# 第一阶段 注册页面的验证

# 第二阶段 完成登录和注册的功能

## 软件架构

饭店的架构

服务员 厨师 采购

软件的三层架构

表示层（表现层、web层）

* 用户可以直接访问到的内容
* HTML页面 Servlet

业务逻辑层

* 业务逻辑层负责处理网站的一些业务逻辑（登录、注册、买书、结账）
* Service层

持久化层

* 负责操作数据库做增删改查等基本操作
* DAO层

数据库

持久化层

业务逻辑层

表示层

分层的目的就是解耦！！！

由于业务逻辑层和持久化层在开发过程中非常容易发生改变的，所以这两层我们采用一个面向接口的编程方式。

## 项目结构

## 开发步骤

1. 创建一个新的动态web工程。
2. 创建数据库bookstore\_0816
3. 创建表bs\_user
4. 创建类User
   1. Integer id
   2. String username
   3. String password
   4. String email
5. 导入第三方的jar包
   1. c3p0-0.9.1.2.jar
   2. commons-dbutils-1.3.jar
   3. mysql-connector-java-5.1.7-bin.jar
6. 导入c3p0配置文件
7. 创建JDBCUtils工具类
   1. Connection getConnection()
   2. void releaseConnection(Connection conn)
8. 创建BaseDao<T>
   1. int update()
   2. List<T> getBeanList()
   3. T getbBean()
9. 创建UserDao接口及实现类
   1. User getUserByUsernameAndPassword(User user)
   2. int saveUser(User user)
10. 创建UserService接口及实现类
    1. User login(User user)
    2. boolean regist(User user)
11. 创建LoginServlet和RegistServlet
12. 导入已经写好的页面
    1. 在页面中加入base标签
    2. 修改页面中的路径
    3. 修改表单的method和action属性

# 第三阶段 项目优化（修改html为jsp）

## 将所有的html修改为jsp

* 1. 新建一个jsp文件，并将html页面复制进新页面（主要page指令要保留）
  2. 在html页面中加入一个page指令，然后修改文件扩展名为jsp
  3. 提取出页面中重复的部分，在通过include指令将这些内容分别引入网页

## 登录和注册页面的错误消息显示

* 登录或注册失败以后再页面中显示错误消息
* 登录
  + 登录失败时也就是用户名或密码输错了，我们需要在域中设置一个错误消息然后在页面显示出来。

## 表单的回显

* 登录或注册失败以后，在表单中显示用户上次输入的信息

## 项目优化

* 问题1：目前我们的项目中是一个功能对应一个Servlet，登录有LoginServlet、注册有RegistServlet、删除用户有DelUserServlet、修改用户有UpdateUserServlet、添加图书有AddBookServlet。。。这样会导致我们的项目被淹没在Servlet海洋中。
* 问题1思路：一个Serlvet可以处理多个请求，比如有一个UserServlet那么用户相关的请求都可交给该Servlet，比如登录对应一个login方法，注册对应一个regist方法。
  + 步骤：
    - 1.先创建一个UserServlet
    - 2.在Servlet中创建两个方法，一个是login 一个是regist
    - 3.当用户要登录时调用login 注册时调用regist
    - 4.我们希望用户可以告诉Servlet要掉login还是调regist
    - 5.我们通过用户传递method属性来判断要调用的方法。
  + 新问题：
    - 在这个UserServlet中每添加一个方法，就要添加一个与之对应if else语句，这样会导致if else很多。
  + 解决：
    - 如果可以直接根据方法名，获取到方法，我们就可以直接去调用方法，而不需要再手动调用
    - 通过反射动态的获取用户要调用的方法对象，然后在去调用这个方法，这样做的好处是，当在添加新的方法进Servlet时，不需要写if else，用户只需传递相应的方法名即可调用方法。
    - 注意：我们所定义的方法要和doGet和doPost方法结构一样，可以通过重写doGet方法然后改名的形式创建方法。
  + 新问题：
    - 我们将反射这一系列代码编写到了当前Servlet，那么我们每新创建一个Servlet这一堆代码就得在写一遍，我们的代码不能复用。
    - 解决：
      * 创建一个BaseServlet，将这些重复的代码在BaseServlet中编写，其他Servlet只需继承BaseServlet而不用再次编写重复的代码。
  + 注意：
    - 当我们使用get请求时，get请求会覆盖action中的请求参数
    - 当使用get请求时，可以在表单中设置一个表单隐藏域，这个域的name属性是method，值是要调用的方法

|  |
| --- |
| <input type=*"hidden"* name=*"method"* value=*"login"* /> |

* 问题2：在大部分的Servlet中都需要先获取用户发送的请求参数，然后将这些请求参数封装为对象，像这些操作都是一些比较简单，但是又比较繁琐（尤其是当我们需要做转型操作）
* 问题2解决：写一个通用工具类来替咱们做这件事（BeanUtils）。
* JavaBean
  + 属性私有化，通过getter和setter方法对属性进行读取和设置
  + 有一个无参的构造器
  + JavaBean的属性指的是getter方法中属性名，属性名指的getter或setter方法将get或set去掉，然后首字母小写。getName()属性名叫name，如果有一个getAge()属性名叫age。
* 使用BeanUtils可以直接将一个Map转换一个JavaBean的对象。
* 我们想可不不可以获取到请求参数的map，然后将这个map直接转换为我们想要的对象。
* 我们通过BeanUtils的populate方法将请求参数的map直接封装为对象，但是这块有一个问题需要注意表单项中name属性要和类中的属性名一样，否则将不能进行封装。

# 第四阶段 使用EL替换所有jsp表达式

* login.jsp
* regist.jsp
* base.jsp

# 第五阶段 图书的增删改查

## 第一部分：后台图书的增删改查

### 创建图书的类Book和图书的表bs\_book



Interger id

书名：String title

作者：String author

价格：double price

销量：int sales

库存：int stock

封面：String imgPath

### 创建BookDao接口及实现类

int saveBook(Book book) 🡪 向数据库中添加一本图书

int delBook(String bookId) 🡪 根据ID从数据库删除一本图书

int updateBook(Book book) 🡪 修改一本图书

List<Book> getBookList() 🡪 获取所有的图书

Book getBookById(String bookId) 🡪 根据id获取一本图书

### 创建BookService接口及实现类

int saveBook(Book book) 🡪 向数据库中添加一本图书

int delBook(String bookId) 🡪 根据ID从数据库删除一本图书

int updateBook(Book book) 🡪 修改一本图书

List<Book> getBookList() 🡪 获取所有的图书

Book getBookById(String bookId) 🡪 根据id获取一本图书

### 创建BookManagerServlet

addBook() 🡪 添加图书

delBook() 🡪 删除图书

updateBook() 🡪 修改图书

* 修改图书，首先需要将要修改的图书的信息在表单中回显，所以在修改之前需要先做查询操作，查询到要修改的图书，并在页面中进行回显。

bookList() 🡪 获取所有的图书的列表

getBookById() 🡪 获取一本的图书信息 \*\*\*

### 问题

* 目前我们使用SELECT \* FROM bs\_book 一下子将数据库中图书全部查询出来，这样如果数据库的图书较多的话，会使得性能变的非常非常的差。同时也会使得用户体验变得非常差。所以像上边那样的SQL我们在开发能不写就不写。
* 解决方案：
  + 我们可以不一次将所有的数据全都查出来，可以一次查询4条数据。
  + 这里我们就要为图书的显示做一个分页操作。
* 我们需要使用如下SQL语句，进行分页查询
  + SELECT \* FROM bs\_book LIMIT 开始的索引 , 每页显示的条数
  + 由于在页面加入了分页显示，所以之前我们使用List已经不能满足我们的需求了，这里我们就需要来创建一个新的类，来封装分页信息
  + Page



int pageSize

int totalRecord

int totalPage

int pageNumber

List<T> data

int index

* + Page<T>类
    - int pageNumber //当前页码 ， 该数据在Servlet中获取
    - int pageSize //每页显示的条数 ， 在Servlet中指定
    - int index // 分页开始的索引 ， 通过计算获得
    - int totalPage // 总页数 ， 需要通过计算获得
    - int totalRecord // 总记录 ， 通过查询数据库获得
    - List<T> data; // 分页的数据 ， 通过数据库查询
* 分页查询时，DAO、Service的分工
  + Service
    - pageNumber
    - pageSize
  + DAO
    - int totalRecord
    - List<T> data;
* 在BookDao中添加一个新的方法
  + Page<Book> findBook(Page<Book> page)
* 在BookService中添加一个新的方法
  + Page<Book> findBook(String pageNumber , int pageSize)
* 在BookManagerServlet中添加一个新的方法
  + findBook()

## 第二部分：前台页面图书的显示