

**自主学习项目详细设计**

**V1.0**

**广西民族大学**

评审日期： 2016年2月28日

目录

[1．导言 3](#_Toc75462518)

[1.1 目的 3](#_Toc75462519)

[1.2 范围 3](#_Toc75462520)

[1.5 版本更新信息 3](#_Toc75462521)

[2．详细设计简述 4](#_Toc75462522)

[2.1 设计简介 4](#_Toc75462523)

[3、界面详细设计 4](#_Toc75462524)

[3.1、页面设计说明 4](#_Toc75462525)

[3.2、静态页面详细设计 4](#_Toc75462526)

[4、客户端模块详细设计 5](#_Toc75462527)

[4.1、公用模块的详细设计 5](#_Toc75462528)

[4.1.1 表现层 5](#_Toc75462529)

[4.1.2 控制层 5](#_Toc75462530)

[4.1.3 数据持久层 6](#_Toc75462531)

[4.2、登录模块 6](#_Toc75462532)

[4.2.1 表现层 7](#_Toc75462533)

[4.2.2 控制层 8](#_Toc75462534)

[4.2.3 业务逻辑层 9](#_Toc75462535)

[4.2.4 数据持久层 10](#_Toc75462536)

[4.2.5 域模型层 12](#_Toc75462537)

# 1．导言

## 1.1 目的

该文档是对课程资料自主学习系统概要设计进行的详细说明，是为了指导和规范课程资料自主学习系统开发而制定的详细开发设计文档。在概要设计说明书中，已经解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、模块结构的设计等。在以下的详细设计报告中将对在本阶段中对系统所做的所有详细设计进行说明。

本文档的预期读者包括：

* 设计开发人员
* 项目管理人员
* 测试人员
* 用户

## 1.2 范围

该文档内容涵盖课程资料自主学习系统系统的所有功能模块的详细设计描述。

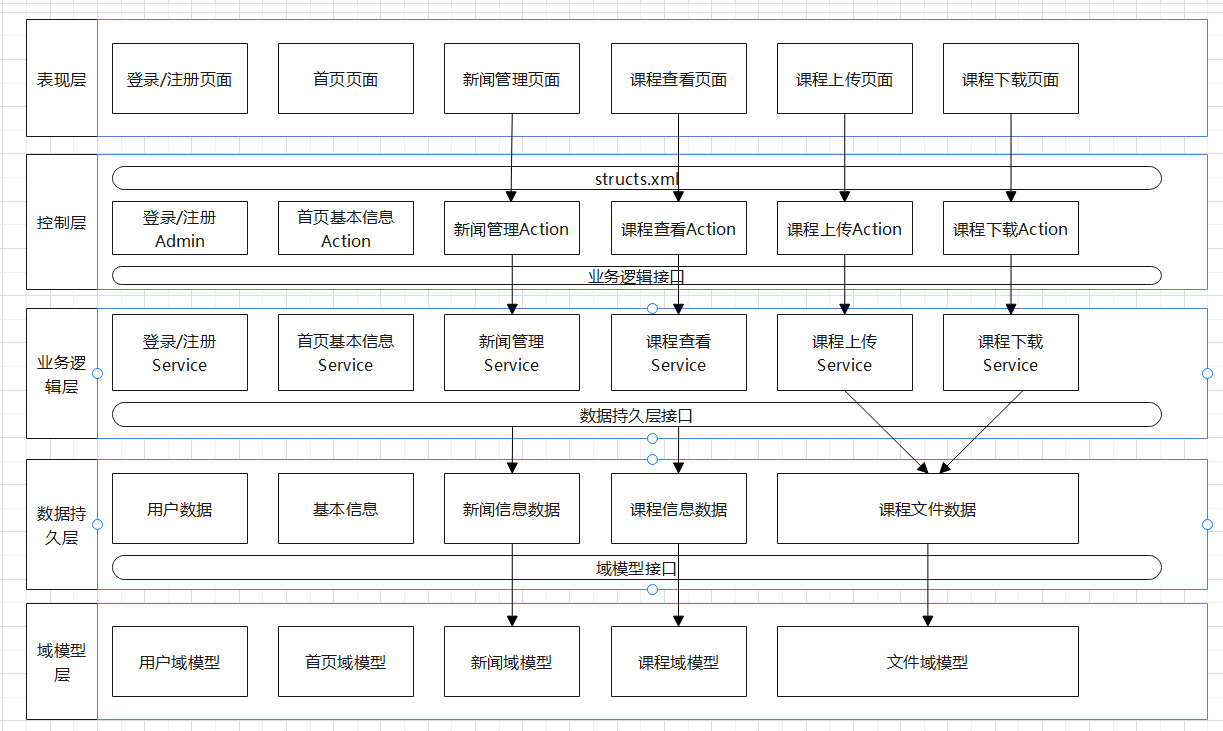
## 1.5 版本更新信息

表1-1 文档更新记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 修改编号 | 修改日期 | 修改后版本 | 修改位置 | 修改内容概述 |
| 001 | 2021.6.24 | 1.0 | 全部 | 初始发布版本 |

# 2．详细设计简述

## 2.1 设计简介



# 3、界面详细设计

[界面详细设计 包括界面设计的技术，以及静态页面设计]

## 3.1、页面设计说明

清新简介，分为用户名与登录密码两个选项。

## 3.2、静态页面详细设计

表3-1 页面表现层设计

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录界面 | Login.jsp | 实现用户的登录功能 |
| 主页界面 | Main.jsp | 略 |

# 4、客户端模块详细设计

客户端模块主要包括登录模块，下载模块，上传模块，基本上包括了所有内容。

## 4.1、公用模块的详细设计

[公用模块是指在系统在运行过程中将会被其他模块调用的基础资源]。

### 4.1.1 表现层

公用模块的表现层主要是一些页面出错时的跳转页面，显示当前错误的结果处理报告。具体见表？-？。

表 4-1 。。。。。。

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录出错页面 | loginError.jsp | 登入时出现严重错误时跳转到该页面。 |
| 统一异常页面 | exceptionPage.jsp | 在系统运行时出现未知错误时跳转到该页面。 |

### 4.1.2 控制层

**public** class ActionSupport implements Action, Validateable, ValidationAware, TextProvider, LocaleProvider, Serializable {

**protected** static Logger LOG = LoggerFactory.getLogger(ActionSupport.class);

**private** **final** ValidationAwareSupport validationAware = new ValidationAwareSupport()；

**private** transient TextProvider textProvider;

**private** Container container;

**public** **void** setActionErrors(Collection<String> errorMessages) {  }

**public** Collection<**String**> getActionErrors() {  }

**public** void setActionMessages(Collection<String> messages) {  }

**public** Collection<**String**> getActionMessages() {  }

}

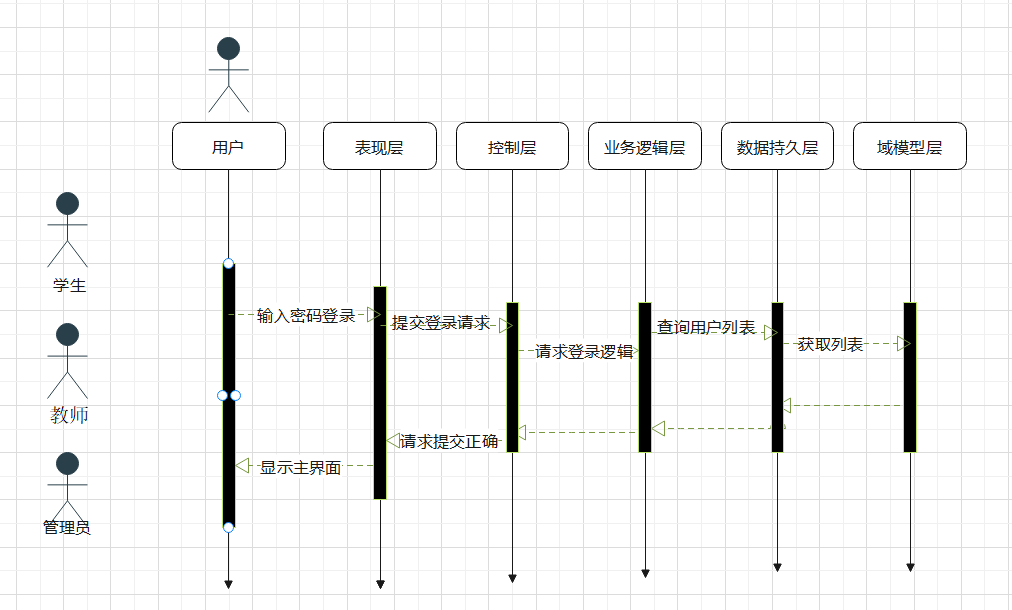
### 4.1.3 数据持久层

BaseDAOImpl.java对UserDAO.java实现：

public class UserDao {  
public User login(User user) throws SQLException {  
QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());  
User existUser = queryRunner.query("select \* from user where username = ? and password = ?",  
new BeanHandler<User>(User.class), user.getUsername(), user.getPassword());  
  
return existUser;  
}  
public Student getStu(String number) throws SQLException {  
QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());  
Student existStu = queryRunner.query("select \* from student where number = ? ",  
new BeanHandler<Student>(Student.class),number);   
return existStu;  
}  
}

## 4.2、登录模块

在登录模块时，系统内部的相应响应操作示意图如图4-2所示。

图 4-2 系统响应示意图

### 4.2.1 表现层

登录模块的表现层主要完成不同用户的登录功能，登录页面时要求用户输入账号密码的基本信息，确认后页面给出响应消息，提示登录成功或失败的提示。表现层对应的JSP页面列表见表?-?所示。

表 4-3 登入模块表现层JSP列表

|  |  |  |
| --- | --- | --- |
| 界面 | JSP | 功能描述 |
| 登录页面 | login.jsp | 用户（教师、学生）登入功能，当登入出错时给出提示。 |

login.jsp的流程图如图4-3所示。

进入游首页

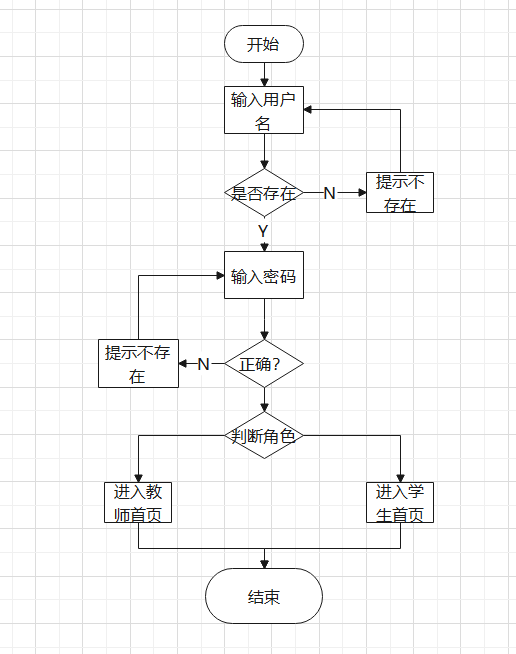


图 4-3 登录模块详细设计流程图

### 4.2.2 控制层

登录模块的控制层负责接受来自login.jsp的用户输入，同时调用登入模块的业务逻辑接口，将用户名与密码等用户关键信息传递到业务逻辑层进行判定。等到业务逻辑处理完成之后，将来自业务逻辑层的相应信息传到表现层，并决定显示页面。登入模块控制层列表见表4-4所示。

表 4-4 登入模块控制层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Action | 转移说明 | 出口 |
| 登录 | LoginAction.java | SUCCESS | mainFrame.jsp—登入成功，显示提示窗口。 |
| ERROR | login.jsp---登入失败，显示提示窗口。 |

在控制层中LoginAction.java的描述如下所示：

**public** **class** LoginAction **extends** ActionSupport

{

**private** User user;

*@Resource*

**private** IUserService userService;

**public** String login(){}

}

### 4.2.3 业务逻辑层

登录模块的业务逻辑层主要完成对用户登录逻辑的判定，同时调用登录模块的业务逻辑接口。比如：用户到登录是输入的用户名是否存在、密码是否正确，同时在对用户的身份进行判定。登录模块业务逻辑层列表如图4-5所示。

表 4-5登入模块业务逻辑层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | Service | 调用说明 | 出口 |
| 登录 | IUserService.java  UserServiceImpl.java | 调用IUserDao | 返回给Action |

在登入模块的业务逻辑层是调用了公用的IUserService接口，同时在实现该接口。

IUserService.java接口主要方法：

**public** **interface** IUserService {

**public** User login(String userName,String password);

**public** User findUser(String userName);

**public** boolean deleteUser(String username);

**public** boolean addUser(User user);

}

UserServiceImpl.java实现了IUserService.java接口， 同时需要调用数据持久层的UserDao.Java，利用@Resource来实现对数据持久层接口的调用。

UserServiceImpl.java主要实现属性与方法:

**public** **class** UserServiceImpl **implements** IUserService {

*@Resource*

**private** IUserDao iUserDao;

*@Override*

**public** User login(String userName,String password);

*@Override*

**public** User findUser(String userName);

*@Override*

**public** boolean deleteUser(String username);

*@Override*

**public** boolean addUser(User user);

}

### 4.2.4 数据持久层

登录模块的是数据持久层能对用户数据进行增删改查。登入模块数据持久层列表见表4-6所示。

表 4-6 登入模块数据持久层列表

|  |  |  |  |
| --- | --- | --- | --- |
| 事件 | DAO | 调用数据模型 | 说明 |
| 登录 | IUserDao.java  UserDaoImpl.java | 调用 user.Java | 对用户信息进行增删改查操作 |

IUserDao.java定义了用户信息进行增、删、改、查的接口：

**public** **interface** IUserDao {

**public** User findUser(User user);

**public** boolean insertUser(User user);

**public** List<User> searchUser(List<Object> choose);

**public** boolean deleteUser(String username);

**public** boolean addUser(User user);

}

UserDaoImpl.java是IUserDao.java接口的实现：

@Repository

**public** **class** UserDaoImpl extends BaseDAOImpl implements IUserDao {

**public** User findUser(User user);

**public** boolean insertUser(User user);

**public** List<User> searchUser(List<Object> choose);

**public** boolean deleteUser(String username);

**public** boolean addUser(User user);

}

### 4.2.5 域模型层

登录模块用到域模型层中User.java，User.java是一个公用域模型，在涉及到用户信息查询等操作时，就会调用到该模型，登录模块域模型层列表如表4-7所示。

表 4-7登入模块域模型层列表

|  |  |
| --- | --- |
| 域模型 | 描述 |
| User.java | 对用户信息的增、删、改、查操作 |

User.java主要属性与方法：

**public** **class** User **implements** Serializable{

**private** **String** id;

**private** **String** userName;

**private** **String** password;

**private** **String** password1;

**private** **String** position;

**private** **String** userId;

**private** **int** videoTime;

}