**Web技术与应用课程作业（论文）**

**基于Web技术的简单班级管理系统**

学 院（系）： 软件学院

专 业： 网络工程

学 生 姓 名： 杨晨

指 导 教 师： 卢炳先

评 阅 教 师： 卢炳先

完 成 日 期： 2022.1.3

大连理工大学

Dalian University of Technology

# 摘 要

随着社会的发展，信息与数据量激增，互联网普及后高效便捷的优势使得班级管理系统的开发成为必需。班级管理系统主要是借助计算机，通过对首页、班级综合数据、综合评测数据等信息进行管理。减少管理员的工作，同时也方便广大用户对个人所需班级综合评测数据的及时查询以及管理。

数字化校园建设是各大高校网络信息化建设部门重点实施的项目，它从人、财、物等方面全面提高高校的教学、科研和管理的水平。狭义上讲，数字化校园建设是指以数字化信息和网络为基础，实现高校里面的各项资源、管理和服务最大程度的数字化，并提供信息化的手段对这些数字化资源进行高效的使用。数字化校园建设形成了数字化的科研、教学和生活环境，为广大师生提供简便、快捷的网络化信息服务。

基于以上原因，本文客户端代码采用html+css+jquery+ajax方式，服务器端采用的是pycharm中的flask框架，核心代码为python,还涉及到数据库的存储过程脚本，开发一款简单班级管理系统。采用软件工程的开发流程，增强网页开发的可维护性，总体上可以给用户带来较良好的体验。

目 录

[摘 要 1](#_Toc123921015)

[1 绪论 1](#_Toc123921016)

[1.1 选题和技术背景 1](#_Toc123921017)

[2 需求分析和设计 1](#_Toc123921018)

[2.1 需求分析 1](#_Toc123921019)

[2.2 软件设计概览 2](#_Toc123921020)

[2.2.1 概要设计和详细设计 2](#_Toc123921021)

[2.2.2 网页界面设计 3](#_Toc123921022)

[3 编码实现 6](#_Toc123921023)

[3.1 客户端实现 6](#_Toc123921024)

[3.2 服务端实现 9](#_Toc123921025)

[3.3 数据库设计 11](#_Toc123921026)

[学习心得 13](#_Toc123921027)

# 1 绪论

## 选题和技术背景

对教育进行改革、多样性、质量等等的要求，使教育系统的管理和运营比过去十年前更加理性化。依照这一现实为基础，设计一个快捷而又方便的网上班级管理系统是一项十分重要并且有价值的事情。对于传统的班级管理系统控制模型来说，班级管理系统具有许多不可比拟的优势，首先是快速更新班级管理系统的信息，其次是大量信息的管理，最后是高度安全，以及使用简单等特性，这使得班级管理系统的管理和运营非常方便。进入21世纪，因为科技和经济的迅速发展，本系统是为了实现这些目标而提出来的。

在本门课程中学习了基础的Web开发知识，包括Django以及HTML、CSS、JS，等Web技术。此外，jQuery 是一个 JavaScript 库，极大地简化了 JavaScript 编程。异步的 JavaScript 和 XML最大的优点是在不重新加载整个页面的情况下，可以与服务器交换数据并更新部分网页内容，不需要任何浏览器插件，但需要用户允许 JavaScript 在浏览器上执行。Flask相对于Django而言是轻量级的Web框架。和Django不同，Flask轻巧、简洁，通过定制第三方扩展来实现具体功能。可定制性，通过扩展增加其功能，这是Flask最重要的特点。故本文客户端代码采用的是html+css+jquery+ajax方式，服务器端采用的是pycharm中的flask框架，数据库的操作基本为增删查改，实现简单班级管理系统

# 2 需求分析和设计

## 2.1 需求分析

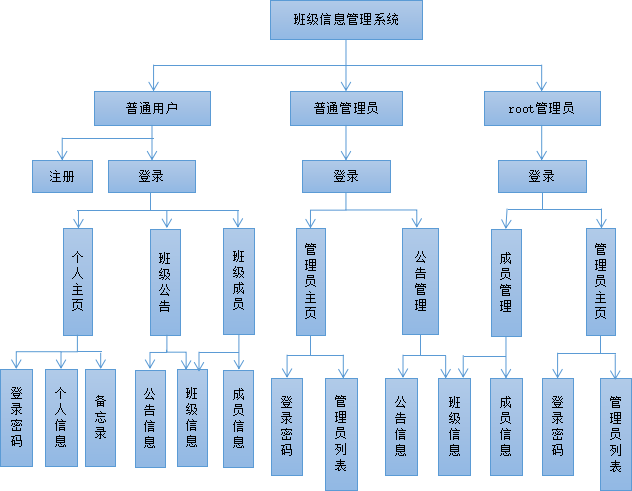
班级管理系统需求概览：

1. 用户可以注册和登录该网站（账户合法性检测）
2. 用户可以获取个人信息和相应的班级信息和成员信息
3. 管理员可以登录该网站（账户合法性检测）
4. 管理员可以获取班级信息和成员信息
5. root权限管理员可以新建普通权限管理员

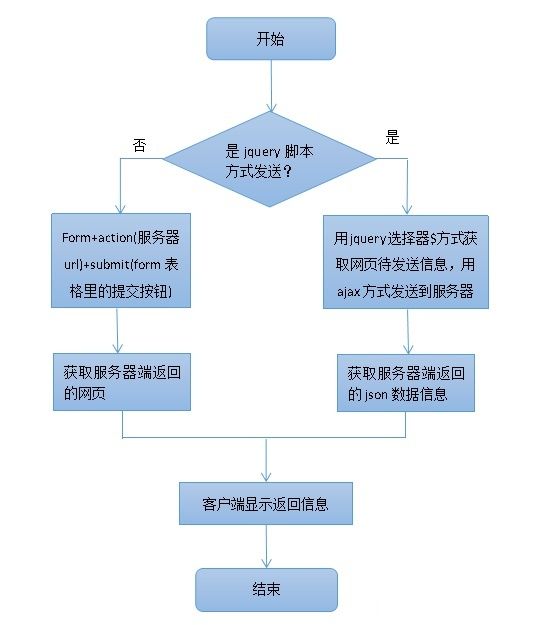
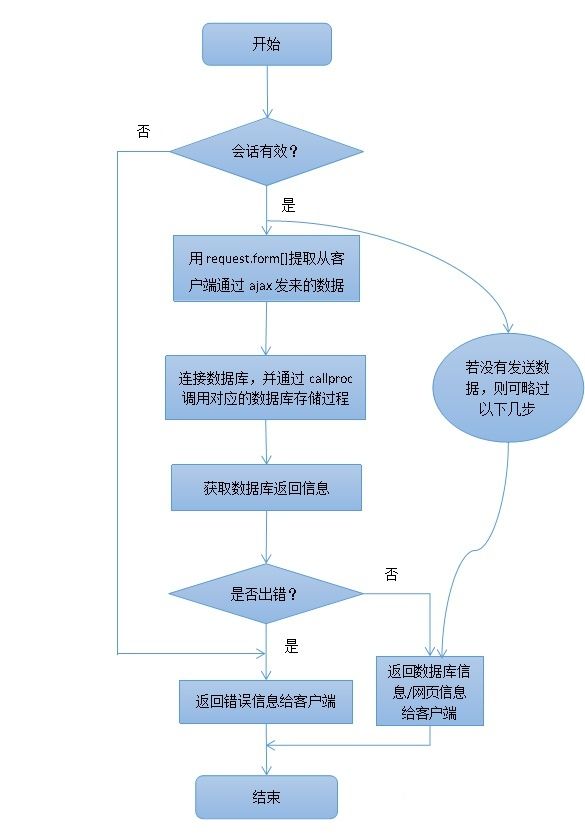
## 2.2 软件设计概览

### 2.2.1 概要设计和详细设计

概要设计：



详细设计(客户与服务端)：



详细设计(数据库设计)：

下面将详细描述本系统的关系实体，用到的所有关系模式如下所示：

1.用户（用户id，账户，密码，姓名，性别，居住地，电子邮箱）

2.管理员（管理员id，账户，权限等级，密码）

3.备忘录（备忘录id，标题，内容，日期）

4.公告（公告id，标题，内容，日期）

5.班级（班级id，名称）

6.班级成员（班级id,用户id）

### 2.2.2 网页界面设计

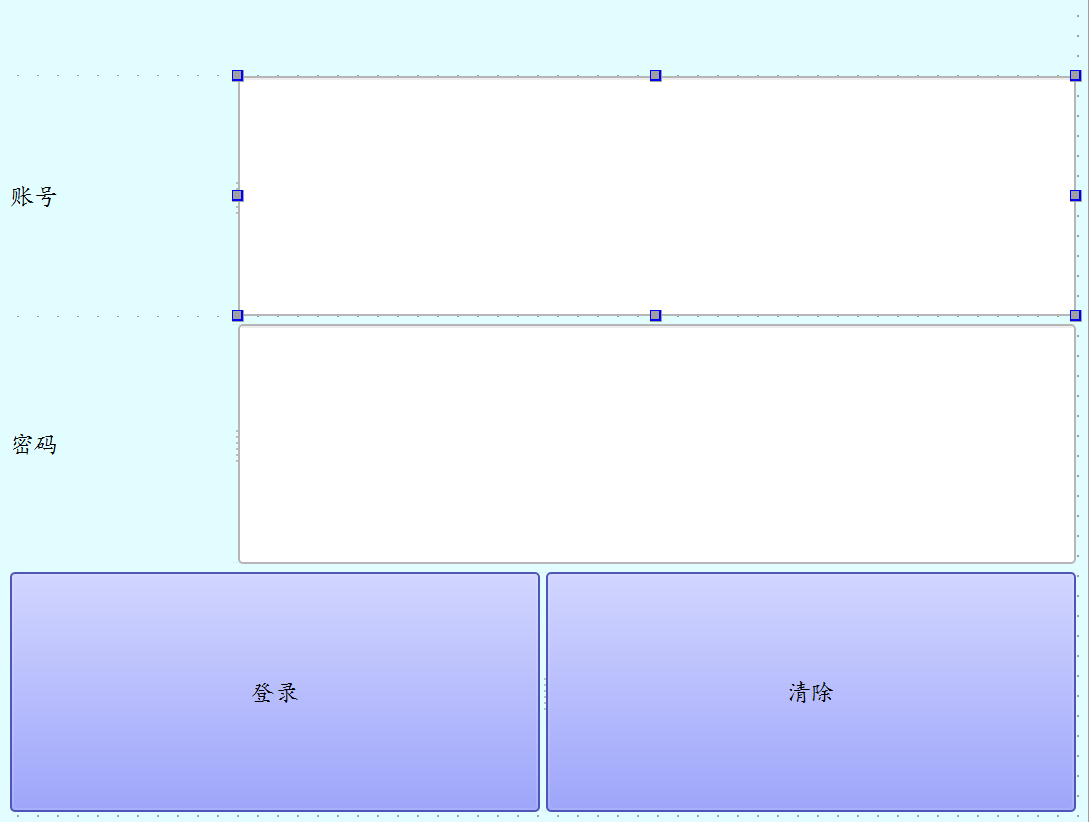
在完成了数据库后端设计后，如何设计UI界面，让系统使用者能够友好快速的使用该系统就成为了当前急需解决的问题。良好的UI界面不仅能增加用户对该系统的好感，也能够将系统的功能最大限度的展现在使用者面前。因此，接下来将从输入界面设计和输出界面设计两方面开始阐述该系统的输入输出界面设计。

系统的输入主要是靠html的表格和模态框来完成的，并且输入输出界面有很大重合性（获得输入结果后又输出显示在当前界面上(通过对话框或是网页列表等)），典型的输入界面包括如下几个：

1.注册登录页面

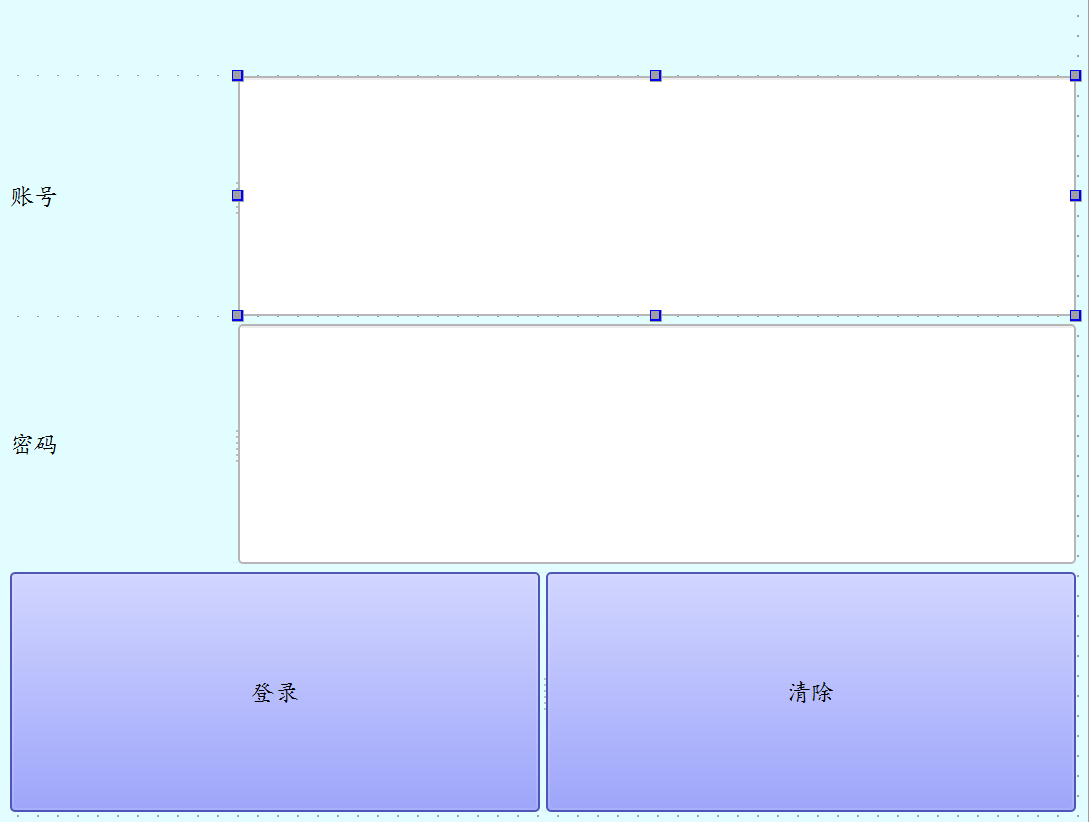
【用户注册登录页面】

用户使用系统前要完成注册和登录功能，因此，单独列出注册登录页面会比较方便，该页面原型如下：



【管理员登录页面】

管理员只需要登录即可使用该系统，新管理员的产生是通过root权限管理员授权而得到的。因此采用对话框方式实现，具体如下所示：



【其余信息输入界面】

包括密码修改界面新密码的输入输出均采用对话框方式，类似于管理员登录页面，在此不做赘述。



输出界面和输入界面都是在一个html文件里，区别在于：动态输出页面主要是通过改变html中的无序列表来设计的。

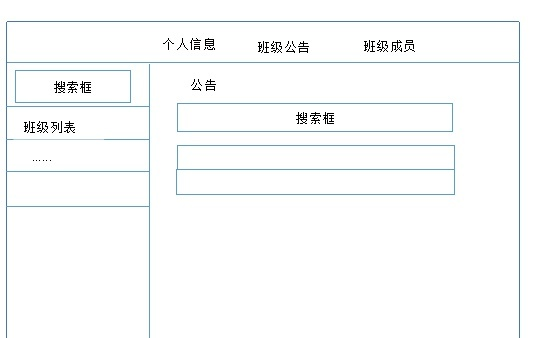
【个人信息界面】

用户个人信息界面总体上分两栏设计，左边放置用户个人信息，右边放置备忘录信息；管理员信息页面布局与用户界面布局类似，只是将个人信息转变为账户信息，备忘录转变为系统中所有管理员信息即可。初始样例如下（非最终实现界面）：



【班级信息界面】

班级信息主要包括班级公告和班级成员，两者采用的界面具有高度的相似性，样例如下所示：



# 3 编码实现

## 3.1 客户端实现

客户端采用jquery方式编写，主要的工作是从网页获取信息，并发送给服务器端，接收服务器端的返回信息，并显示在网页上。可能会有变化的情况：是否需要利用ajax发送数据，返回json 数据还是网页。

选取该页面的原因是：该页面布局相对之下比较有代表性，用到的html元素比较多，涉及到脚本动态响应的部分也比较具有代表性；最主要的是，该页面的功能也是系统的核心功能。代码节选如下：

<body>

<nav class="navbar navbar-inverse navbar-fixed-top">

<div class="container-fluid">

<div class="navbar-header">

<button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">find your class</a>

</div>

<div id="navbar" class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li><a href="/show\_manager\_home">管理员主页</a></li>

<li><a href="/show\_manage\_class">公告管理</a></li>

<li class="active"><a href="#">成员管理</a></li>

<li><a href="/manager\_logout">退出</a> </li>

</ul>

</div>

</div>

</nav>

<div class="container-fluid">

<div class="row">

<div class="col-sm-4 col-md-3 sidebar">

<div class="row placeholder">

<label>班级信息</label>

<div class="pull-right action-buttons">

<button type="button" onclick="AddClassShow()"><span class="glyphicon glyphicon-plus">添加</span></button>

</div>

</div>

<div class="row input-group">

<input type="text" class="form-control" id="search\_class\_key" placeholder="请输入班级名称">

<span class="input-group-btn">

<button class="btn btn-primary" type="button" onclick="SearchClass()" id="btnSearchMemo"><span class="glyphicon glyphicon-search"></span></button>

</span>

</div>

<div class="nav nav-sidebar">

<div class="row">

<div class="panel-body">

<ul id="list\_class" class="list-group">

</ul>

</div>

</div>

</div>

</div>

</div>

<div class="row placeholder" style="margin-left: 10px">

<label class="checkbox-inline">

<input type="checkbox" id="search\_by\_name" value="1">按姓名查找

</label>

<label class="checkbox-inline">

<input type="checkbox" id="search\_by\_sex" value="2">按性别查找

</label>

<label class="checkbox-inline">

<input type="checkbox" id="search\_by\_mail" value="4">按邮箱查找

</label>

<label class="checkbox-inline">

<input type="checkbox" id="search\_by\_loc" value="3">按居住地查找

</label>

</div>

下一部分就是完成页面数据的采集和传输工作，比如：在输入框中输入数据，点击按钮之类的要做些什么。该页面的脚本功能比较丰富，支持模糊查找和复选框查找，以及各种修改选项，故采用该页面的脚本作为代表展示整个系统功能。（jquery+ajax）

/\*获得当前班级的成员信息\*/

function GetUserNext(){

$('#ShowClassMate').html("<h4>"+localStorage.getItem('classNo')+"->成员信息</h4>"); /\*获得直接下标label名\*/

$.ajax({

url: '/get\_user\_by\_class', //这里是route到服务器端的功能名称

data:{

Id:localStorage.getItem('classId') //获取html页面上ID叫classId的元素

},

type: 'POST',

success: function (res) { //成功后服务器端查数据库返回的数据

var userObj = JSON.parse(res);

$('#list\_user').empty();

$('#list\_User\_Temp').tmpl(userObj).appendTo('#list\_user'); //显示在html页面对应的元素位置

},

error: function (error) {

console.log(error);

}

});

}

/\*第一次获得当前班级的成员信息\*/

function GetUserFirst(elem){

localStorage.setItem('classId', $(elem).attr('class-id'));

localStorage.setItem('classNo', $(elem).children().text());

GetUserNext();

}

## 3.2 服务端实现

服务器端采用python语言编写，主要的工作是从客户端获取通过ajax发送过来的消息，查询数据库后获得相应的信息并通过json返回数据信息。可能会有变化的情况：接收数据是利用ajax还是form表格发送的，返回json 数据还是网页。主要的有代表性的按操作分为一下四类：增/删/查/改。

# 返回root管理员主页

@app.route('/show\_manager\_home')

def show\_manager\_home():

m = session.get('manager')

if m is 1:

return render\_template('root\_maghome.html')

elif m > 0:

return render\_template('ord\_maghome.html')

else:

return render\_template('error.html', error=u'未授权访问！')

# root权限的管理员添加普通管理员功能，不成功时需要返回错误信息

@app.route('/create\_manager', methods=['POST'])

def create\_manager():

try:

if session.get('manager'): # 检测会话是否有效

account = request.form['Account']

password = request.form['Password']

conn = mysql.connect()

cursor = conn.cursor() # 连接数据库，获得数据指针

hash\_psd = generate\_password\_hash(password)

cursor.callproc('sp\_createManager', (account, hash\_psd))

data = cursor.fetchall() # 获取数据库返回信息

if len(data) == 0:

conn.commit()

return json.dumps({'status': 'OK'})

else:

return json.dumps({'status': '管理员已经存在！'})

else:

return render\_template('error.html', error=u'未授权访问！')

except Exception as e:

print e

return render\_template('error.html', error=str(e))

finally:

cursor.close()

conn.close()

# 管理员登录功能，登陆失败时返回错误信息

@app.route('/manager\_validate\_login', methods=['POST'])

def manager\_validate\_login():

try:

account = request.form['inputAccount']

password = request.form['inputPassword']

conn = mysql.connect()

cursor = conn.cursor()

cursor.callproc('sp\_validateManagerlogin', (account,))

data = cursor.fetchall()

if len(data) > 0:

if check\_password\_hash(str(data[0][1]), password):

session['manager'] = data[0][0]

return redirect('/show\_manager\_home')

else:

return render\_template('error.html', error=u'账号或密码错误！')

else:

return render\_template('error.html', error=u'未授权访问！')

except Exception as e:

print e

return render\_template('error.html', error=str(e))

finally:

cursor.close()

conn.close()

## 3.3 数据库设计

该部分将从数据库的初始化设计和通用存储过程模块设计两部分来阐述数据库的编码设计。

新建存储表的过程：

DROP TABLE IF EXISTS `tb\_manager`;

CREATE TABLE `tb\_manager` (

`manager\_id` bigint(20) NOT NULL AUTO\_INCREMENT,

`manager\_account` varchar(40) DEFAULT NULL,

`manager\_level` varchar(10) DEFAULT NULL,

`manager\_password` varchar(256) DEFAULT NULL,

PRIMARY KEY(`manager\_id`)

) ENGINE = InnoDB AUTO\_INCREMENT = 1 DEFAULT CHARSET = utf8;

一些常用的增删查改模块（特别要注意的是在删除记录时会出现的连带删除的情况，否则会出现错误）

--

-- Create User Procedure（新增模块）

--

DELIMITER ;;

CREATE DEFINER = `root`@`localhost` PROCEDURE `sp\_createUser`(

IN p\_account VARCHAR(40),

IN p\_password VARCHAR(256)

)

BEGIN

if (select exists(select 1 from tb\_user where user\_account = p\_account)) THEN

select '用户已经存在!';

ELSE

insert into tb\_user(user\_account,user\_password)values(p\_account,p\_password);

END IF;

END;;

DELIMITER ;

--

-- Create deleteClass Procedure（删除模块，注意连带删除的情况）

--

DELIMITER ;;

CREATE DEFINER = `root`@`localhost` PROCEDURE `sp\_deleteClass`(

IN p\_id bigint(20)

)

BEGIN

delete from tb\_class where class\_id = p\_id;

delete from tb\_notice where no\_class\_id = p\_id; --连带班级删除公告

delete from tb\_userclass where class\_id = p\_id; --连带删除班级成员

END;;

DELIMITER ;

--

-- Create getUser\_byid Procedure（查询模块）

--

DELIMITER ;;

CREATE DEFINER = `root`@`localhost` PROCEDURE `sp\_getUser\_byid`(

IN p\_id bigint(20)

)

BEGIN

select user\_name,user\_sex,user\_loc,user\_mail from tb\_user where user\_id = p\_id;

END;;

DELIMITER ;

--

-- Create updateClass\_byid Procedure（修改模块，修改特定信息即可）

--

DELIMITER ;;

CREATE DEFINER = `root`@`localhost` PROCEDURE `sp\_updateClass\_byid`(

IN p\_id bigint(20),

IN p\_title VARCHAR(40)

)

BEGIN

update tb\_class set class\_title=p\_title where class\_id = p\_id;

END;;

DELIMITER ;

学习心得

HTML语言是我们学习Web开发的基础，我们要掌握HTML语言常用标签、HTML表单、HTML窗口框架标记等等知识点，同时掌握HTML标记、元素、属性的使用，超链接标记和表格标记的使用方法，这个地方难点在于使用表格标记进行页面布局。在学习完HTML语言后，我们得对HTML页面进行美化，这时候我们就要学习CSS语言，通过掌握CSS样式细节来对HTML页面进行美化。后面我们就接触了JavaScript，我们需要掌握JavaScript基本语法、JavaScript函数和JavaScript对象等前端界面编程技术。

经过Web技术与应用课程的学习，我对前端和后端开发技术有了更全面的了解，并且能够使用这些技术开发网站。后端上学到的新知识更加丰富，体会Django可以方便管理数据库，在处理表单验证的优越性，并且在Django的MTV架构下，可以将应用的功能和界面分开，从而使代码更加清晰和易维护。这些技能是我们终身受用的，并且为我们以后的生活和工作提供了很大帮助。