1. 网站开发需求

我们不可避免的要和许多数据打交道，而一张图表可以让数据变的形象而直观。JavaScript 图表库 ECharts 可以在前端生成各种各样的图表，但对图表的初始化配置却稍显繁琐。利用这次课程作业的机会，我实现了一个有着基本图表管理功能的网站 ChartE（E 取 Easier、Evolution 之意）。

ChartE 的核心功能有两个：数据管理与图表生成。

对数据的管理可以说是绝大多数网站都离不开的功能，ChartE 设计创建了 Articles 与 Users 两张表来储存数据（具体参见 init.sql），Users 表中有用户名、密码和管理员标识这三列，Articles 表中有编号、用户名、标题、数据和公开访问标识这五列。此外，在 init.sql 中初始化了 admin 用户（及密码 123456）并创建了 3 条示例数据。

ChartE 选用了 json 来实现图表的生成。如同在 excel 中的图表一样，各种图表大都可以类似的抽象为一串 json 数据。ChartE 通过读取 json 中的”type”键来判断类型，进而根据类型来获取预先存放的初始化参数，读取”data”键中的信息来生成图表。

目前，ChartE 设计了图表管理，图表编辑，密码设置三个子页面。

2. 主页设计与实现

2.1 布局设计和主页效果图

ChartE 的主页设计的比较简单，如图 2-1-1 所示，首次访问从上到下依次为导航栏，bootstrap 网格（一行三列，每列中为图片展示和示例链接），登录表单。

在用户登录后，导航栏会显示对其他页面的链接，且再次访问主

页时不显示登录表单。

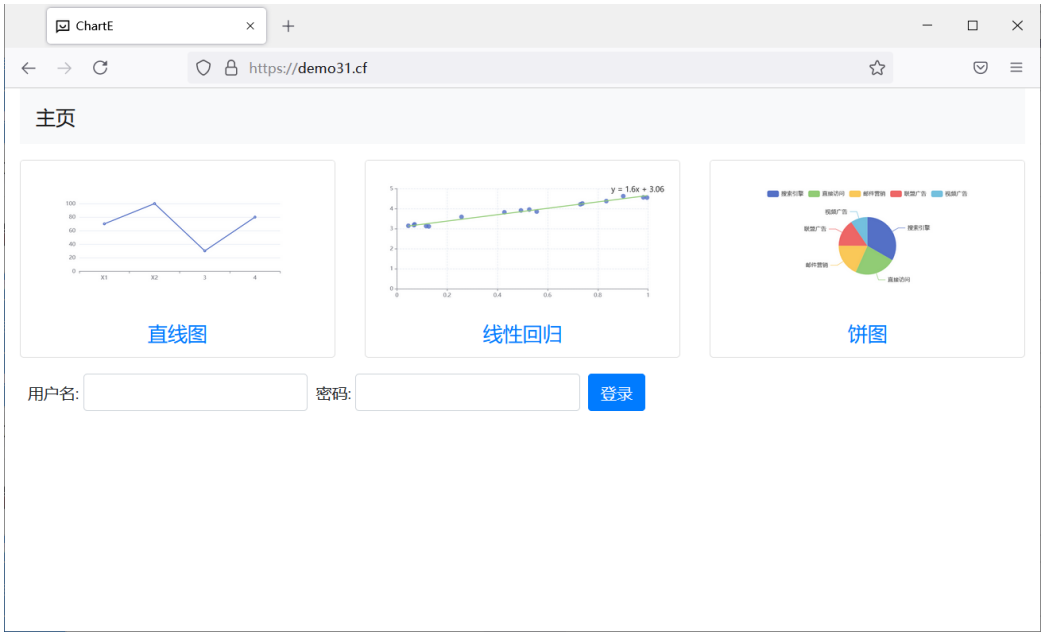

图 2-1-1 首次访问主页

2.2 核心 css 代码

由于使用了 bootstrap，页面布局大多通过类来实现。如图 2-2-1 所示，导航栏使用了 bg-light 类来改变背景颜色，而类 mb-i、ml-i、mt-i 分别设置了下方（below）、左边（left）、上方（top）的边距（margin）。此外，子页面中的居中效果也是用 m-auto 类而非单独的 css 代码实现。

```
<nav class="navbar navbar-expand-lg navbar-light bg-light mb-3">…</nav> flex
<div class="row"> flex
  <div class="col-md-4 mb-2">…</div>
  <div class="col-md-4 mb-2">…</div>
  <div class="col-md-4 mb-2">…</div>
</div>
<form class="form-inline" method="post" action="Login" role="form"> flex
  <label class="ml-2 mt-2" for="unInput">…</label> flex
```

图 2-2-1 布局 HTML

2.3 登录模块

为实现登录功能，前端创建了一个 Post 表单向后端 Servlet 发送请

求，后端通过用户名及密码参数查询数据库中是否有匹配项。若有查询结果，则登录成功，同时在 session 储存用户名等数据用于后续验证，若无结果，则提示失败。

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    // TODO Auto-generated method stub
    String username = request.getParameter("un");
    String pwd = request.getParameter("pwd");
    String sql = "select isAdmin from Users where Uname=? and pwd=?";
    HttpSession session = request.getSession(true);
    try {
        ArrayList<Object> arr = JDBCUtils.fetchOne(sql, username, pwd);
        if (arr != null) {
            Boolean isAdmin = (Boolean) arr.get(0);
            session.setAttribute("un", username);
            session.setAttribute("isAdmin", isAdmin);
            response.sendRedirect("manage.jsp");
        } else {
            session.setAttribute("alert", "用户名或密码错误");
            response.sendRedirect("index.jsp");
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        response.getWriter().append("jdbc failed");
    }
}
```

图 2-3-1 Login.java

值得一提的是，登录模块中实现了前端密码加密功能。大致思路为设计一个类型为 hidden 的 input，同时用 javascript 绑定密码输入框的 onchange 事件，每次用户键入密码时计算字符串”username_password”的 sha256 值并设置为 hidden 元素值。

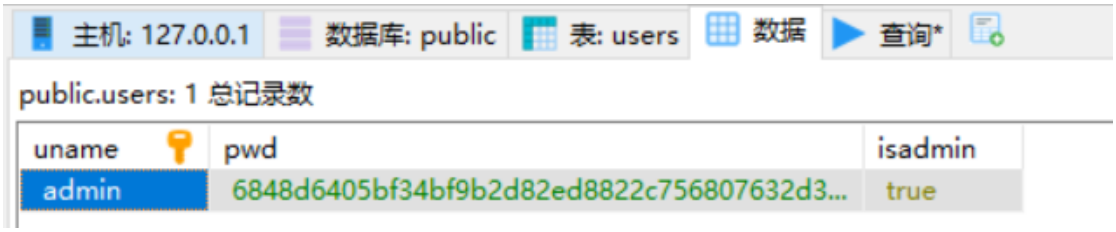

图 2-3-2 数据示例

3. 管理页面的设计与实现

3.1 目的

展示已创建图表；增加或删除图表。（由于个人精力有限，网站暂未实现用户管理功能）

3.2 页面设计图

页面从上到下设计了导航栏，数据表格，数据分页，新建表单。

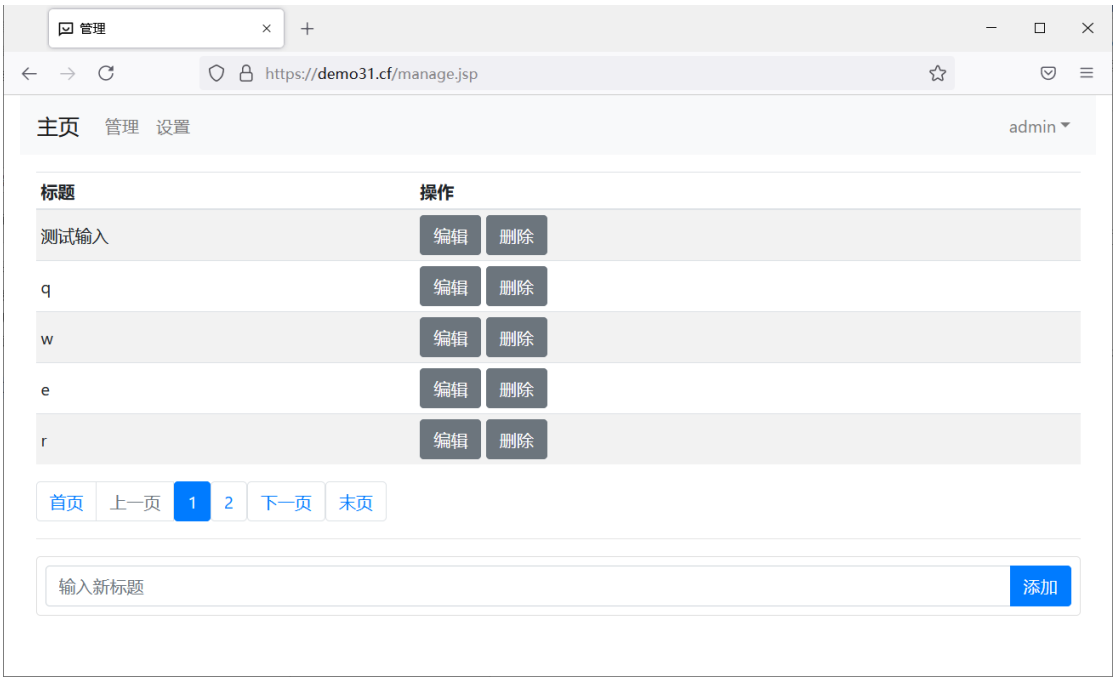

图 3-2-1 管理页面

3.3 核心服务端代码

```
<div class="row">
    <div class="col-md-12">
        <%@ include file="navbar.jsp" %>
        <div class="col-md-12">
            <div class="table-responsive">
                <table class="table table-striped table-sm">
                    <thead>
                        <tr>
                            <th>标题</th>
                            <th>操作</th>
                        </tr>
                    </thead>
                    <tbody>
                        <%@ include file="table_articles.jsp" %>
                    </tbody>
                </table>
            </div>
            <nav>
                <ul class="pagination" id="pages">
                    <li class="page-item">
                        <a class="page-link" href="?p=1">首页</a>
                    </li>
                    <li id="prevLink" class="page-item">
                        <a class="page-link" href="#">上一页</a>
                    </li>
                </ul>
            </nav>
```

图 3-3-1 manage.jsp

管理页面的核心是生成表格部分。为简化代码，manage.jsp 引入了 navbar.jsp 和 table_articles.jsp。前者顾名思义，通过 session 参数生成了页面中的导航栏部分；后者通过数据库交互生成表格。

```
ArrayList<ArrayList<Object>> result = JDBCUtils.fetchAB(sql2, from, from+5, username);

if (result!=null) {
    // 生成表格
    for (int i=0;i<result.size();i++) {
        ArrayList<Object> cols = result.get(i);
        String title = (String)cols.get(0);
        String Ano = String.valueOf(cols.get(1));
%>

<tr>
    <td><%=title%></td>
    <td><%=Ano%></td>
</tr>
<%
}
}
%>
<div>
    <script>
        var activeP = <%= p %>;
        const totalP = <%= totalP %>;
    </script>
</div>
```

图 3-3-2 表格生成（table_articles.jsp 后半部分）

如图 3-3-2 所示，for 循环产生了需要的行，下方的 script 传递了一些分页参数。

```
String p2 = request.getParameter("p");

int eachP=5;

int p=-1;
if(p2==null) p=1;
else p=Integer.parseInt(p2);

String username = session.getAttribute("un").toString();

String sql1 = "select count(*) from Articles where Uname=?";

String sql2 = "select title, Ano from Articles where Uname=? ORDER BY Ano";

long total = JDBCUtils.getCount(sql1, username);

long totalP = ((total-1) / eachP) +1;

if(p<1) response.sendRedirect("manage.jsp?p=1");
else if (p>totalP) response.sendRedirect("manage.jsp?p="+String.valueOf(totalP));

int from = (p-1)* 5;

ArrayList<ArrayList<Object>> result = JDBCUtils.fetchAB(sql2, from, from+5, username);
```

图 3-3-3 分页参数（table_articles.jsp 前半部分）

如图 3-3-3 所示，table_articles.jsp 执行了一些参数校检工作，然后根据数据库查询结果计算分页参数。这里对数据库的查询用到了自己实现的 getCount 和 fetchAB 函数，这将在后面的第六部分介绍。

```
var pages = document.getElementById("pages");

var prev = document.getElementById("prevLink");

// 部分代码省略

function initP() {
  for (var i = start; i <= end; i++) {
    var t = document.createElement("li");
    t.innerHTML = `<li class="page-item"><a class="page-link" href="?p=${i}">${i}</a></li>`;
    if (i === activeP) t.children[0].classList.add("active");
    pages.append(t);
  }
  var t = document.createElement("li");
  t.innerHTML = `<li class="page-item"><a class="page-link" href="#">下一页</a></li>`;
  if (end === activeP) t.children[0].classList.add("disabled");
  else t.children[0].children[0].href = `?p=${activeP + 1}`;
  pages.append(t);

  var t2 = document.createElement("li");
  t2.innerHTML = `<li class="page-item"><a class="page-link" href="?p=${totalP}">末页</a></li>`;
  pages.append(t2);
}

initP();
```

图 3-3-4 分页生成（pages.js 部分内容）

如图 3-3-4 所示，管理页面的分页实际上是用 javascript 而非 jsp 生成的。Jsp 实际上只传入了当前页数和总页数两个参数，原始的分页部分只包含首页、上一页两个节点，javascript 根据传入的参数创建了后续分页节点。

此外，为了防止 jsp 生成标题时出现<script>之类的 xss 问题，服务端对数据进行了 base64 编码，当然 manage.js 中也有对应的解码。

3.4 运行效果

运行效果见 3.2 的页面设计部分。后续页面运行效果也一并在页面设计部分展示，不再单独列出。

4. 编辑页面的设计与实现

4.1 目的

预览与编辑图表。

4.2 页面设计图

导航栏；页面左边用于展示图表，右边上部分为引入的编辑器，下部分为编辑表单。

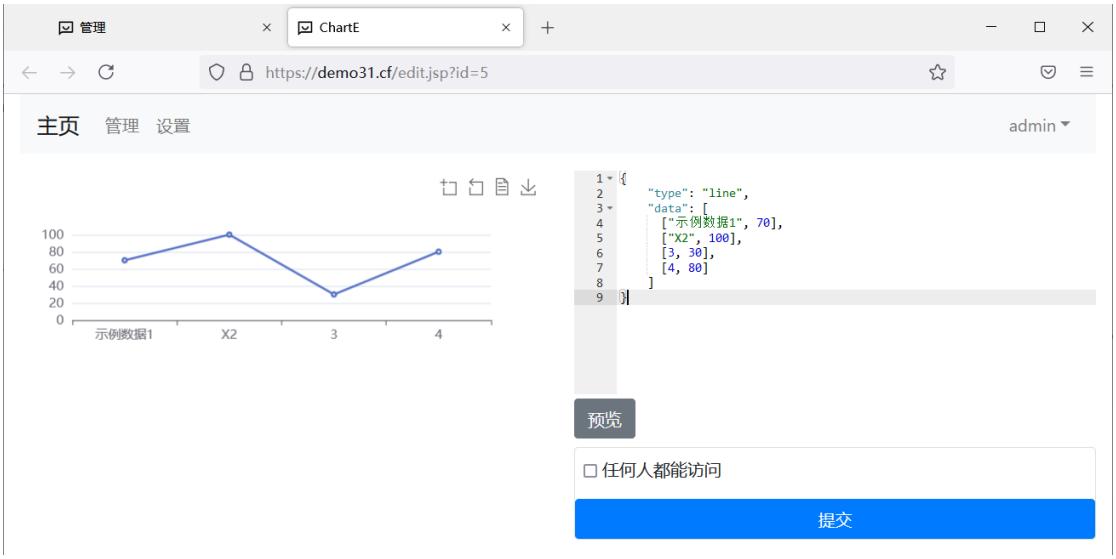

图 4-2-1 编辑页面

```
<body>
    <div class="container-fluid">
        <div class="row">
            <div class="col-md-12">
                <%@ include file="navbar.jsp" %>
                <div id="mainDiv" class="row">
                    <div class="col-md-6">
                        <div id="chartDiv"></div>
                    </div>
                    <div class="col-md-6">
                        <div id="editor"></div>
                        <button id="preview" type="button" class="btn btn-secondary mt-1">预览</butt
                        <form id="form" class="form card mt-2" method="POST" action="EditAData" rol
                            <input type="hidden" name="id" value="<%=Ano%>" />
                            <input id="dataH" type="hidden" name="data" />
                            <div class="checkbox p-2">
                                <label>
                                    <input id="cBox" name="isPub" type="checkbox" /> 任何人都能访问
                                </label>
                            </div>
                            <button type="submit" class="btn btn-primary btn-block">提交</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
```

图 4-2-2 edit.jsp（部分）

4.3 核心服务端代码

```
String sql = "select b64data, isPub, Uname from Articles where Ano=?";

String b64data = null;
String Uname="";
boolean exists=false;
boolean isPub=false;

ArrayList<Object> arr = JDBCUtils.fetchOne(sql, Integer.parseInt(Ano));

if (arr!=null) {
    exists=true;
    b64data = (String)arr.get(0);
    Object isPub2=arr.get(1);
    if(isPub2!=null) isPub=(Boolean)isPub2;
    String Uname2=(String)arr.get(2);
    if(Uname2!=null) Uname=Uname2;
}

boolean isMine = Uname.equals(session.getAttribute("un"));

if(!exists||(!isMine&&!isPub)){
    response.sendRedirect("manage.jsp");
    return;
}

%>

<head>
    <script>
        var isMine = "<%=isMine%>";
        var isPub = "<%=isPub%>";
        var b64str = "<%=b64data%>";
```

图 4-2-3 edit.jsp（部分）

编辑页面的核心是图表生成，但这是通过客户端 javascript 实现的，且生成部分主要是按照文档对所使用库进行了初始化等操作，此处略去。（具体实现可在 edit.js、getOpt.js 中找到）

如图 4-2-3 所示，除了常规的数据库交互，服务端也对图表进行了权限判断，公开文件可以被所有人访问。

```
HttpSession session = request.getSession(true);
if (session.getAttribute("un") == null) {
    response.sendRedirect(Common.indexPage);
    return;
} else {
    String username = session.getAttribute("un").toString();
    String Ano = request.getParameter("id");
    String data = request.getParameter("data");
    String isPub_ = request.getParameter("isPub");
    boolean isPub = (isPub_ != null && isPub_ != "");
    String sql = "update Articles set b64data=?, isPub=? where Ano=? and Uname=?";
    try {
        if (Ano != null && data != null) {
            String b64data = Base64.getEncoder().encodeToString(data.getBytes("utf-8"));
            if (!JDBCUtils.exUPD(sql, b64data, isPub, Integer.parseInt(Ano), username))
                session.setAttribute("alert", "编辑失败");
            else
                session.setAttribute("alert", "提交成功");
            String referer = request.getHeader("Referer");
            if (referer == null)
                referer = "manage.jsp";
            response.sendRedirect(referer);
        } else {
            response.getWriter().append("failed");
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        response.getWriter().append("jdbc failed");
    }
}
```

图 4-2-4 EditAdata.java（部分）

如图 4-2-4 所示，后端用 servlet 实现了数据编辑。另外，操作结果的提示信息放到了 session 中。这是因为所返回页面中引入了 getAlert.jsp，这个 jsp 会检查 session 是否为空，不为空则会输出一段 html 的 script 代码生成提示。

由于 ChartE 中用到的 servlet 大都是对数据的增删改查，重复度较高，之后就不进行说明了。

此外，类似 3.3，服务端对发送的 json 也进行了 base64 编码。

5. 设置页面的设计与实现

5.1 目的

修改用户密码。

5.2 页面设计图

导航栏；输入表单（这里用 javascript 实现了密码明文显示及输入验证）。

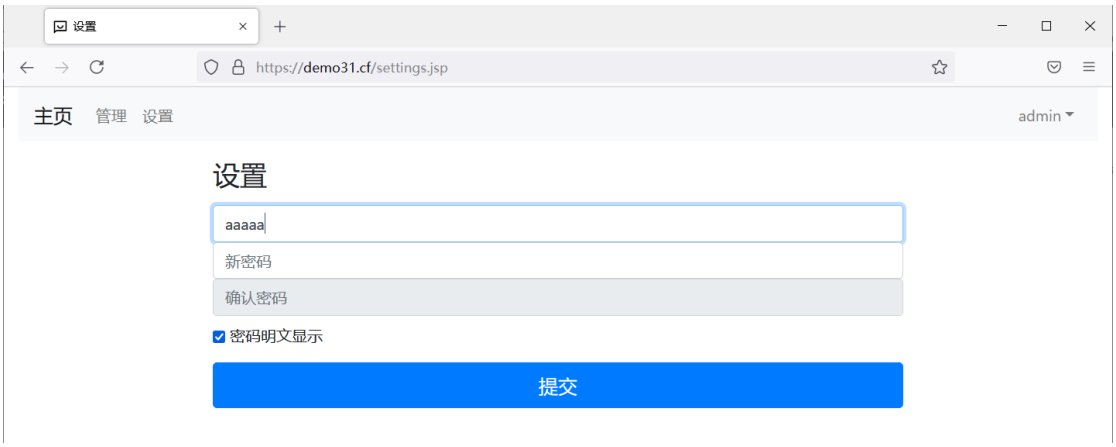

图 5-2-1 设置页面

```
var cbox = document.getElementById("cbox");
var inputP2 = document.getElementById("inputP2");
var inputP3 = document.getElementById("inputP3");

cbox.onclick = function () {
  if (cbox.checked) {
    inputP1.type = "text";
    inputP2.type = "text";
    inputP3.attributes.removeNamedItem("required");
    inputP3.value = "";
    inputP3.attributes.setNamedItem(document.createAttribute("disabled"));
  } else {
    inputP1.type = "password";
    inputP2.type = "password";
    inputP3.attributes.setNamedItem(document.createAttribute("required"));
    inputP3.attributes.removeNamedItem("disabled");
  }
};

document.getElementById("sForm").onsubmit = function () {
  if (cbox.checked) return true;
  else {
    var result = inputP2.value === inputP3.value;
    if (result === false) alert("密码不一致");
    return result;
  }
};
```

图 5-2-2 setting.js（部分）

5.3 核心服务端代码

设置页面的核心是后端 servlet 处理，可参考图 4-2-4。

6. 其他说明

6.1 响应式布局

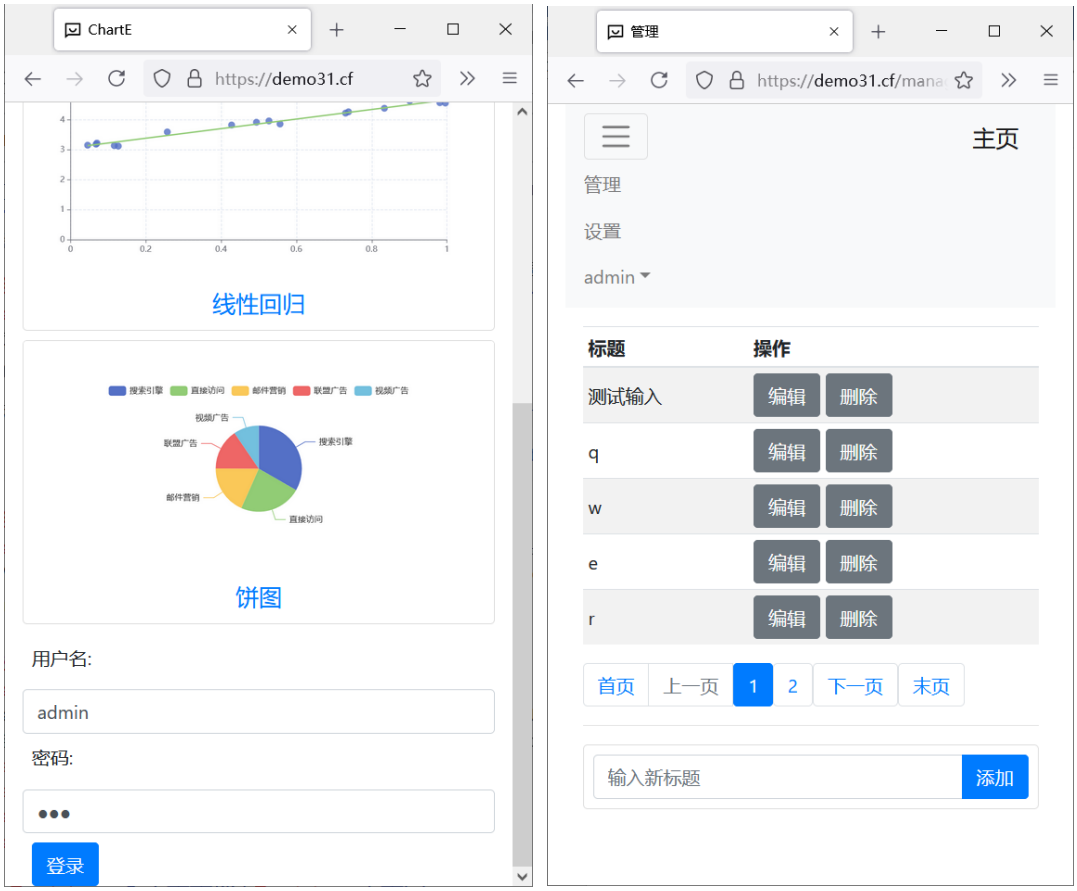

图 6-1-1 编辑页面

如图 6-1-1 所示，网站借助 bootstrap 实现了响应式布局。

6.2 网页整合

网站的各页面都引入了 navbar.jsp。

```
<%
if (username2 != null) {
%>
<ul class="navbar-nav">
    <li class="nav-item"><a class="nav-link"
        href="manage.jsp">管理</a></li>
    <li class="nav-item"><a class="nav-link"
        href="settings.jsp">设置</a></li>
</ul>
<ul class="navbar-nav ml-md-auto">
    <li class="nav-item dropdown"><a
        class="nav-link dropdown-toggle" id="navbarDropdownMenuLink"
        data-toggle="dropdown"><%=username2.toString()%></a>
        <div class="dropdown-menu dropdown-menu-right"
            aria-labelledby="navbarDropdownMenuLink">
            <a class="dropdown-item" href="logout.jsp">注销</a>
        </div></li>
</ul>
<%
}
%>
```

图 6-2-1 navbar.jsp（部分）

6.3 风格一致性

网站通过 bootstrap 的使用及网页整合大致实现了风格一致。

6.4 公共类设计

```
public static ArrayList<Object> fetchOne(String sql, Object... args) throws Exception {
    ArrayList<Object> arr = new ArrayList<Object>();

    Connection conn = getConnection();

    PreparedStatement pst = conn.prepareStatement(sql);
    for (int i = 0; i < args.length; i++) {
        pst.setObject(i + 1, args[i]);
    }

    ResultSet rs = pst.executeQuery();
    try {
        if (rs.next())
            for (int i = 0; i < rs.getMetaData().getColumnCount(); i++) {
                Object a = rs.getObject(i + 1);
                arr.add(a);
            }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        release(conn);
        release(pst);
        release(rs);
    }

    if (arr.size() == 0)
        arr = null;
    return arr;
}
```

图 6-4-1 JDBCUtils（fetchOne）

如图 6-4-1 所示，服务端设计了 fetchOne 方法，通过返回 ArrayList 简化连接释放操作。fetchAB 方法也是类似，只不过额外增加了一些循环与条件检查，返回二维数组。而 getCount 实际上是对 fetchOne 的调用，该方法要求输入的 sql 查询语句为 count 语句，然后处理了下返回值。

6.5 网页安全方面

如 6.4 公共类设计所述，服务端借助 JDBCUtils 公共类操作数据库，没有直接拼接 sql 语句，尝试规避 sql 注入问题。

如 2.3 登录模块所述，网站在前端对密码进行了 hash 加密，以降低用户密码泄露的危害。

如 3.3、3.4 所述，服务端对部分数据进行了 base64 编码，尝试规避 xss 攻击。

如图 4-2-3 所示，编辑页面有着一定的权限检查功能，此外，管理、设置页面也进行了 session 检查。

6.6 部署