyUser;
16. **import** com.rwybbs.core.Time;
17. **import** com.rwybbs.dao.AddUserDao;
19. /\*\*
20. \*
21. \* AddUserService
22. \* 创建人:chenPeng
23. \* 时间：2018年7月13日-下午4:34:03
24. \* @version 1.0.0
25. \*
26. \*/
27. @Service
28. **public** **class** AddUserService {
29. @Autowired
30. **private** AddUserDao addUserDao;
32. /\*\*
33. \* 拿到rwy表里面的数据
34. \* 方法名：getRwyUser
35. \* 创建人：chenPeng
36. \* 时间：2018年7月13日-下午4:41:27
37. \* 手机:17673111810
38. \* @param username
39. \* @return RwyUser
40. \* @exception
41. \* @since  1.0.0
42. \*/
43. **public** RwyUser getRwyUser(String username){
44. **return** addUserDao.getRwyUser(username);
45. }
47. /\*\*
48. \* 写入
49. \* 方法名：addUser
50. \* 创建人：chenPeng
51. \* 时间：2018年7月13日-下午9:44:24
52. \* 手机:17673111810
53. \* @param user void
54. \* @exception
55. \* @since  1.0.0
56. \*/
57. **public** **void** addUser(BbsUser user){
58. Time time = **new** Time();
59. user.setUbirthday(time.strToDate(user.getUbirthday()));
60. user.setUregDate(time.cstToStr(user.getUregDate()));
61. addUserDao.addUser(user);
62. }
63. }

代码10

* 1. 回到web层
* 注入service 通过service调用dao中的接口来得到数据库中的数据然后传入四大域中
* 如:代码11
  + 28行public class ??? 前面加上 @Controller让spring扫到
  + 33~34行中

@Autowired

private AddUserService addUserService;

自动注入AddUserService

这个东西约等于AddUserService addUserService = new AddUserService（）；

* + addUserService.addUser(bbsUser);  通过service中的AddUserService来使用addUser方法传入bbsUser数据

1. /\*\*
2. \* chenPeng
3. \* com.rwybbs.web
4. \* AddUserWeb.java
5. \* 创建人:chenpeng
6. \* 时间：2018年7月13日-下午5:15:38
7. \* 2018陈鹏-版权所有
8. \*/
9. **package** com.rwybbs.web;
11. **import** org.springframework.beans.factory.annotation.Autowired;
12. **import** org.springframework.stereotype.Controller;
13. **import** org.springframework.web.bind.annotation.RequestMapping;
14. **import** org.springframework.web.servlet.ModelAndView;
16. **import** com.rwybbs.bean.BbsUser;
17. **import** com.rwybbs.service.AddUserService;
19. /\*\*
20. \*
21. \* AddUserWeb
22. \* 创建人:chenPeng
23. \* 时间：2018年7月13日-下午5:15:38
24. \* @version 1.0.0
25. \*
26. \*/
27. @Controller
28. **public** **class** AddUserWeb {
30. @Autowired
31. **private** AddUserService addUserService;
33. @RequestMapping("/addUserDo")
34. **public** ModelAndView addUser(
35. Integer uid,
36. String uname,
37. String password,
38. String umailbox,
39. Integer upoint,
40. String uregdate,
41. String uhead,
42. Integer sex,
43. String birth,
44. String satement
45. ){
46. ModelAndView andView = **new** ModelAndView();

49. BbsUser bbsUser = **new** BbsUser();
50. bbsUser.setUid(uid);
51. bbsUser.setUname(uname);
52. bbsUser.setUpassword(password);
53. bbsUser.setUemail(umailbox);
54. bbsUser.setUpoint(upoint);
55. bbsUser.setUregDate(uregdate);
56. bbsUser.setUhead(uhead);
57. bbsUser.setUsex(sex);
58. bbsUser.setUbirthday(birth);
59. bbsUser.setUsatement(satement);
60. bbsUser.setUisSectioner(0);
61. bbsUser.setUstate(1);
63. addUserService.addUser(bbsUser);
65. andView.setViewName("bbs/bbsindex");
66. **return** andView;
67. }


71. }

代码11

* 最后参考第二步中如何在前台获取到四大域中的值自此一个业务流程走完

注意边写边测试不要到最后了错都不知道错在什么地方