自己写的代码

日历

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MyCalendar</title>
    <link rel="stylesheet" href="new_style.css">
    <script src="new_script.js" defer></script>
</head>
<body>
    <div class="contianer">
        <div class="calendar">
          <div class="calendar-header">
            <span id="year"> 年份 </span>
            <div class="month-picker" id="month-picker">
              <span class="month-change" id="pre-month">
                <
              </span>
              <span id="month"> 月份 </span>
              <span class="month-change" id="next-month">
                >
              </span>
            </div>
          </div>


          <div class="calendar-body">
            <div class="calendar-week-days">
              <div>Sun</div>
              <div>Mon</div>
              <div>Tue</div>
              <div>Wed</div>
              <div>Thu</div>
              <div>Fri</div>
              <div>Sat</div>
            </div>
            <div class="calendar-days">
            </div>
          </div>

        </div> <!-- calender结束 -->
      </div> <!-- container结束 -->
    </body>
  </html>
```

```
const isLeapYear = (year) => { // 闰年为可以被4整除但不能被100整除，或者能被400整除
    return (
      (year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
    );
  };

/*
（1）能被4整除且不能被100整除（如2004年是闰年，而1900年不是）
（2）能被400整除（如2000年是闰年）
还可以看2月份天数。2月份有29日，则是闰年
*/
const getFebDays = (year) => {
    return isLeapYear(year) ? 29 : 28;
  };

const month_names = [
  'January',
  'February',
  'March',
  'April',
  'May',
  'June',
  'July',
  'August',
  'September',
  'October',
  'November',
  'December',
];

let days_of_month = [
  31,
  getFebDays(year),
  31,
  30,
  31,
  30,
  31,
  31,
  30,
  31,
  30,
  31,
];

const generateCalendar = (month, year) => {
    let calendar_days = document.querySelector('.calendar-days'); // day
    calendar_days.innerHTML = '';

    let calendar_header_year = document.querySelector('#year'); // year
    let calendar_header_month = document.querySelector('#month'); // year

    // 根据传入的month修改英文字母
    // 根据传入的year修改后数字
    calendar_header_month.innerHTML = month+1;
    calendar_header_year.innerHTML = year;

    /*
    年份和月份生成Date，后面用getDay()获得第一天
    getDay()返回值：0代表星期日，1代表星期一，2表示星期二.....
    */
    let first = new Date(year, month);
    let first_day = first.getDay();

    let currentDate = new Date();
    let currentMonth = currentDate.getMonth();
    let currentYear = currentDate.getFullYear();
    let currentDay = currentDate.getDate(); // 返回31天中的某一天

    for (let i = 0; i <= days_of_month[month] + first_day - 1; i++) {

      let day = document.createElement('div');

      if (i >= first_day) {
        day.innerHTML = i - first_day + 1;
        if (i - first_day + 1 === currentDay &&
          year === currentYear &&
          month === currentMonth
        ) {
          day.classList.add('current-date');
        }
      }
      calendar_days.appendChild(day);
    }
  };


// 获取当前的年份和月份，便于动态生成日历
let currentDate = new Date();
let currentMonth = {"val": currentDate.getMonth()};
let currentYear = {"val": currentDate.getFullYear()};

// 根据选择的年份和月份生成日历
generateCalendar(currentMonth.val, currentYear.val);

let subMonth = (month) => {
  if (month === 0) return 11;
  else return month - 1;
}

let addMonth = (month) => {
  if (month === 11) return 0;
  else return month + 1;
}

document.querySelector('#pre-month').onclick = function() {
  if (currentMonth.val === 0) --currentYear.val;
  currentMonth.val = subMonth(currentMonth.val);
  generateCalendar(currentMonth.val, currentYear.val);
};

document.querySelector('#next-month').onclick = function() {
  if (currentMonth.val === 11) ++currentYear.val;
  currentMonth.val = addMonth(currentMonth.val);
  generateCalendar(currentMonth.val, currentYear.val);
};
```

```
  body {
    font-family: consolas;
    background:linear-gradient(to right, #9796f0, #fbc7d4);
    overflow: hidden;
  }

.contianer {
  position: relative;
  display: flex;
  justify-content: center;
}

.calendar {
  height: 600px;
  background-color: white;
  border-radius: 25px;
  padding: 30px 50px 0px 50px;
}

.calendar-header {
  background: #9796f0;
  color: white;
  display: flex;
  font-weight: 700;
  padding: 10px;
  user-select: none;
}

.month-picker {
  margin-left: 200px;
  padding-left: 20px;
}
.month-change:hover {
  cursor: pointer;
  transform: scale(2);
}

.calendar-week-days {
  display: flex;
  justify-content: space-around;
  /* display: grid;
  grid-template-columns: repeat(7, 1fr); */
  font-weight: 600;
  margin-top: 20px;
}

.calendar-days {
  display: grid;
  grid-template-columns: repeat(7, 1fr);
}

.calendar-days div{
  width: 37px;
  height: 33px;
  display: flex;
  align-items: center;
  justify-content: center;

  margin-top: 10px;
  padding: 5px;
  cursor: pointer;
  animation: to-top 1s forwards;
}


 .calendar-days div:hover {
  transition: width 0.2s ease-in-out, height 0.2s ease-in-out;
  background-color: #fbc7d4;
  border-radius: 20%;
  color: dark;
}

.calendar-days div.current-date {
  color: dark;
  background-color:#9796f0;
  border-radius: 20%;
}
```

输入表单验证

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        body, input, label {
            margin: 0;
            padding: 0;
        }
        .container {
            width: 400px;
            margin: 100px auto 0; /*居中*/
            padding: 1rem;
            box-shadow: rgba(0, 0, 0, 0.24) 0px 3px 8px;
        }
        label {
            display: block;
            color: grey;
        }
        .collector {
        }
        .head{
            text-align: center;
        }
        .form-item {
            /*居中输入框和label: 外部box的宽度+margin自动调节*/
            width: 280px;
            margin: 0 auto 1.5rem; /*居中*/
        }
        input {
            width: 280px;
            border: 1px solid lightgrey;
            border-radius: 6px;
            padding: 0.8rem; /*用padding把input撑起来*/
        }

        button {
            cursor: pointer;
            background-color: #3498db;
            border: 2px solid #3498db;
            border-radius: 4px;
            color: #fff;
            font-size: 16px;
            padding: 10px;
            margin-top: 20px;

            display: block;
            width: 100%;
        }
        small {
            /*空间占据*/
            visibility: hidden;
        }
        /*错误信息*/
        .error input{
            border: 1.5px solid rgb(247, 44, 4);
            display: block;
        }
        .error small {
            color:rgb(247, 44, 4);
            visibility:visible;
        }
        /*正确消息*/
        .success input{
            border: 1px solid green;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2 class="head">收集表单</h2>
        <form class="collector" id="form">
            <div class="form-item">
                <label for="username">Username</label>
                <input type="text" id="username" placeholder="请输入用户名">
                <small>Error message</small>
            </div>

            <div class="form-item">
                <label for="email">Email</label>
                <input type="text" id="email" placeholder="请输入邮箱">
                <small>Error message</small>
            </div>

            <div class="form-item">
                <label for="password">Password</label>
                <input type="password" id="password" placeholder="请输入密码">
                <small>Error message</small>
            </div>

            <div class="form-item">
                <label for="password2">Confirm Password</label>
                <input type="password"id="password2">
                <small>Error message</small>
            </div>
            <button type="submit">Submit</button>
        </form>

    </div>

    <script>
        const form = document.getElementById('form');
        const username = document.getElementById('username');
        const email = document.getElementById('email');
        const password = document.getElementById('password');
        const password2 = document.getElementById('password2');
        form.addEventListener('submit', (e) => {
            console.log("Submit")
            // 使用e.preventDefault()可以防止表单的默认提交行为,
            // 从而允许我们使用JavaScript来处理表单数据并采取其他操作
            e.preventDefault();
            // 判断是否都输入的数据
            if (!checkIsValid([username, email, password, password2])) {
                // 检查用户名长度和邮箱长度
                checkLength(username, 3, 15);
                checkLength(password, 6, 25);
                // 邮箱格式
                checkEmail(email);
                // 密码是否相同
                console.log("检查是否相同")
                checkPasswordsMatch(password, password2);
            }
        })


        // Get fieldname
        function getFieldName(input) {
            return input.id.charAt(0).toUpperCase() + input.id.substr(1);
        }

        function checkIsValid(params) {
            params.forEach((input) => {
                // 只输入空格也不行，trim去除两端空格
                if (input.value.trim()==="") {
                    // 显示错误
                    showError(input, `Please enter a valid
${getFieldName(input)}`);
                    return false;
                } else {
                    showSuccess(input);
                    return true;
                }
            })
        }

        // Show input error message
        function showError(input, message) {
            const formControl = input.parentElement;
            formControl.className = 'form-item error';
            const small = formControl.querySelector('small');
            small.innerText = message;
        }

        // Show success outline
        function showSuccess(input) {
            const formControl = input.parentElement;
            formControl.className = 'form-item success';
        }

        // Check email is valid
        function checkEmail(input) {
            const re = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|
(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-
9]+\.)+[a-zA-Z]{2,}))$/;
            if (re.test(input.value.trim())) {
            showSuccess(input);
            } else {
            showError(input, 'Email is not valid');
            }
        }

        // Check input length
        function checkLength(input, min, max) {
            if (input.value.length < min) {
```

```
            showError(
                input,
                `${getFieldName(input)} must be at least ${min} characters`
            );
            } else if (input.value.length > max) {
            showError(
                input,
                `${getFieldName(input)} must be less than ${max} characters`
            );
            } else {
            showSuccess(input);
            }
        }

        // Check passwords match
        function checkPasswordsMatch(input1, input2) {
            console.log(88);
            if (input1.value !== input2.value) {
                showError(input2, 'Passwords do not match');
            }
        }
    </script>
</body>
</html>
```

猜数字

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .continer{
            display: flex;
            position: relative;
            justify-content: center;
        }
        .guess-game {
            margin-top: 40px;
        }
        .gussed-num {
            margin-top: 15px;
        }
        .btn-restart {
            margin-top: 20px;
        }
        .wrong{
            margin-top: 5px;
            background-color: red;
        }
        .right{
            margin-top: 5px;
            background-color: rgb(0, 255, 81);
        }
    </style>
</head>
<body>
    <div class="continer">
        <div class="guess-game">

            <label for="input">请输入：</label>
            <input class="input" id="input" type="text" />
            <button class="guess">我猜！</button>

            <div class="gussed-num">
                <div>
                    <span>你已经猜过</span>
                    <span class="guessNum">0</span>
                    <span>次：</span>
                    <span class="guessVal"></span>
                </div>
                <div>
                    <span>还剩: </span>
                    <span class="restNum">5</span>
                    <span>次。</span>
                </div>
            </div>

            <div class="wrong">
                <!-- 错误提示 -->
            </div>
            <div class="right">
                <!-- 正确提示 -->
            </div>
            <div class="restart">
                <!-- 重新开始 -->
            </div>
        </div>
    </div>

<script>
    const getRandom = ()=>parseInt(Math.random()*100);
    // 生成随机数
    let randNum = getRandom();

    let container = document.querySelector('.container');
    let input = document.querySelector('.input')
    let button = document.querySelector('.guess');
    let guessNum = document.querySelector('.guessNum')
    let guessVal = document.querySelector('.guessVal')
    let wrong = document.querySelector('.wrong');
    let right = document.querySelector('.right');
    let restNum = document.querySelector('.restNum');
    let restart = document.querySelector('.restart')
```

```
// 次数限制
let limit = 5;
// 已经猜测的次数
let curGuess = 0;
console.log(randNum);
// 添加监听
button.onclick = startGuess;

// 重新开始的按钮
const generateRestartButton = () => {
    input.disabled = true;
    button.disabled = true;
    let newButton = document.createElement('button');
    newButton.innerHTML = "TRY AGAIN";
    newButton.onclick = startNewGame;
    newButton.classList.add('btn-restart');
    restart.appendChild(newButton);
}

// 开始新游戏，清除以前的变量
function startNewGame() {
    input.value = '';
    input.disabled = false;
    button.disabled = false;
    // 次数限制
    limit = 5;
    // 已经猜测的次数
    curGuess = 0;
    guessNum.innerHTML = curGuess;
    restNum.innerHTML = limit - curGuess;
    randNum = getRandom();
    //alert(randNum);
    console.log(randNum);
    wrong.innerHTML = '';
    right.innerHTML = '';
    restart.firstElementChild.remove();
}

// 验证输入数字的合法性
function isvaild(num) {
    var reg = /^((?!0)\d{1,2}|100)$/;
    if (!num.match(reg)) {
    return false;
    } else {
    return true;
    }
}

// 开始猜数字
function startGuess() {
    let inputVal = input.value;
    input.focus();
    // 判断空
    if (inputVal === ''){
        alert('Enter null');
```

```
            return;
        }
        // 正则表达式匹配0-100
        if (!isvaild(inputVal)){
            alert('Enter number is not a integer in 0-100');
            return;
        }

        curGuess++;
        guessNum.innerHTML = curGuess;
        restNum.innerHTML = limit - curGuess;
        if (curGuess === 5) {
            generateRestartButton();
            return;
        }

        wrong.innerHTML = "";
        if (inputVal > randNum) {
            //alert('你猜大了')
            wrong.innerHTML = "你猜大了"
        }
        else if(inputVal < randNum){
            //alert('你猜小了');
            wrong.innerHTML = "你猜小了"
        }
        else{
            //alert('你猜对了');
            right.innerHTML = "恭喜你，猜对了！"
            generateRestartButton();
        }

    }
</script>
</body>
</html>
```

CSU登录

管理员查询

使用form表单提交数据给Servlet

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
    <title>管理界面</title>
    <link rel="stylesheet" href="css/admin.css">
</head>
<body>
<div class="container">
```

```
    <h2>用户信息列表</h2>
    <%--  查询输入部分--%>
    <div class = "select">
        <form class="form-inline"
action="${pageContext.request.contextPath}/SelectUserServlet" method="post">
            <div class="form-group">
                <label>学号</label>
                <input  type="text" name="userid" value="${condition_userid}"
class="form-control" id="userid" style="width: 80px">
            </div>
            <div class="form-group">
                <label>姓名</label>
                <input type="text" name="name" value="${condition_name}"
class="form-control" id="name" style="width: 80px" >
            </div>
            <div class="form-group">
                <label>手机号</label>
                <input type="text" name="tel" value="${condition_tel}"
class="form-control" id="tel" style="width: 120px">
            </div>

            <div class="form-group">
                <label>邮箱</label>
                <input type="text" name="email" value="${condition_email}"
class="form-control" id="email"  >
            </div>
            <div class="btn">
                <input class ="select-btn"  type="submit" value="查询">
            </div>
        </form>

    </div>



    <%--  展示用户信息列表--%>
    <table  class="table">
            <tr class="thead">
                <th>学号</th>
                <th>姓名</th>
                <th>密码</th>
                <th>手机号</th>
                <th>邮箱</th>
                <th>操作</th>
            </tr>
            <c:forEach items="${sessionScope.users}" var="user" varStatus="s">
                <tr>
                    <td>${user.userid}</td>
                    <td>${user.username}</td>
                    <td>${user.password}</td>
                    <td>${user.tel}</td>
                    <td>${user.email}</td>
                    <td>
```

```
                        <a class="btn btn-default btn-sm"
href="${pageContext.request.contextPath}/UpdateUserF?userid=${user.userid}" >修改
</a>&nbsp;
                        <a class="btn btn-default btn-sm"
href="${pageContext.request.contextPath}/DeleteServlet?userid=${user.userid}">删
除</a>
                    </td>
                </tr>
            </c:forEach>
        </table>
    <div class = "add-btn"><a href="addUser.jsp">添加用户</a></div>
    <div class = "add-btn"><a href="successAdmin.jsp">退出查询</a></div>
</div>
```

对应的Servlet:

```
@WebServlet(name = "SelectUserServlet", value = "/SelectUserServlet")
public class SelectUserServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        HttpSession session = request.getSession();

        // 获取查询条件参数
        String userid = request.getParameter("userid");        // 输入的动态码
        String name = request.getParameter("name");
        String tel = request.getParameter("tel");
        String email = request.getParameter("email");

        User user = new User(userid,name,"",tel,email);

        // 查询满足条件的用户
        UserService userService = new UserServiceImpl();
        List<User> users = userService.showPartUser(user);

        //  设置查询条件参数，在页面中显示
        request.setAttribute("condition_userid",userid );
        request.setAttribute("condition_name",name );
        request.setAttribute("condition_tel",tel );
        request.setAttribute("condition_email",email );
        // 设置查询结果参数
        session.setAttribute("users",users);
        request.getRequestDispatcher("/admin.jsp").forward(request,response);

    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

使用ajax修改后的:

```
<h2>用户信息列表</h2>
<div class="select">
    <div class="form-inline" id="form-select">
        <div class="form-group">
            <label>学号</label>
            <input type="text" name="userid" value="${condition_userid}"
class="form-control" id="userid" style="width: 80px">
        </div>
        <div class="form-group">
            <label>姓名</label>
            <input type="text" name="name" value="${condition_name}" class="form-
control" id="name" style="width: 80px">
        </div>
        <div class="form-group">
            <label>手机号</label>
            <input type="text" name="tel" value="${condition_tel}" class="form-
control" id="tel" style="width: 120px">
        </div>
        <div class="form-group">
            <label>邮箱</label>
            <input type="text" name="email" value="${condition_email}"
class="form-control" id="email">
        </div>
        <div class="btn">
            <button class="select-btn">查询</button>
        </div>
    </div>
</div>

<table class="table" id="table-users">
    <thead>
        <tr>
            <th>学号</th>
            <th>姓名</th>
            <th>密码</th>
            <th>手机号</th>
            <th>邮箱</th>
            <th>操作</th>
        </tr>
    </thead>
    <tbody>
        // 待插入
    </tbody>
</table>
```

form.elements是所有输入项, 可以通过name属性获取输入的dom元素和输入值

`insertAdjacentHTML` 和 `appendChild` 都是用来向 DOM 中插入 HTML 内容的方法，不过它们有些许的不同之处：

- `appendChild` 方法将指定的节点插入到目标节点的子节点列表的末尾处。如果要添加多个子节点，需要逐一调用该方法。示例：`targetNode.appendChild(newNode)`。
- `insertAdjacentHTML` 方法将指定的 HTML 字符串解析为 DOM 节点，并插入到调用该方法的节点之前或之后。该方法一次可以插入多个节点。示例：`targetNode.insertAdjacentHTML('beforeend', '<div>Hello</div><div>World</div>')`。
    - `'beforebegin'`：在当前元素的前面插入 HTML 字符串。
    - `'afterbegin'`：在当前元素的开始标签后面插入 HTML 字符串。
    - `'beforeend'`：在当前元素的结束标签前面插入 HTML 字符串。**(默认)**
    - `'afterend'`：在当前元素的后面插入 HTML 字符串。

因此，`insertAdjacentHTML` 可以更方便的一次性添加多个节点，而 `appendChild` 只能逐一添加单个节点。另外，`insertAdjacentHTML` 也更加灵活，可以插入在目标节点的前面或后面，以及作为目标节点的第一个或最后一个子节点，而 `appendChild` 只能作为目标节点的最后一个子节点。

```
// 获取表单元素和按钮元素
var form = document.querySelector('form');
var submitBtn = document.querySelector('.select-btn');

// 注册按钮点击事件
submitBtn.addEventListener('click', function(event) {
    // 阻止表单默认提交事件
    event.preventDefault();

    // 获取表单数据
    var userid = form.elements['userid'].value;
    var name = form.elements['name'].value;
    var tel = form.elements['tel'].value;
    var email = form.elements['email'].value;

    // 发送AJAX请求
    var xhr = new XMLHttpRequest();
    xhr.open('POST', '${pageContext.request.contextPath}/SelectUserServlet');
    xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
            if (xhr.status === 200) {
                // 获取后端返回的用户数据
                var users = JSON.parse(xhr.responseText);
                // 渲染用户数据
                renderUsers(users);
            } else {
                console.error('Request failed: ' + xhr.status);
            }
        }
    };
    xhr.send('userid=' + encodeURIComponent(userid) +
             '&name=' + encodeURIComponent(name) +
             '&tel=' + encodeURIComponent(tel) +
             '&email=' + encodeURIComponent(email));
});

function renderUsers(users) {
    // 获取用户列表元素
    var userList = document.querySelector('.table tbody');

    // 清空用户列表
    userList.innerHTML = '';

    // 遍历用户数据，生成HTML代码
    users.forEach(function(user) {
        var html = '<tr>' +
                   '<td>' + user.userid + '</td>' +
                   '<td>' + user.username + '</td>' +
                   '<td>' + user.password + '</td>' +
                   '<td>' + user.tel + '</td>' +
                   '<td>' + user.email + '</td>' +
                   '<td>' +
                   '<a class="btn btn-default btn-sm"
href="${pageContext.request.contextPath}/UpdateUserF?userid=' + user.userid + '">
修改</a>&nbsp;' +
                   '<a class="btn btn-default btn-sm"
href="${pageContext.request.contextPath}/DeleteServlet?userid=' + user.userid +
'">删除</a>' +
                   '</td>' +
                   '</tr>';
        // 将HTML代码添加到用户列表中
        userList.insertAdjacentHTML('beforeend', html);
    });
 }
```

ajax修改后对应的Servlet

```
@WebServlet("/SelectUserServlet")
public class SelectUserServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 设置字符编码
        response.setContentType("application/json;charset=utf-8");

        // 获取查询条件
        String userid = request.getParameter("userid");
        String name = request.getParameter("name");
        String tel = request.getParameter("tel");
        String email = request.getParameter("email");

        // 根据查询条件查询用户
        List<User> userList = UserService.selectUser(userid, name, tel, email);

        // 转换为JSON格式字符串
        String jsonStr = JSON.toJSONString(userList);

        // 响应结果
        PrintWriter out = response.getWriter();
        out.write(jsonStr);
        out.flush();
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

数据库查询操作

```
public List<User> selectUser(String userid, String name, String tel, String
email) {
    // 初始化用户列表
    List<User> userList = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    try {
        // 获取数据库连接
        conn = getConnection();
        // 构建查询语句
        String sql = "SELECT * FROM user WHERE 1=1";
        if (userid != null && !userid.equals("")) {
            sql += " AND userid=?";
        }
        if (name != null && !name.equals("")) {
            sql += " AND username=?";
        }
        if (tel != null && !tel.equals("")) {
            sql += " AND tel=?";
        }
        if (email != null && !email.equals("")) {
            sql += " AND email=?";
        }
        // 构建 PreparedStatement 对象
        pstmt = conn.prepareStatement(sql);
        // 绑定参数
        int index = 1;
        if (userid != null && !userid.equals("")) {
            pstmt.setString(index++, userid);
        }
        if (name != null && !name.equals("")) {
            pstmt.setString(index++, name);
        }
        if (tel != null && !tel.equals("")) {
            pstmt.setString(index++, tel);
        }
        if (email != null && !email.equals("")) {
            pstmt.setString(index++, email);
        }
        // 执行查询操作
        rs = pstmt.executeQuery();
        // 遍历结果集，将每条记录封装成 User 对象并添加到 userList 中
        while (rs.next()) {
            User user = new User();
            user.setUserid(rs.getString("userid"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            user.setTel(rs.getString("tel"));
            user.setEmail(rs.getString("email"));
            userList.add(user);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        // 关闭数据库连接和资源
        closeConnection(conn, pstmt, rs);
    }
    return userList;
}
```

管理员修改/删除/添加用户

- 修改:
  - Servlet中id->user->修改.jsp
  - 修改.jsp填写表单, Servlet处理表单并删除数据库中的user
  - 同步当前数据库中所有user到session中
  - 跳转到用户列表, 重新渲染所有用户
- 删除
  - Servlet: id->user->删除user
  - 同步当前数据库中所有user到session中
  - 跳转到用户列表, 重新渲染所有用户
- 添加
  - 添加.jsp填写表单, Servlet处理表单并向数据库添加user
  - 同步当前数据库中所有user到session中
  - 跳转到用户列表, 重新渲染所有用户

修改

1. Servlet中id->user->修改.jsp

```
@WebServlet(name = "UpdateUserF", value = "/UpdateUserF")
public class UpdateUserF extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        String userid = request.getParameter("userid");
        UserService userService = new UserServiceImpl();
        User user = userService.haveId(userid);
        request.setAttribute("user",user);
request.getRequestDispatcher("/updateUser.jsp").forward(request,response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

2. 修改.jsp填写表单

```
<div class="container">
    <h2>修改用户信息</h2>
    <div class = "container_main">
        <form action="/UpdateUserServlet" method="post">
            <div class="form-group">
                <label>学号：</label>
                <input type="text" class="form-control" id="userid" name="userid"
value="${requestScope.user.userid}" readonly>
            </div>

            <div class="form-group">
                <label>姓名：</label>
                <input type="text" class="form-control" id="username"
name="username" value="${requestScope.user.username}">
            </div>

            <div class="form-group">
                <label >密码：</label>
                <input type="text" class="form-control" id="password"
name="password" value="${requestScope.user.password}">
            </div>

            <div class="form-group">
                <label >手机号：</label>
                <input type="text" class="form-control" id="tel" name="tel"
value="${requestScope.user.tel}">
            </div>

            <div class="form-group">
                <label >邮箱：</label>
                <input type="text" class="form-control" id="email" name="email"
value="${requestScope.user.email}">
            </div>

            <div class="btn">
                <input class ="add-btn"  type="submit" value="修改">
            </div>
        </form>

    </div>

</div>
```

输入框的css

```
.container .container_main label{
    display: inline-block;
    width: 70px;
    padding: 10px;
}
.container .container_main input{
    border: 1px solid #ccc;
    border-radius: 4px;
    line-height: 30px;
    height: 30px;
    width: 300px;
    margin: 10px 0;
}
.container .container_main input:focus{
    outline:none;
}
```

3. Servlet处理表单并删除数据库中的user

```
@WebServlet(name = "UpdateUserServlet", value = "/UpdateUserServlet")
public class UpdateUserServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        String userid = request.getParameter("userid");
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String tel = request.getParameter("tel");
        String email = request.getParameter("email");

        UserService userService = new UserServiceImpl();
        User user = userService.haveId(userid);
        user.setUsername(username);
        user.setPassword(password);
        user.setTel(tel);
        user.setEmail(email);

        userService.updateUser(user);

        request.getRequestDispatcher("/AdminServlet").forward(request,response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

4. **同步当前数据库中所有user到session中; 跳转到用户列表, 重新渲染所有用户**

```
@WebServlet(name = "AdminServlet", value = "/AdminServlet")
public class AdminServlet extends HttpServlet {         // 产生所有用户
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        // 加载得到所有的用户信息
        UserService userService = new UserServiceImpl();
        List<User> user = userService.showAllUser();

        //创建Jackson的核心对象  ObjectMapper
        ObjectMapper mapper = new ObjectMapper();
        String json = mapper.writeValueAsString(user);
        System.out.println(json);

        HttpSession session = request.getSession();
        session.setAttribute("json",json);
        session.setAttribute("users",user);
        request.getRequestDispatcher("/admin.jsp").forward(request,response);
        //response.sendRedirect(request.getContextPath() +"/admin.jsp");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

删除

1. Servlet: id->user->删除user

```
@WebServlet(name = "DeleteServlet", value = "/DeleteServlet")
public class DeleteServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        String userid = request.getParameter("userid");
        UserService userService = new UserServiceImpl();
        userService.delete(userid);

        request.getRequestDispatcher("/AdminServlet").forward(request,response);
    }
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

2. 同步当前数据库中所有user到session中; 跳转到用户列表, 重新渲染所有用户

添加

1. 添加.jsp填写表单, Servlet处理表单并向数据库添加user

```
<div class="container">
    <h2>修改用户信息</h2>
    <div class = "container_main">
        <form action="/UpdateUserServlet" method="post">
            <div class="form-group">
                <label>学号：</label>
                <input type="text" class="form-control" id="userid" name="userid"
value="${requestScope.user.userid}" readonly>
            </div>

            <div class="form-group">
                <label>姓名：</label>
                <input type="text" class="form-control" id="username"
name="username" value="${requestScope.user.username}">
            </div>

            <div class="form-group">
                <label >密码：</label>
                <input type="text" class="form-control" id="password"
name="password" value="${requestScope.user.password}">
            </div>

            <div class="form-group">
                <label >手机号：</label>
                <input type="text" class="form-control" id="tel" name="tel"
value="${requestScope.user.tel}">
            </div>

            <div class="form-group">
                <label >邮箱：</label>
                <input type="text" class="form-control" id="email" name="email"
value="${requestScope.user.email}">
            </div>

            <div class="btn">
                <input class ="add-btn"  type="submit" value="修改">
            </div>
        </form>
    </div>

</div>
```

```
@WebServlet(name = "UpdateUserServlet", value = "/UpdateUserServlet")
public class UpdateUserServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        String userid = request.getParameter("userid");
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String tel = request.getParameter("tel");
        String email = request.getParameter("email");

        UserService userService = new UserServiceImpl();
        User user = userService.haveId(userid);
        user.setUsername(username);
        user.setPassword(password);
        user.setTel(tel);
        user.setEmail(email);

        userService.updateUser(user);

        request.getRequestDispatcher("/AdminServlet").forward(request,response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```

2. 同步当前数据库中所有user到session中; 跳转到用户列表, 重新渲染所有用户

中转Servlet, 用于将管理员界面选择的用户id->用户实体, 并传输给下一个jsp页面

```
@WebServlet(name = "UpdateUserF", value = "/UpdateUserF")
public class UpdateUserF extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        String userid = request.getParameter("userid");
        UserService userService = new UserServiceImpl();
        User user = userService.haveId(userid);
        request.setAttribute("user",user);
request.getRequestDispatcher("/updateUser.jsp").forward(request,response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request,response);
    }
}
```