中国矿业大学计算机学院

2017 级本科生课程报告

课程名称 Web 应用开发技术

报告时间 2020.10.30

学生姓名 袁孝健

学　　号 06172151

专　　业 信息安全

任课教师 杨勇

《Web 技术》评分表

编号	毕业要求	课程教学目标	考查方式与考查点	占比	得分
1	1.3	目标 1：掌握 Web 应用程序的工作原理， CSS3 的语法，应用 CSS3+DIV 技术实现页面美化与布局，JavaScript 内置核心语言对象以及基本语法 JSP 页面的基本构成，掌握 Servlet 的工作原理、技术特点、分类等基础的 Web 应用程序开发的基础知识，了解通过 Web 技术解决复杂工程问题的基本方法。	1) 针对具体要求，运用 HTML、CSS、JavaScript、JSP 内置对象设计并开发静态页面和动态页面。 2) 能够针对实际问题，合理地分析和选择 Web 前端技术与 Web 服务器端开发技术（JSP、Servlet）进行 Web 系统设计。	50%	
2	3.4	目标 2:熟悉目前几种主流的 Web 开发技术，掌握 4 种内置对象属性范围的区别与应用原则，掌握 MVCII 模式的原理与特点，使用该模式进行 Web 应用程序开发。能够针对具体复杂应用问题，在开发平台上设计实施方案，完成程序的编码、调试和运行，并能对结果进行初步的分析和评价。	1）能够针对实际应用问题，合理地分析和选择 Html，第三方 JavaScript 库解决客户端设计问题，选择合理的开发模式进行 Web 系统开发。 2）结合具体较为复杂工程需求，设计与编程实现、调试，并能对结果进行初步分析与评价。	50%	
总分				100%	

评阅人：

日 期：

摘　要

在本次 Web 应用开发课程设计中，进行了 3 个实验，实验一，我开发了一个电影主题的网站，并使用 HTML，CSS，JavaScript 技术首先完成了静态页面的编写，在实验过程中为了方便开发并且设计出更美观的视觉效果，使用前端框架 Bootstrap 和 jQuery 库辅助开发；实验二中，我将实验一中的静态页面改写为动态页面,使用了 JSP 和 Servlet 技术对相应的功能进行了一定的实现，使网站开发的效果更进一步；实验三中，在前两个实验的基础上，将网站改写为 MVC 模式，并使用了数据库知识，对网站功能进行分析，并设计了相应的数据表，成功实现了相应数据的增删改查，最终完成了一个具有完善实际功能的电影主题网站。

关键词：Web 开发，电影网站，前端技术，Servlet

目 录

实验一 静态 Web 页面设计

1.1 实验说明

1.1.1 实验目的

1、配置 Web(TOMCAT)服务器，了解 Web 工作原理。
2、熟悉常用 HTML 5 标记的含义，能够熟练使用这些标记设计静态 Web 页面，实现 Web 页面上各种元素的合理布局，如表单、表格、图片以及框架等标记的使用。
3、了解 CSS 样式表的定义和使用方法，能够使用 CSS 美化和布局 Web 页面。
4、掌握 JavaScript 脚本语言的基本语法。
5、能够使用 JavaScript 与浏览器对象进行交互。
6、能够使用 JavaScript 处理表单和表单元素事件。

1.1.2 实验要求

1、使用 HTML 5 开发 Web 静态页面。按照 HTML 5 的规范设计与开发网站。
2、练习 HTML 5 的新 HTML5 新特性和效果。
3、练习使用 Web 页面开发工具（MyEclipse、Dreamwear、VS.NET 或其它）
4、练习第三方 JavaScript 库的使用。
5、完成实验报告和实验成果

1.2 实验内容

1.2.1 页面设计

（1）网站主题

出于本人对电影的喜爱，本次实验设计了一个与电影主题相关的网站，包括首页、电影分类页面、电影详情页面、注册与登录页面以及用户主页共六个静态页面。

（2）页面展示

- 首页：

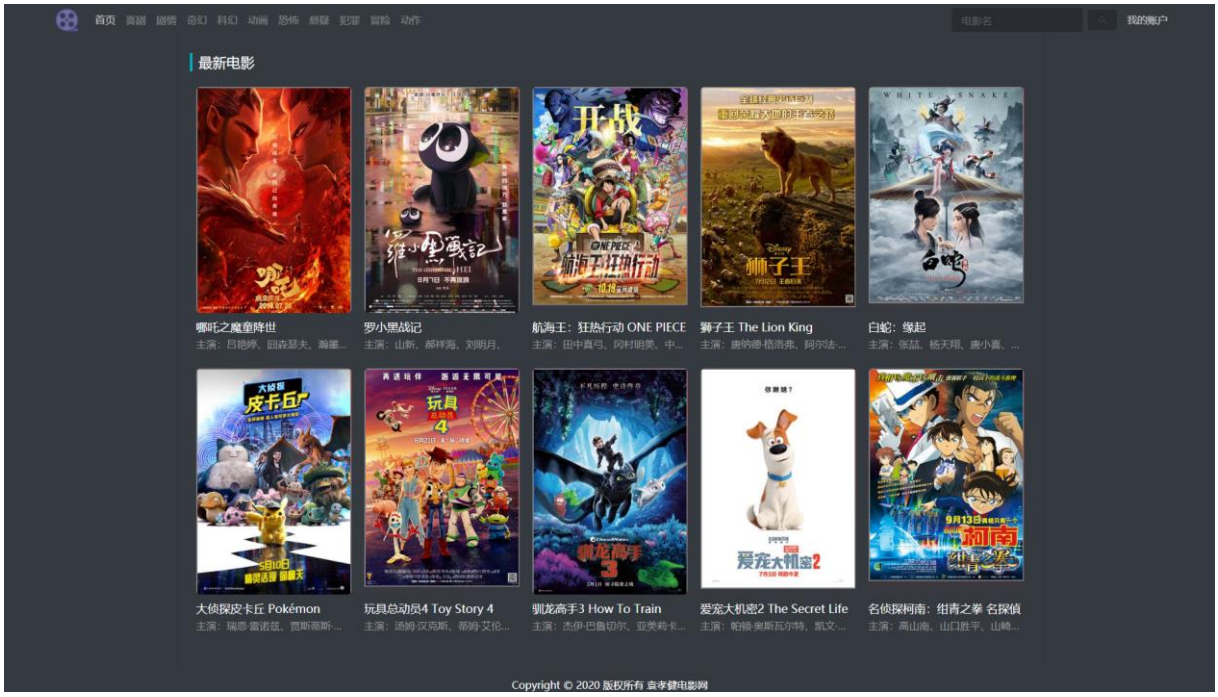

图 1-1 首页

- 分类页面：

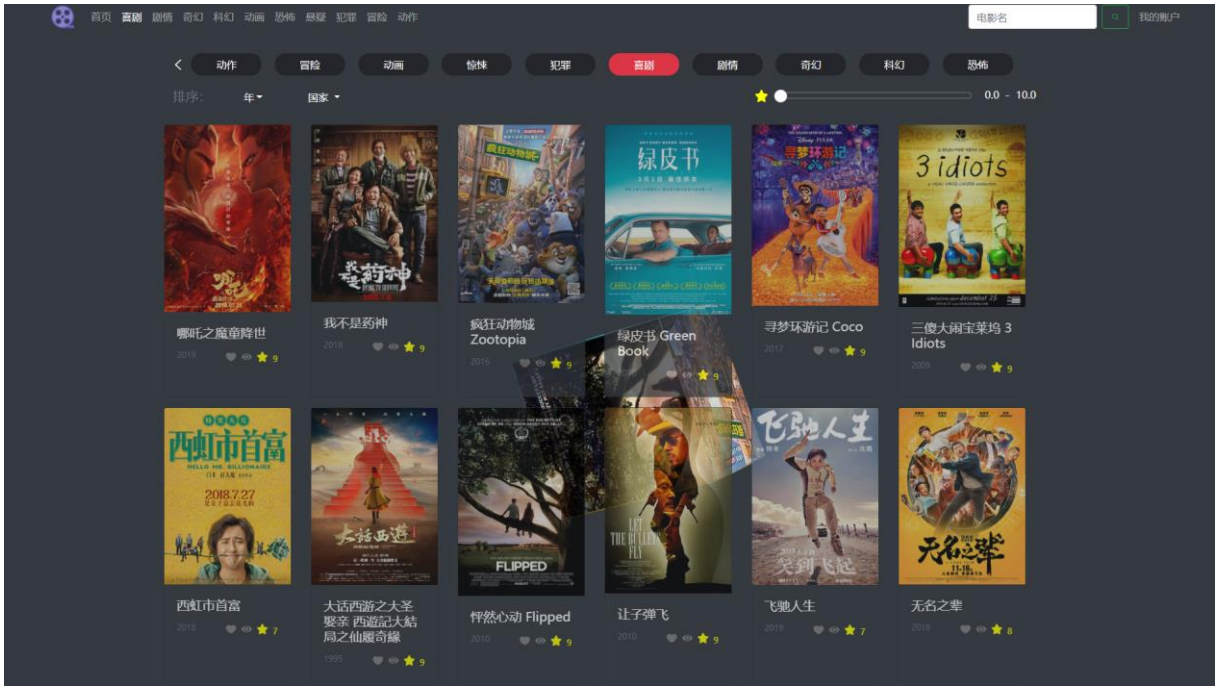

图 1-2 分类页面

- 电影详情：

图 1-3 电影详情

- 注册页面：

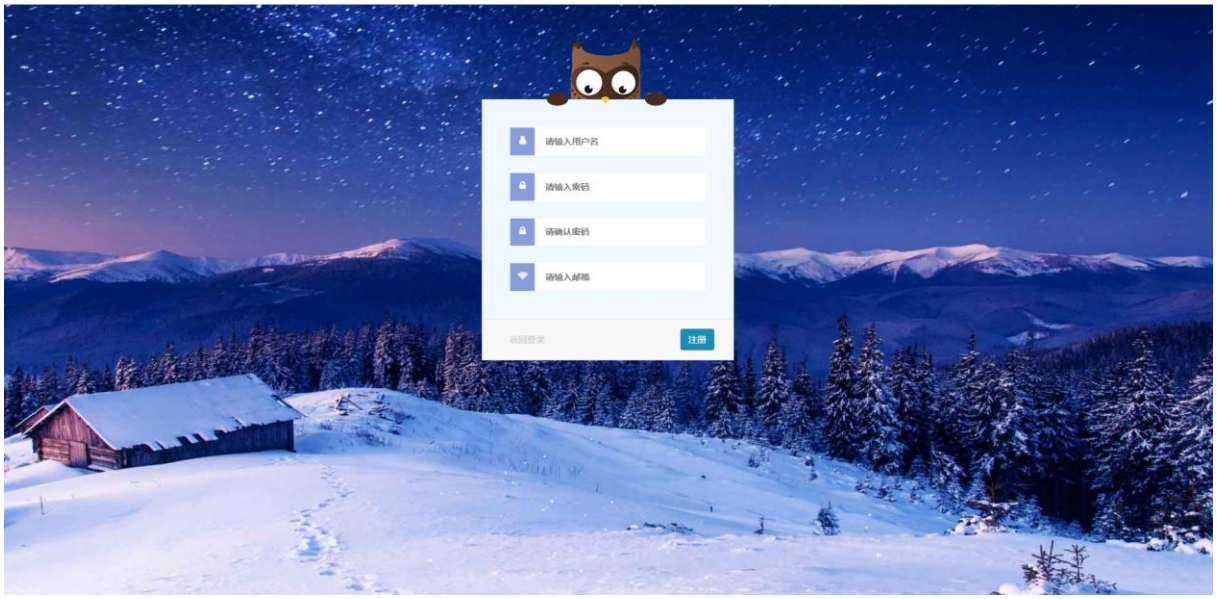

图 1-4 注册页面

- 登陆页面：

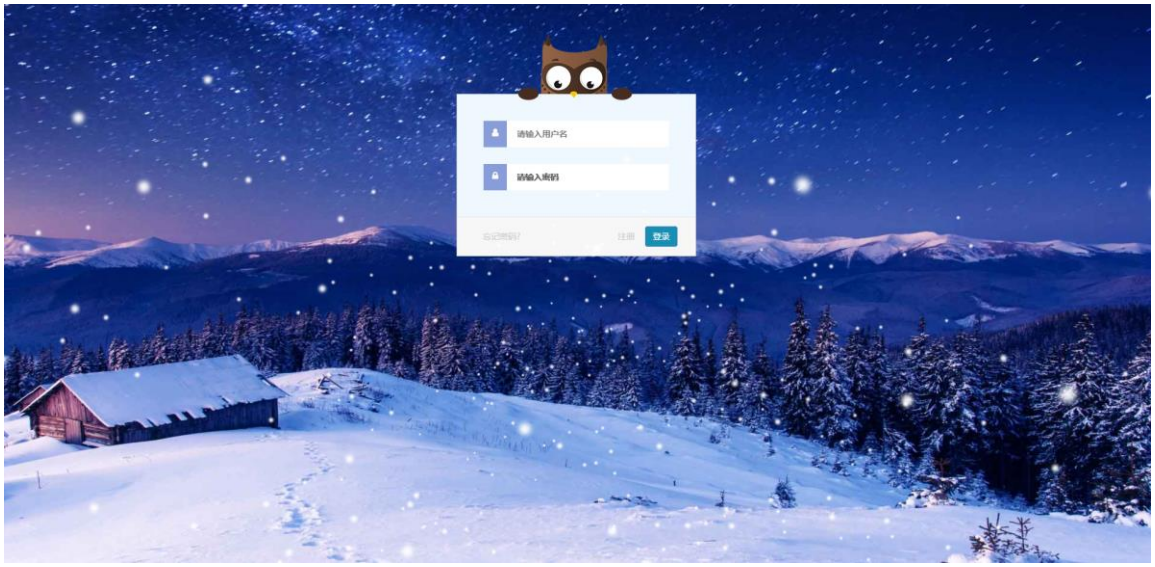

图 1-5 登录页面

- 用户主页：

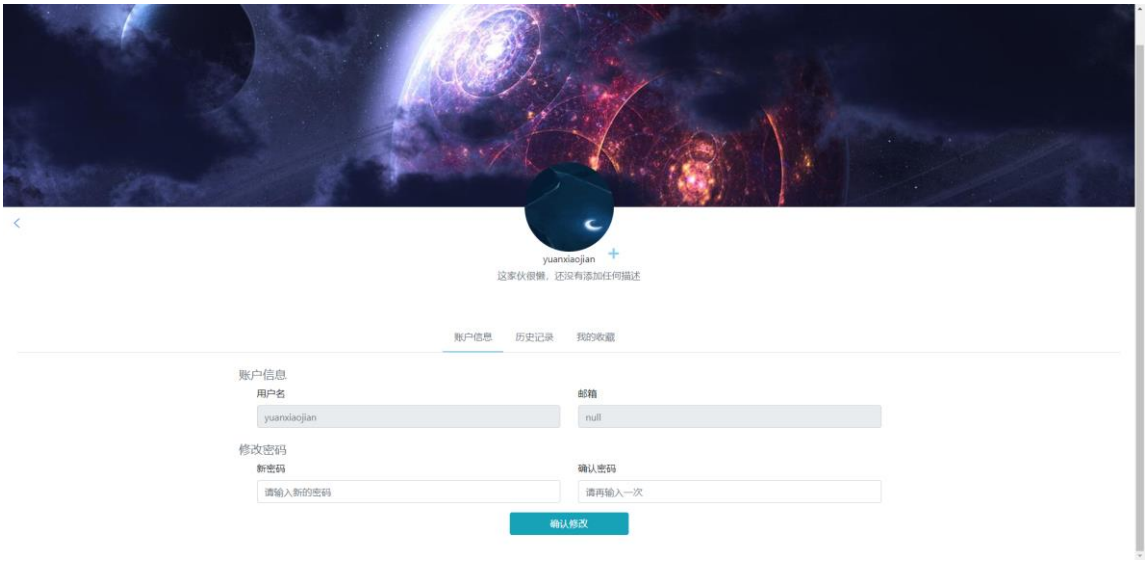

图 1-6 用户主页

1.2.2 CSS 布局与美化

（1）Bootstrap 简介

本实验中，我使用了 bootstrap 框架来快速构建页面，使用前需要在页面前先进引入：

```
<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->
<script src="./files/jquery-3.4.1.min.js"></script>
<script src="./files/popper.min.js"></script>
<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->
<script src="./files/bootstrap.min.js"></script>
<script src="./files/bootstrap.bundle.min.js"></script>
```

图 1-7 引入 bootstrap 框架

（2）主页导航栏部分 html 代码如下：

```
<header class="ml-5">
    <nav class="navbar navbar-expand-md navbar-dark">
        <a class="navbar-brand" href="/main.do">
            <img src="./files/movie.png" alt="" style="height: auto; width:
40px;">
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarCollapse"
            aria-controls="navbarCollapse" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <!-- 分类 -->
        <div class="collapse navbar-collapse" id="navbarCollapse">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="/main.do">首页
                        <span class="sr-only">(current)</span>
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">喜剧</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">剧情</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">奇幻</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/">科幻</a>
                </li>
                <li class="nav-item">
```

```
                <a class="nav-link" href="/">动画</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/">恐怖</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/">悬疑</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/">犯罪</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/">冒险</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="/">动作</a>
            </li>
        </ul>

        <!-- 搜索 -->
        <form class="form-inline mt-2 mt-md-0 mr-3" action="/search.do"
method="get">
            <input class="form-control mr-sm-2" type="text" name="search"
placeholder="电影名" value=""
                aria-label="Search">
            <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">
                <span class="iconfont iconsousuo"></span></button>
        </form>

        <!-- 用户 -->
        <ul class="navbar-nav mr-4">
            <li class="nav-item dropdown">
```

```
                <a href="/loginOrRegister.do">我的账户</a>

                <div class="dropdown-menu">
                    <a class="dropdown-item" href="/history">历史浏览</a>
                    <a class="dropdown-item" href="/userInfo">修改账户</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="/login.do">登录</a>
                </div>
            </li>
        </ul>
    </div>
</nav>
</header>
```

由于这一部分的效果是直接利用 Bootstrap 实现的，因此其 css 代码位于 bootstrap.min.css 中，而我们只需要在对应的标签中用 class 调用相应的 css 格式即可。

效果如下图所示：

图 1-8 首页导航栏

（3）首页主体部分 html 代码如下（省略部分重复代码）：

```
<section>
<div class="middle-show">
<div class="middle-cont">
<div class="middle-content">
<div class="title">
    <h2>
        <a class="text-white" href="/main.do#">最新电影</a>
    </h2>
</div>
<div class="movie">
    <ul class="movie-img">

        <li>
            <div class="box">
                <a href="/detail.do?movieName=哪吒之魔童降世" style="height:
354.667px;">
                    <img class="thumb lazy initial loaded" src="./files/1.jpg"
```

```
alt="">
                </a>
                <div class="box-content">
                    <a href="/detail.do?movieName=哪吒之魔童降世">
                        <h3 class="title">哪吒之魔童降世</h3>
                    </a>
                    <br>
                    <a
                        href="https://www.douban.com/link2/?url=">
                        <span class="iconfont iconbofang1" style="font-size:
2.5em;"></span>
                    </a>
                </div>
            </div>

            <h4 class="dytit">
                <a target="_blank" href="/detail.do?movieName=哪吒之魔童降世">
                    哪吒之魔童降世</a>
            </h4>
            <p class="inzhuy">主演：吕艳婷、囧森瑟夫、瀚墨、陈浩、绿绮、张珈铭、杨卫、
</p>
        </li>

        <li>
            <div class="box">
                <a href="/detail.do?movieName=罗小黑战记" style="height:
354.667px;">
                    <img class="thumb lazy initial loaded" src="./files/2.jpg"
alt="">
                </a>
                <div class="box-content">
                    <a href="/detail.do?movieName=罗小黑战记">
                        <h3 class="title">罗小黑战记</h3>
                    </a>
                    <br>
                    <a
                        href="https://www.douban.com/link2/?url=">
                        <span class="iconfont iconbofang1" style="font-size:
2.5em;"></span>
                    </a>
                </div>
            </div>
```

```
            <h4 class="dytit">
                <a target="_blank" href="/detail.do?movieName=罗小黑战记">
                    罗小黑战记</a>
            </h4>
            <p class="inzhuy">主演：山新、郝祥海、刘明月、</p>
        </li>

        <!-- 省略重复代码 -->

        <li>
            <div class="box">
                <a href="/detail.do?movieName=驯龙高手 3 How To Train Your Dragon:
The Hidden World"
                    style="height: 354.667px;">
                    <img class="thumb lazy initial loaded" src="./files/30.jpg"
alt="">
                </a>
                <div class="box-content">
                    <a href="/detail.do?movieName=驯龙高手 3 How To Train Your
Dragon: The Hidden World">
                        <h3 class="title">驯龙高手 3 How To Train Your Dragon: The
Hidden World</h3>
                    </a>
                    <br>
                    <a
                        href="https://www.douban.com/link2/?url=">
                        <span class="iconfont iconbofang1" style="font-size:
2.5em;"></span>
                    </a>
                </div>
            </div>

            <h4 class="dytit">
                <a target="_blank" href="/detail.do?movieName=驯龙高手 3 How To
Train Your Dragon: The Hidden World">
                    驯龙高手 3 How To Train Your Dragon: The Hidden World</a>
            </h4>
            <p class="inzhuy">
                主演：杰伊·巴鲁切尔、亚美莉卡·费雷拉、F·默里·亚伯拉罕、凯特·布兰切特、
杰拉德·巴特勒......
            </p>
```

```
        </li>

        <li>
            <div class="box">
                <a href="/detail.do?movieName=爱宠大机密 2 The Secret Life of Pets
2" style="height: 354.667px;">
                    <img class="thumb lazy initial loaded" src="./files/31.jpg"
alt="">
                </a>
                <div class="box-content">
                    <a href="/detail.do?movieName=爱宠大机密 2 The Secret Life of
Pets 2">
                        <h3 class="title">爱宠大机密 2 The Secret Life of Pets 2</h3>
                    </a>
                    <br>
                    <a
                        href="https://www.douban.com/link2/?url=">
                        <span class="iconfont iconbofang1" style="font-size:
2.5em;"></span>
                    </a>
                </div>
            </div>

            <h4 class="dytit">
                <a target="_blank" href="/detail.do?movieName=爱宠大机密 2 The
Secret Life of Pets 2">
                    爱宠大机密 2 The Secret Life of Pets 2</a>
            </h4>
            <p class="inzhuy">
                主演: 帕顿·奥斯瓦尔特、凯文·哈特、哈里森·福特、艾瑞克·斯通斯崔特、珍妮·斯
蕾特......
            </p>
        </li>
    </ul>
</div>
</div>
</div>
</div>
</section>
```

其 css 代码如下：

```
.movie li a {
    display: block;
```

```
    position: relative;
    overflow: hidden;
}
.row {
    background: #ffffff;
}
.movie {
    max-width: none;
    height: auto;
    margin: 0 auto;
    margin-top: 20px;
    padding: 0 20px;
}
.movie-img::after {
    content: ' ';
    display: table;
    clear: both;
}
.movie-img li {
    margin-bottom: 2%;
    position: relative;
    float: left;
    box-sizing: border-box;
    width: 20%;
    padding: 0 10px;
}
.movie-img li a img {
    width: 100%;
    max-width: 100%;
    transform: scale(1);
    object-fit: cover;
    border: 2px solid gray;
}
ul {
```

```
    list-style: none;
}
a {
    color: #999;
    text-decoration: none;
    cursor: pointer;
}
img {
    vertical-align: top;
}
.dytit {
    text-align: left;
    height: 30px;
    line-height: 30px;
    margin-top: 7px;
}
.dytit {
    color: #666;
    display: block;
    /* white-space: nowrap; */
    overflow: hidden;
    font-family: "Microsoft YaHei";
}
.inzhuy {
    height: 24px;
    line-height: 24px;
    color: #999;
    white-space: nowrap;
    overflow: hidden;
    font-family: "Microsoft YaHei";
    text-overflow: ellipsis;
}
.mid {
    overflow: hidden;
```

```
    max-height: 550px;
    max-width: 1360px;
    margin: 0 auto;
    padding: 0 20px;
}
.dropdown {
    position: relative;
    margin-right: 20px;
}
.carousel-inner img {
    width: 100%;
    /* height: auto;*/
    /* max-width: 100%; */
    max-height: 500px;
}
body {
    font-family: -apple-system, BlinkMacSystemFont, 'Microsoft YaHei',
sans-serif;
    font-size: 16px;
    color: #333;
    font-weight: 400;
}
.box {
    background: linear-gradient(#ff0b30, #343a40);
    font-family: 'Merriweather Sans', sans-serif;
    border-radius: 7px;
    position: relative;
    overflow: hidden;
}
```

效果如下图所示：

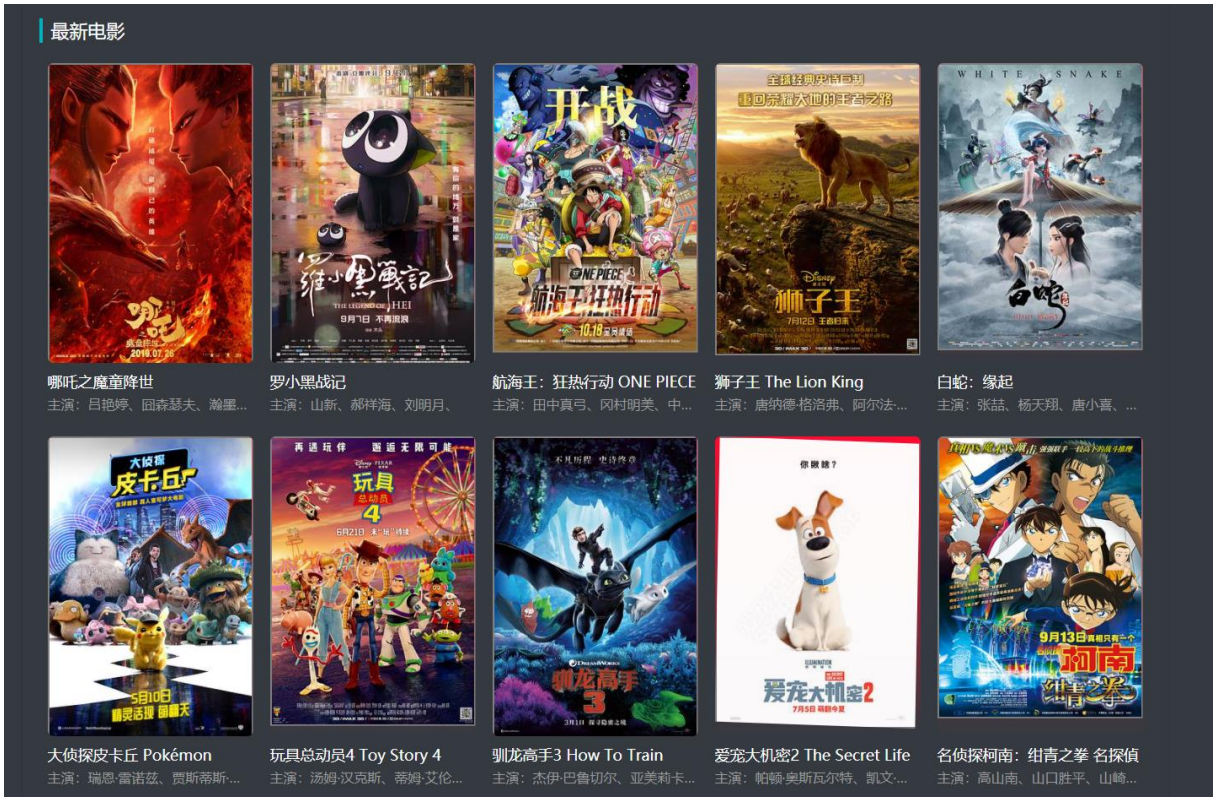

图 1-9 首页主体部分

1.2.3 JavaScript 验证

（1）登录页面

使用 JavaScript 验证用户名和密码是否为空，以及限制用户名只能为 15 位以下的字母，代码如下：

```
function login() { //登录
    var username = $("#login-username").val(),
        password = $("#login-password").val(),
        validatecode = null,
        flag = true;
    //判断用户名密码是否为空
    if (username == "") {
        $.pt({
            target: $("#login-username"),
            position: 'r',
            align: 't',
            width: 'auto',
```

```
            height: 'auto',
            content: "用户名不能为空"
        });
        flag = false;
    }
    if (password == "") {
        $.pt({
            target: $("#login-password"),
            position: 'r',
            align: 't',
            width: 'auto',
            height: 'auto',
            content: "密码不能为空"
        });
        flag = false;
    }
    //用户名只能是 15 位以下的字母或数字
    var regExp = new RegExp("^[a-zA-Z0-9_]{1,15}$");
    if (!regExp.test(username)) {
        $.pt({
            target: $("#login-username"),
            position: 'r',
            align: 't',
            width: 'auto',
            height: 'auto',
            content: "用户名必须为 15 位以下的字母或数字"
        });
        flag = false;
    }

    if (!flag) {
        return false;
    } else { //登录
        //调用后台登录验证的方法
```

```
        return true;
    }
}
```

效果如图所示：

- 合法时：

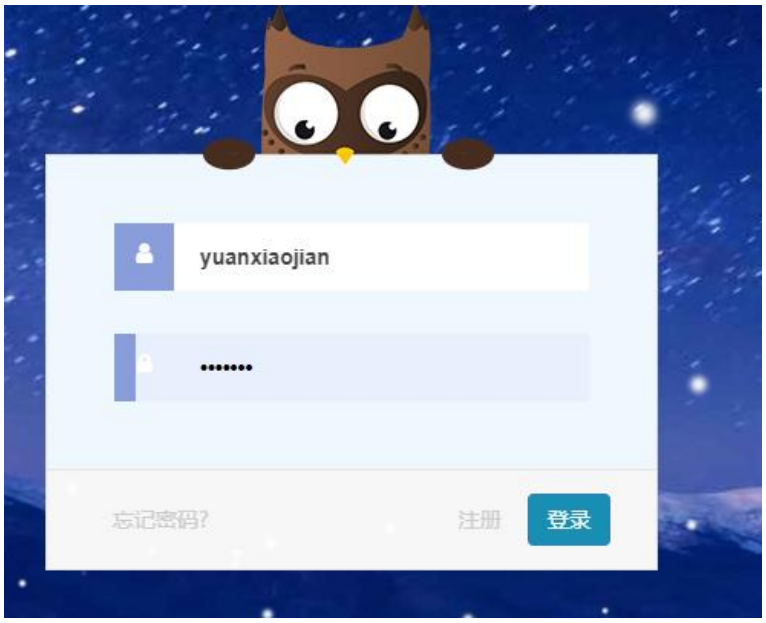

图 1-10 登录合法

- 非法时：

图 1-11 登录非法

（2）注册页面

使用 JavaScript 验证用户名和邮箱格式，并且确认两次密码输入相同，代码如下：

```
//注册
function register() {
    var username = $("#register-username").val(),
        password = $("#register-password").val(),
```

```
        repassword = $("#register-repassword").val(),
        code = $("#register-code").val(),
        flag = true,
        validatecode = null;
    //判断用户名密码是否为空
    if (username == "") {
        $.pt({
            target: $("#register-username"),
            position: 'r',
            align: 't',
            width: 'auto',
            height: 'auto',
            content: "用户名不能为空"
        });
        flag = false;
    }
    if (password == "") {
        $.pt({
            target: $("#register-password"),
            position: 'r',
            align: 't',
            width: 'auto',
            height: 'auto',
            content: "密码不能为空"
        });
        flag = false;
    } else {
        if (password != repassword) {
            $.pt({
                target: $("#register-repassword"),
                position: 'r',
                align: 't',
                width: 'auto',
                height: 'auto',
```

```
            content: "两次输入的密码不一致"
        });
        flag = false;
    }
}
//用户名只能是 15 位以下的字母或数字
var regExp = new RegExp("^[a-zA-Z0-9_]{1,15}$");
if (!regExp.test(username)) {
    $.pt({
        target: $("#register-username"),
        position: 'r',
        align: 't',
        width: 'auto',
        height: 'auto',
        content: "用户名必须为 15 位以下的字母或数字"
    });
    flag = false;
}
```

效果图如下：

- 合法时：

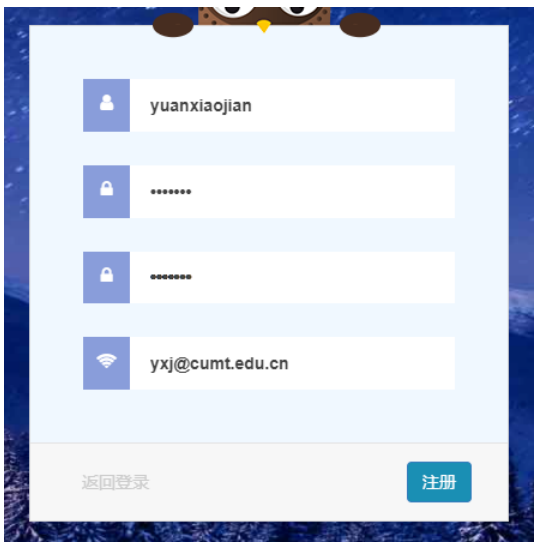

图 1-12 注册合法

- 邮箱格式非法：

图 1-13 注册邮箱非法

• 密码不一致：

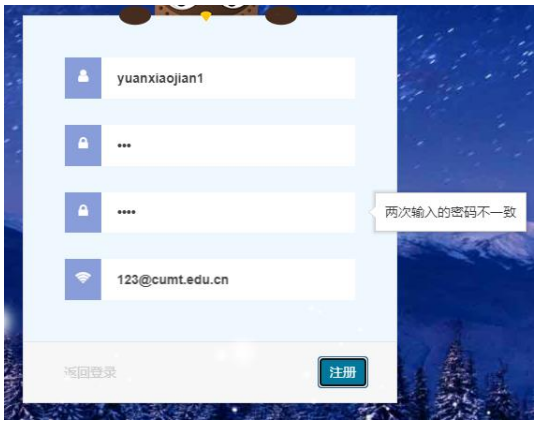

图 1-14 密码不一致

1.2.4 JavaScript 内置事件和事件

（1）HTML DOM 对象，header 部分根据登录情况动态生成用户信息，代码如下：

```
$("#exit").click(function () {
    alert(" 您确认要退出吗？ ");
    localStorage.setItem("into_flag", "0");
    location.reload();
});
if (localStorage.getItem("into_flag") == "0") {
    /* 如果退出状态，显示登录按键 */
    $("#exit").attr("style", "display:none;");
}
else {
    /* 如果登录状态，显示退出按键 */
    $("#login").attr("style", "display:none;");
    /* 获取用户名显示欢迎信息 */
    document.getElementById("info_user").innerHTML = " 欢迎你!
    "+localStorage.getItem("userName");
}
```

（2）使用 Onclick 事件进行登录表单的切换，即在登陆时点击“忘记密码”，由 onclick 事件触发 goto_forget()函数，从而切换到忘记密码表单样式，代码如下：

```
function goto_forget() {
    $("#forget-username").val("");
    $("#forget-password").val("");
    $("#forget-code").val("");
```

```
    $("#tab-3").prop("checked", true);
}

<!-- 省略 -->
<div class="form-actions">
    <a tabindex="4" class="btn pull-left btn-link text-muted" onclick="goto_forget()">忘记密码?</a>
    <input class="btn btn-primary" type="submit" tabindex="3" value="登 录 " style="color:white;">
</div>
<!-- 省略 -->
```

1.2.5 JavaScript 第三方库的使用

本次实验中，大量使用了 JavaScript 的第三方库 jQuery-3.4.1，在使用前首先需要在 html 文档开始处进行引用，且前面提到的前端框架 BootStrap 的使用也是需要引入 jQuery 的。

以电影分类页面为例，我们利用 jQuery 实现如下功能：

- 进入该类别时，该类别的标签为红色；
- 鼠标移动到其他类别上时，其他类别标签变为红色，移走时恢复；
- 可以对电影的年份和国家的进行筛选（使用 click 事件）

```
// 获得地址栏中 category 的值
var url = decodeURI(window.location.search);
var type = url.substring(url.lastIndexOf('=') + 1);
var country;
var year;
var score;

$(document).ready(function () {
    // 将对应种类的标签背景置红
    $("#menu a").each(function () {
        if ($(this).text() === type) {
            $(this).addClass("bg-danger");
        } else {
            $(this).removeClass("bg-danger");
        }
```

```
    });
    var indexs = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11];
    indexs = getRandom(indexs);
    console.log(indexs);
    $("#trans3DBoxes1").children().each(function (index) {
        var imgUrl = $(".card").find("img").eq(indexs[index + 6]).attr("src");
        if (imgUrl !== "") {
            console.log(imgUrl);
            $(this).css("background-image", "url(" + imgUrl + ")");
        }
    });
});

// 种类被点击时，更换按钮样式
$("#menu>a").click(function () {
    $("#menu a").removeClass("bg-danger");
    $(this).addClass("bg-danger");
});

//选择年份触发的点击事件
$("#year a").click(function () {
    year = $(this).text();
    $("#year a").first().text(year);
    $("#year a").first().toggleClass("bg-danger");
    if ($(this).hasClass("dropdown-item")) {
        //判断一下当前是第几页，如果不是第一页，从第一页开始
        $("#fenye li").children().filter(".myactive").removeClass("myactive");
        $("#fenye li").eq(1).children().addClass("myactive");
        conditionAjax(type, year, country, score, 1);
    }
});

//选择国家触发的点击事件
$("#country a").click(function () {
```

```
    country = $(this).text();
    $("#country a").first().text($(this).text());
    $("#country a").first().toggleClass("bg-danger");
    console.log($(this).text());
    if ($(this).hasClass("dropdown-item")) {
        //判断一下当前是第几页，如果不是第一页，从第一页开始
        $("#fenye li").children().filter(".myactive").removeClass("myactive");
        $("#fenye li").eq(1).children().addClass("myactive");
        conditionAjax(type, year, country, score, 1);
    }
});
```

效果如下：

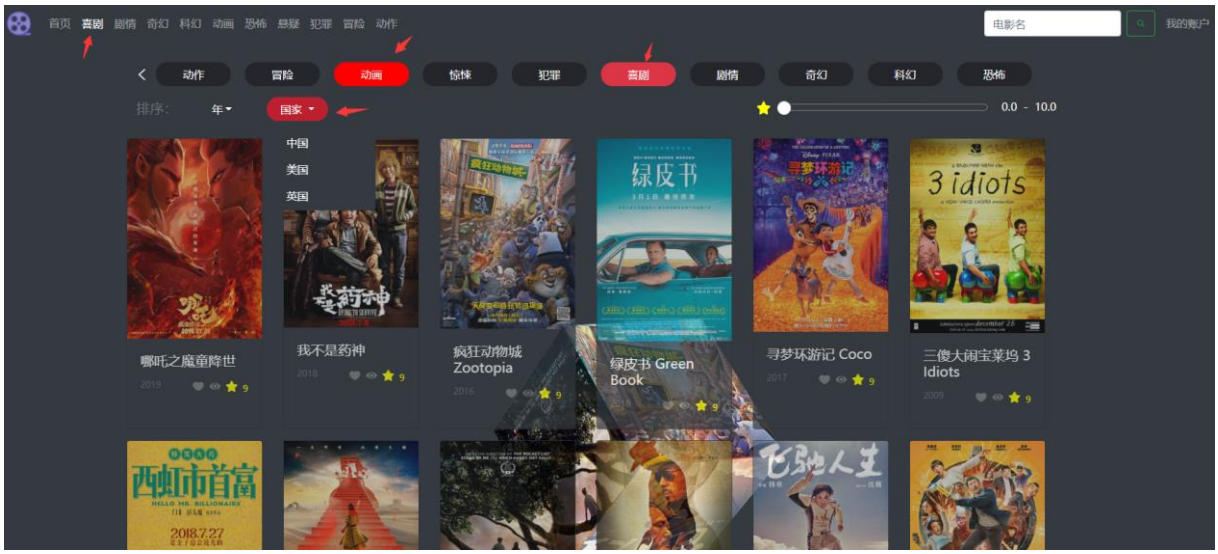

图 1-15 jQuery 示例

1.3 其它展示

1.3.1 旋转魔方

用前端技术在电影分类页面中心显示一个选择的正方体，其每个面由当前类别一个电影封面图组成，图 1-16 中也可看出效果，代码如下：

```
var trans3DDemo1 = $("#trans3DDemo1"), trans3DBoxes1 = $("#trans3DBoxes1"), boxes1
= $("#trans3DBoxes1 div"),
    threeDTimeline = new TimelineMax({onUpdate: updateCube, repeat: -1}), stageW
= ($(window).width()) / 2,
    stageH = ($(window).height()) / 2, stageX = (stageW - (trans3DBoxes1.width()
/ 2)), stageY = (stageH - (250 / 2));
```

```
TweenMax.set(trans3DBoxes1, {css: {transformPerspective: 3000, transformStyle:
"preserve-3d"}});
threeDTimeline.set(boxes1[0], {
    rotationX: 0,
    rotationY: 0,
    x: 0,
    y: 0,
    z: 125,
    opacity: 0.85
}).set(boxes1[1], {rotationX: 0, rotationY: -90, x: -125, y: 0, z: 0, opacity:
0.85}).set(boxes1[2], {
    rotationX: 0,
    rotationY: 90,
    x: 125,
    y: 0,
    z: 0,
    opacity: 0.85
}).set(boxes1[3], {rotationX: 90, rotationY: 0, x: 0, y: -125, z: 0, opacity:
0.85}).set(boxes1[4], {
    rotationX: -90,
    rotationY: 0,
    x: 0,
    y: 125,
    z: 0,
    opacity: 0.85
}).set(boxes1[5], {rotationX: 0, rotationY: 180, x: 0, y: 0, z: -125, opacity:
0.85}).set(trans3DBoxes1, {
    x: 150,
    y: 150,
    transformOrigin: "125px 125px 0px"
});
boxes1.each(function (f, e) {
    $(e).hover(h, g);
    function h() {
```

```
        TweenMax.to(e, 0.15, {opacity: 0.33})
    }
    function g() {
        TweenMax.to(e, 0.15, {opacity: 0.85})
    }
});
threeDTimeline.to(trans3DBoxes1, 15, {
    css: {rotationY: 360, rotationX: -720, transformOrigin: "125px 125px 0px"},
    ease: Power0.easeNone
});
function updateCube() {
    stageW = ($(window).width()) / 2;
    stageH = ($(window).height()) / 2;
    stageX = (stageW - (trans3DBoxes1.width() / 2));
    stageY = (stageH - (250 / 2));
    TweenMax.to(trans3DBoxes1, 1, {css: {x: stageX, y: stageY}})
};
```

其 css 代码如下：

```
#trans3DDemo1 a {
    color: rgba(255, 255, 255, 0.6);
    outline: none;
    text-decoration: none;
    -webkit-transition: 2s;
    transition: 4s;
}
#trans3DDemo1 a:hover,
a:focus {
    color: #74777b;
    text-decoration: none;
}
.htmleaf-container {
    margin: 0 auto;
}
#trans3DDemo1 {
```

```
    position: absolute;
    display: inline-block;
    width: 250px;
    height: 250px;
    top: 50px;
    left: 50px;
    /*top:250px;   margin:0px auto 0px auto;*/
}
#trans3DBoxes1 div {
    position: absolute;
    width: 248px;
    height: 248px;
    padding: 0;
    margin: 0;
    border: 1px solid rgba(255, 255, 255, .2);
    display: block;
    text-align: center;
    font-size: 36px;
    font-weight: bold;
}
```

效果图如下（仅保留魔方后）：

图 1-16 旋转魔方效果图

1.3.2 电影分页

该功能需要后续进一步完善网站后才可添加，但由于主要涉及前端技术，所以放在报告中实验一的位置，即利用 jQuery 和 ajax 技术对电影类别页面进行分页，代码如下：

```
var $fenye = $("#fenye").children();


$fenye.click(function () {
    var number=0;
    var hideCount = 0;
    $fenye.each(function () {
        // 如果 li 标签没有这个 class 属性，则代表是可见的，则长度加 1
        if(!$(this).hasClass("d-none")){
            number += 1;
        }else{
            hideCount += 1;
        }
    });
    // console.log($(this).index());
    var index = $(this).index();
    if(index > number){
        index = $(this).index() - hideCount;
    }
    var $now = $fenye.children().filter(".myactive");
    // console.log(index)
    // 左边界
    var left = $now.parent().prev().index();
    // 右边界
    var right = $now.parent().next().index();
    // console.log("left: " + left + " right:" + right)


    // 左右箭头
    if (index === 0 && left !== 0) {
        $now.removeClass("myactive");
        $now.parent().prev().children().addClass("myactive");
        nextData($now.parent().prev().index())
```

```
    } else if (index === number - 1 && right !== number - 1) {
        $now.removeClass("myactive");
        $now.parent().next().children().addClass("myactive");
        nextData($now.parent().next().index())
    }
    if (index > 0 && index < number - 1) {
        $(this).siblings().children().removeClass("myactive");
        $(this).find("a").addClass("myactive");
        nextData(index)
    }
    // 控制左右箭头是否可点击
    $now = $fenye.children().filter(".myactive");
    // 左边界
    left = $now.parent().prev().index();
    // 右边界
    right = $now.parent().next().index();
    console.log("index:" + index + " number: " + number + " right:" + right);
    if (index === 1 || left === 0) {
        $("#leftArr").addClass("disabled");
    } else {
        $("#leftArr").removeClass("disabled");
    }
    if (index === number - 2 || right === number - 1 || index === number - 1) {
        $("#rightArr").addClass("disabled");
    } else {
        $("#rightArr").removeClass("disabled");
    }
});

// 选择下一页时触发的 ajax 事件
function conditionAjax(type, year, country, score, index) {
    var dataUrl;
    // 如果年份、分数、国家都没选择
    if (typeof (year) == "undefined" && typeof (country) == "undefined" && typeof
```

```
(score) == "undefined") {
        dataUrl = "page=" + index + "&type=" + type + "&choose=none";
    } else if (typeof (year) !== "undefined" && typeof (country) == "undefined"
&& typeof (score) == "undefined") {
        // 如果只选择了年份
        dataUrl = "page=" + index + "&type=" + type + "&year=" + year + "&choose=Y";
    } else if (typeof (year) == "undefined" && typeof (country) !== "undefined"
&& typeof (score) == "undefined") {
        // 如果只选择了国家
        dataUrl = "page=" + index + "&type=" + type + "&country=" + country +
"&choose=G";
    } else if (typeof (year) == "undefined" && typeof (country) == "undefined" &&
typeof (score) !== "undefined") {
        // 如果只选择了分数
        dataUrl = "page=" + index + "&type=" + type + "&score=" + score + "&choose=S";
    } else if (typeof (year) !== "undefined" && typeof (country) !== "undefined"
&& typeof (score) == "undefined") {
        // 如果选择了年份和国家
        dataUrl = "page=" + index + "&type=" + type + "&year=" + year + "&country="
+ country + "&choose=YG";
    } else if (typeof (year) !== "undefined" && typeof (country) == "undefined"
&& typeof (score) !== "undefined") {
        // 如果选择了年份和分数
        dataUrl = "page=" + index + "&type=" + type + "&year=" + year + "&score="
+ score + "&choose=YS";
    } else if (typeof (year) == "undefined" && typeof (country) !== "undefined"
&& typeof (score) !== "undefined") {
        // 如果选择了国家和分数
        dataUrl = "page=" + index + "&type=" + type + "&score=" + score + "&country="
+ country + "&choose=GS";
    } else {
        // 如果三者都选择了
        dataUrl = "page=" + index + "&type=" + type + "&year=" + year + "&country="
+ country + "&score=" + score + "&choose=YGS";
```

```
    }
    // console.log(dataUrl);

    $.ajax({
        url: "updateData",
        data: dataUrl,
        success: function (data) {
            var arr = data.split('|');
            // 符合条件的电影数量
            var length = arr[1];
            var pageCount = Math.ceil(length / 12);
            console.log("pageCount: " + pageCount);
            // 获取总的页数，改变页面导航栏
            $("#fenye li").each(function (index) {
                console.log("显示/隐藏：" + index);
                // 如果当前的 li 标签索引大于要显示的页数，则隐藏，否则显示
                if (index > pageCount && index !== $("#fenye li").length - 1) {
                    $(this).addClass("d-none");
                }else{
                    $(this).removeClass("d-none");
                }
            });
            var json = eval(arr[0]);
            // console.log(json);
            update(json);
        }
    });
}
```

效果图如下：

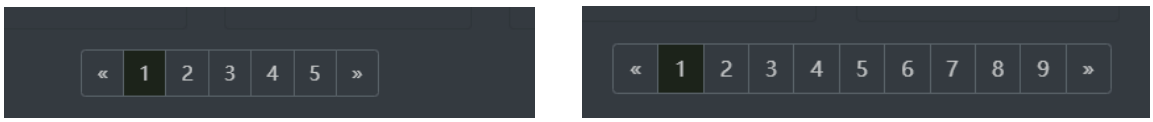

图 1-17 分页效果图

1.4 设计心得

经过本次实验，我对网页前端设计流程有了进一步的深入，并根据自己的喜好，以简洁、美观为原则进行开发与布局。除此之外，深刻意识到了善于利用现有的前端框架的重要性，相比与从零开始手写 CSS 和 JavaScript 代码，利用开源框架中现有的组件、布局等，可快速实现所构思页面的设计与布局。在面对不同类型设备时，还可以通过框架快速实现响应式布局，省去了传统的媒体查询工作。页面布局要尽量简单明了，拒绝层层嵌套的布局方式，这样不利于后期的维护调整。总的来说，这次的实验还是使我的前端开发技术有了大大的提升，但在细节方面还有待提高，以后需要更加注意。

实验二 动态 Web 页面设计

2.1 实验说明

2.1.1 实验目的

1、熟悉 JSP 的开发工具，掌握服务器端 Web 程序的工作原理
2、熟悉 JSP 编译指令，动作标记
3、熟悉 JSP 的隐含对象，正确理解 request、session、application 三个对象的作用域
4、掌握编写 JavaBean 的方法，使用 JSP <jsp:useBean>、<jsp:set Property>、<jsp:getProperty>3 个动作指令。
5、掌握 JSP 中表单和表单 Bean 的映射

2.1.2 实验要求

1、实验之前认真查阅相关资料，准备好实验方案。
2、认真实验，对实验过程、结果进行分析，注意验证实验效果。
3、完成实验报告和实验成果

2.2 实验内容

2.2.1 开发环境与开发工具

开发环境：jdk1.8、Tomcat9.0.26、MySQL 5.7.26
开发工具：IntelliJ IDEA Professional

2.2.2 网站结构

将修改实验一中的静态页面，改为 JSP 页面后网站结构如下图所示：

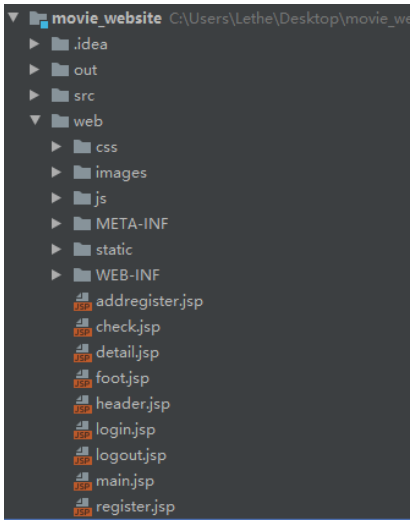

图 2-1 结构图

2.2.3 JavaBean 验证

新建 addregister.jsp 作为注册结果页面,通过 useBeen 和 setProperty 建立 userinfo 对象,并设置该对象的属性为提交上来的表单的属性,property="*"可以将表单中同名称的属性值赋值给 javaBean 对象中的同名属性.代码如下：

```
public class User {
    private int id;
    private String username;
    private String password;
    private String account;

    public String getAccount() {
        return account;
    }
    public void setAccount(String account) {
        this.account = account;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
```

```
    @Override
    public String toString() {
        return "User{" +
                "id=" + getId() +
                ", username='" + getUsername() + '\'' +
                ", password='" + getPassword() + '\'' +
                '}';
    }
}
```

jsp 页面使用的代码如下，将获得的 username 与 password 存储在 userinfo 对象中，并保存到 session 中，键值为 register_use：

```
<%
    request.setCharacterEncoding("utf-8");
%>
<jsp:useBean id="user" class="model.entity.User" scope="page">
    <jsp:setProperty property="username" name="user"/>
    <jsp:setProperty property="password" name="user"/>
</jsp:useBean>
<%
    User register_user = new User();
    register_user.setUsername(user.getUsername());
    register_user.setPassword(user.getPassword());
    session.setAttribute("register_user", register_user);
//System.out.println(register_user);
    response.sendRedirect("./login.jsp");
%>
```

2.2.4 request 对象

在 login.jsp 页面使用了 request 对象实现参数获取，如果 “error”不为空，则代表上次 check.jsp 验证登录失败，同时 number,记录失败次数代码如下：

```
<%
    Object obj = request.getAttribute("error");
    Integer sum = Integer.parseInt(session.getAttribute("number"));
    if (obj != null) {%>
<script>
    $(function () {
```

```
        document.getElementById("result").innerHTML = " 错误 "
    })
</script>
<%if (sum == 1) {%>
<script>
    $(function () {
        alert(" 密码错误，还有 2 次机会 ")
    })
</script>
<%} else if (sum == 2) {%>
<script>
    $(function () {
        alert(" 密码错误，还有 1 次机会 ")
    })
</script>
<%} else {%>
<script>
    $(function () {
        alert(" 密码错误，冻结 ")
    })
</script>
<%}%>
```

负责验证的 check.jsp 代码如下，从 session 中取得 register_user，保存了注册的用户名和密码，与输入的比较，正确则跳转；错误，用 session 的 number 记录错误：

```
<%
    String username = request.getParameter("loginusername");
    String password = request.getParameter("loginpassword");
    User user = (User) session.getAttribute("register_user");
    System.out.println(user);
    if                  (username.equals(user.getUsername())                &&
password.equals(user.getPassword())) {
        User user_info = new User();
        user_info.setUsername(username);
        user_info.setPassword(password);
        session.setAttribute("user_info", user_info);
        response.setHeader("refresh", "1;../../index.jsp");
    } else {
%>
<%
    int number = 1;
// 从 session 对象获取 number
    Object obj = session.getAttribute("number");
```

```
    if (obj == null) {
// 设定 session 对象的变量的值
        session.setAttribute("number", String.valueOf(number));
    } else {
// 取得 session 对象中的 number 变量
        number = Integer.parseInt(obj.toString());
// 统计错误次数
        number += 1;
// 设定 session 对象的 number 变量值
        session.setAttribute("number", String.valueOf(number));
        if (number >= 3) {
            response.setHeader("refresh", "2;url='./register.jsp");
        }
    }
    request.setAttribute("error", " 错误 ");
%>
<jsp:forward page="login.jsp"></jsp:forward>
<%}%>
```

效果图如下：

图 2-2 登录错误效果展示

2.2.5 response 对象

在 logout.jsp 页面使用了 response 对象实现响应头设置,实现 2 秒后跳到登录页面，代码如下：

```
<%
    session.invalidate();
    response.setHeader("refresh","2;url='./login.jsp'");
%>
```

2.2.6 Session 对象

在 check.jsp 页面使用了 session 对象实现登录,代码如下：

```
User user_info=new User();
user_info.setUsername(username);
user_info.setPassword(password);
session.setAttribute("user_info",user_info);
```

在 out.jsp 页面使用了 session 对象实现注销,代码如下:

```
<%session.invalidate();
    response.setHeader("refresh","2;url='./signin.jsp'");
%>
```

2.2.7 客户端跳转与服务器端跳转

在 addregister.jsp 页面实现了客户端跳转，代码如下：

```
<%
    User register_user = new User();
    register_user.setUsername(user.getUsername());
    register_user.setPassword(user.getPassword());
    session.setAttribute("register_user", register_user);
    response.sendRedirect("./signin.jsp");
%>
```

在 check.jsp 页面实现了服务器端跳转，使用 request 对象传递登录结果，代码如下：

```
request.setAttribute("error"," 错误 ");
<jsp:forward page="signin.jsp"></jsp:forward>
```

2.3 其它展示

由于本网站不仅仅需要提高普通用户的注册、登录、访问功能，还需要设置管理员页面来对网站整体进行管理。

我们设管理员的默认用户名为 admin，为了开发方便，使 admin 和普通用户使用同一个登录页面，但是根据用户的不同，利用 session 来进行区分，在登陆后分别跳转向不同的页面，即 admin 用户登陆后跳转向后台管理页面，而普通用户在登陆后则跳转向首页（此时为登录状态），部分代码如下：

```
<%
    User user = (User) session.getAttribute("user");
    if (user != null) {
%>
```

```
<c:if test="${user.username == \"admin\"}">
    <a href="management/index.jsp" onclick="jump()">
            ${user.username }
    </a>
</c:if>
<c:if test="${user.username != \"admin\"}">
    <a class="nav-link dropdown-toggle" href=""
       id="navbardrop" data-toggle="dropdown">
            ${user.username }
    </a>
</c:if>
<%
} else {
%>
<a href="${pageContext.request.contextPath}/loginOrRegister.do">我的账户</a>
<%
    }
%>
```

效果图如下：

- admin 登陆后，跳转至管理后台：

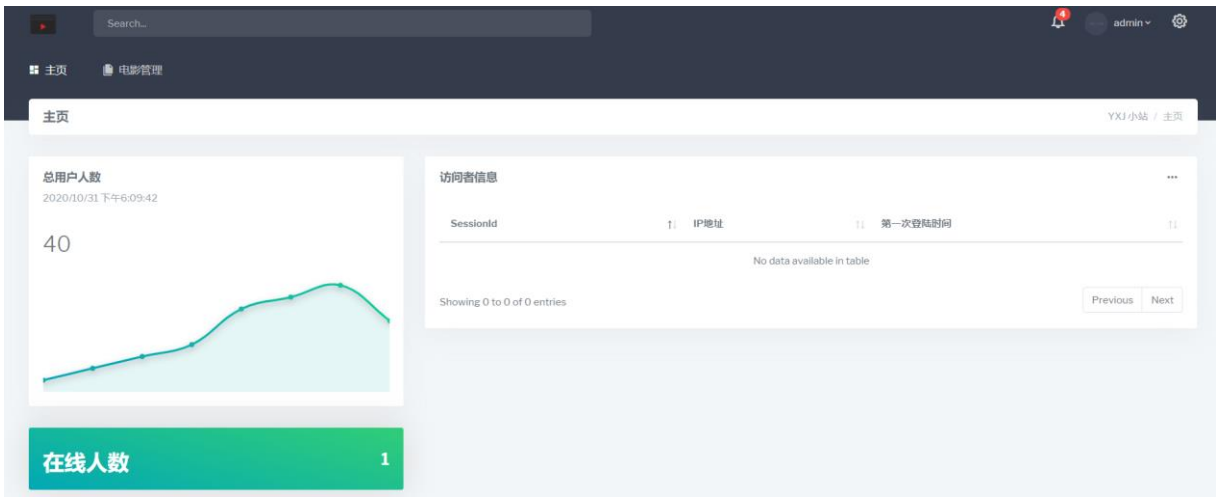

图 2-3 跳转至管理后台

- 普通用户登陆后，以登录状态跳转至首页：

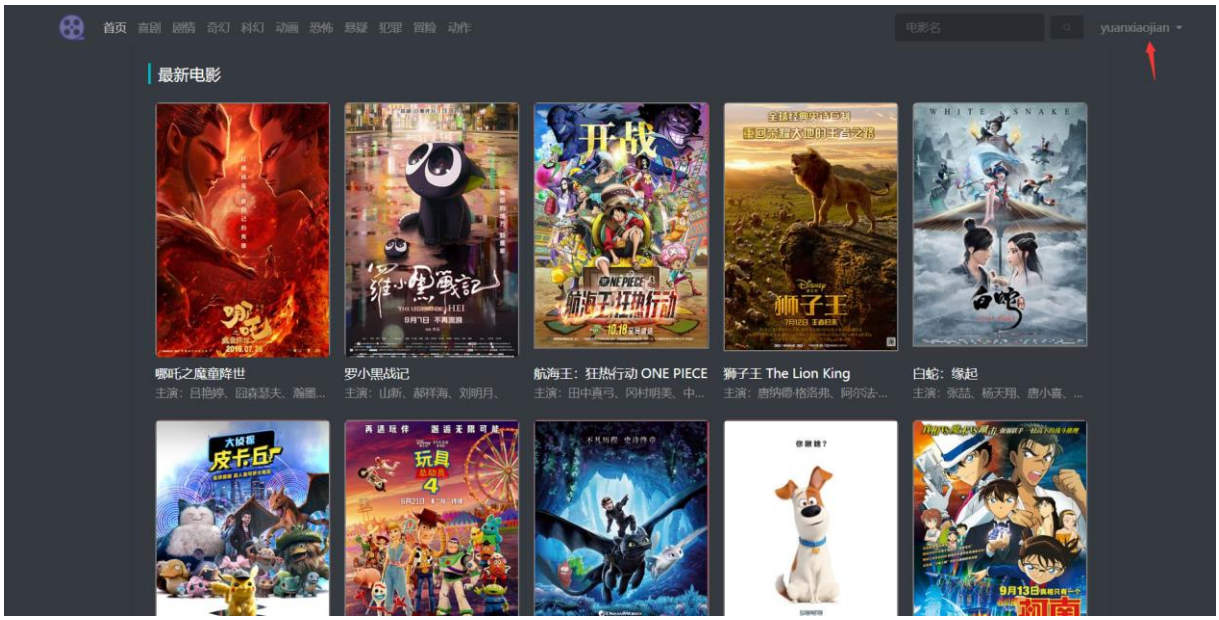

图 2-4 跳转至首页

2.4 设计心得

经过本次实验，将实验一中的静态页面成功改写成了 JSP 动态页面，在这过程中也使我进一步熟悉了 JSP 的开发工具，并掌握了服务器端 Web 程序的工作原理。除此之外，我也不仅仅掌握了如何去开发，也从原理上熟悉了 JSP 的隐含对象，进一步理解 request、session、application 对象的作用域。并在此基础上，我还掌握了 JavaBean 的编写方法，以及对 JSP 动作指令的使用方法，实践了 JSP 中表单和表单 Bean 的映射。总的来说，通过这次实验，成功的在静态网页的基础上，实现了一些动态功能，使我开发的电影网站更加接近真实的网站，但这种开发方式并不适合大型网站的开发，在之后的实验过程中还需要进一步的改进。

实验三 Web 数据库开发实验

3.1 实验说明

3.1.1 实验目的

1、掌握 Servlet 的开发、配置
2、掌握 Filter 的开发与配置
3、熟悉 JDBC 以及 DAO 的概念及工作原理
4、能够熟练运用 Servlet+DAO 模式对数据库进行访问，实现数据查询、添加、修改等常用操作
5、能够熟练运用 MVC 模式开发 Web 应用程序，实现数据查询、添加、修改等常用操作。

3.1.2 实验要求

1、实验之前认真查阅相关资料，准备好实验方案。
2、认真实验，对实验过程、结果进行分析，注意验证实验效果。
3、完成实验报告和实验成果

3.1.3 实验内容

1、请设计一种过滤器实现权限控制机制。如果用户进入 Web 应用没有登录时，要求用户必须进行登录页面。
2、使用 JDBC 技术进行数据库的连接与访问。
3、采用 DAO 设计模式开发。使用自己熟悉的网络数据库，设计一个小的系统（可以在之间的页面上实现），要求如下：
1）至少包括 2 张数据表（字段数不少于 4）；
2）实现对 2 张表格的增、删、改、查操作。
3）具有不同类型用户的登录控制页面。
4）要求每一项操作对应与一张页面。

3.2 详细设计与编码

3.2.1 设计数据库表格

根据本网站的需求，需要用到数据库交互的有如下功能有：用户注册、登录功能，电影相关信息的存储，用户与电影之间的交互(浏览记录、收藏、评论)。因此，又这些需求设计数据库如下：
（1）users 表

- 作用：存储网站用户的账号信息。

- 字段：id（主键），username（用户名），password（密码），gender（性别），email（邮箱），telephone（电话），introduce（介绍），state（状态），role（角色），registTime（最近登陆时间）
- 效果图如下：

id	username	password	gender	email	telephone	introduce	state	role	registTime
1	admin	123456	男	945716994@qq.com	18770411594	管理员	1	admin	2019-10-24 11:12:31
10	test	test	null	(Null)	null	这家伙很懒，还没有添加任何描述	1	commonUser	2020-10-29 00:14:08
12	袁李健	yuan123	null	(Null)	null	这家伙很懒，还没有添加任何描述	1	commonUser	2020-10-30 15:12:47
13	yuanxiaojian	yuan123	null	(Null)	null	这家伙很懒，还没有添加任何描述	1	commonUser	2020-10-30 15:13:10

图 3-1 users 表

（2）allmovies 表

- 作用：存储网站中电影的相关信息。
- 字段：id（主键），name（电影名），score（评分），director（导演），scriptwriter（编剧），actor（演员），year（年份），country（国家），languages（语言），length（片长），image（海报），type（类型），url（链接），des（介绍）
- 效果图如下：

id	name	score	director	scriptwriter	actor	years	country	languages	length	image	des	url	type
1	哪吒之魔童降世	9	饺子、	饺子、易巧、魏芸芸、	吕艳婷、邱	2019	中国大陆	汉语普通话	110分钟	donghua//1.jpg		https://www.douban.com/	动画
2	罗小黑战记	8	木头、	木头、	山新、郝祥	2019	中国大陆	汉语普通话	101分钟	donghua//2.jpg		https://www.douban.com/	动画
3	航海王：狂热行动 ONE PIE	8	大塚隆史、	尾田荣一郎、大塚隆史、	富[田中真弓、	2019	日本	日语	101分钟	donghua//3.jpg		(Null)	动画
4	千与千寻 千と千尋の神隠し	9	宫崎骏、	宫崎骏、	柊瑠美、入	2001	日本	日语	125分钟	donghua//4.jpg		(Null)	动画
5	疯狂动物城 Zootopia	9	拜伦·霍华德、瑞奇·摩尔、杰	拜伦·霍华德、瑞奇·摩尔、杰·金妮弗·古[		2016	美国	英语 / 挪威语	109分钟(中国大陆)	donghua//5.jpg		https://www.douban.com/	动画
6	寻梦环游记 Coco	9	李·昂克里奇、阿德里安·莫利	阿德里安·莫利纳、马修·奥尔	安东尼·冈[	2017	美国	英语 / 西班牙语	105分钟	donghua//6.jpg		https://www.douban.com/	动画

图 3-2 allmovies 表

（3）collection 表

- 作用：存储用户与其收藏电影的对应关系。
- 字段：userName（用户名）、movieName（电影名）、addTime（收藏时间）
- 效果图如下：

userName	movieName	addTime
yuanxiaojian	双子杀手 Gemini Man	2020-10-31 11:32:51
yuanxiaojian	哪吒之魔童降世	2020-10-31 11:32:52

图 3-3 collection 表

（4）comments 表

- 作用：存储用户对于电影的评论信息及对应关系。
- 字段：userName（用户名），movieName（电影名）、description（评论），addTime（评论时间）
- 效果图如下：

userName	movieName	description	addTime
yuanxiaojian	哪吒之魔童降世	[嘻嘻][哆啦A梦微笑]	2020-10-31 21:54:38
gg	哪吒之魔童降世	[黑穿妇][蜘蛛侠]蜘蛛侠Spid	2020-10-31 22:02:28
gg	哪吒之魔童降世	你们好2333[伤心]	2019-12-05 22:13:05

图 3-4 comments 表

（5）history 表

- 作用：存储用户对电影页面的访问记录。
- 字段：movieName（电影名），userid（用户 id），addTime（访问时间）

- 效果图如下：

movieName	userid	addTime
航海王：狂热行动 ONE PIE	13	2020-10-31 16:10:42
哪吒之魔童降世	1	2020-10-31 15:31:53
罗小黑战记	13	2020-10-31 11:32:33
千与千寻 千と千尋の神隠し	10	2020-10-29 16:32:10
哪吒之魔童降世	10	2020-10-29 16:31:59

图 3-5 history 表

3.2.2 连接数据库

本项目中使用了 c3p0 对数据进行操作，c3p0 是一个开源的 JDBC 连接池，它实现了数据源和 JNDI 绑定，支持 JDBC3 规范和 JDBC2 的表征，其具有编码简单易用、连接复用、连接管理等特性，使用它可以大大提高应用程序和数据库之间访问效率的。

使用 c3p0 首先需要下载其包文件，并导入项目中，然后在 src 目录下建立 c3p0-config.xml 文件，内容为数据库的一些配置信息，如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<c3p0-config>
    <default-config>
        <property name="driverClass">com.mysql.jdbc.Driver</property>
        <property
name="jdbcUrl">jdbc:mysql://localhost:3306/moviesdata?serverTimezone=Asia/Shan
ghai</property>
        <property name="user">root</property>
        <property name="password">yuan123</property>
    </default-config>
</c3p0-config>
```

然后在 src 的 utils 目录下创建一些数据库源工具，如连接、开启事务、提交事务、事务回滚、结束事务等操作，如下：

```
package utils;


import com.mchange.v2.c3p0.ComboPooledDataSource;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;


/**
```

```
 * 数据源工具
 */
public class DataSourceUtils {
    /**
     * 创建数据源（c3p0)
     */
    private static DataSource dataSource = new ComboPooledDataSource();
    /**
     * ThreadLocal: 可以为每个线程创建一个副本。每个线程可以访问自己内部的副本
     */
    private static ThreadLocal<Connection> tl = new ThreadLocal<Connection>();

    public static DataSource getDataSource() {
        return dataSource;
    }

    /**
     * 当 DBUtils 需要手动控制事务时，调用该方法获得一个连接
     *
     * @return
     * @throws SQLException
     */
    public static Connection getConnection() throws SQLException {
        // 获得一个连接
        Connection con = tl.get();
        // 判断是否第一次取连接，第一次取连接时，ThreadLocal 中没有连接
        if (con == null) {
            // 如果是第一次取连接，则这个连接为空，创建一个连接
            con = dataSource.getConnection();
            tl.set(con);
        }
        return con;
    }
```

```
/**
 * 开启事务
 *
 * @throws SQLException
 */
public static void startTransaction() throws SQLException {
    Connection con = getConnection();
    if (con != null) {
        // 开启事务,true 表示自动提交事务
        con.setAutoCommit(false);
    }
}

/**
 * 提交事务
 *
 * @throws SQLException
 */
public static void commitTransaction() throws SQLException {
    Connection con = getConnection();
    if (con != null) {
        con.commit();

    }
}

/**
 * 事务回滚
 *
 * @throws SQLException
 */
public static void rollback() throws SQLException {
    Connection con = getConnection();
    if (con != null) {
```

```
            con.rollback();
        }
    }

    /**
     * 从 ThreadLocal 中释放并且关闭 Connection,并结束事务
     *
     * @throws SQLException
     */
    public static void close() throws SQLException {
        Connection connection = getConnection();
        if (connection != null) {
            // 清空 ThreadLocal
            tl.remove();
            // 关闭连接
            connection.close();
            connection = null;
        }
    }
}
```

然后再 DAO 层编写通用的增删改查的方法，并导入该工具类，先调用其中的方法，再对数据库中具体的数据进行操作即可。以查找所有电影的方法 findAllMovies 为例，其代码如下：

```
public class MovieDao {
    /**
     * @return 所有电影的集合
     * @Description: TODO(查找所有电影)
     */
    public List<Movie> findAllMovies() throws SQLException {
        String sql = "select * from allmovies GROUP BY name";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        return runner.query(sql, new BeanListHandler<>(Movie.class));
    }
    //……省略……
```

另外，本项目对每一个数据表建立了相应的 bean 类，其中定义了数据表中的字段属性以及方法，然后建立对应的 DAO 类进行具体数据的操作，这样可以大大减少重复代码的定义，而且调用及其简单。

以对用户对电影的访问记录为例，其 bean 类 History.java 如下：

```
package domain;

import java.util.Date;

/**
 * @ClassName: History.java
 * @Description: 对应数据库中的 history 表，记录用户的观看历史
*/
public class History {
    /**
     * 用户观看过的电影 id
     */
    private String movieName;
    /**
     * 用户 id
     */
    private int userId;

    /**
     * 添加历史记录的时间
     */
    private Date addTime;

    public Date getAddTime() {
        return addTime;
    }

    public void setAddTime(Date addTime) {
        this.addTime = addTime;
    }
```

```
    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getMovieName() {
        return movieName;
    }

    public void setMovieName(String movieName) {
        this.movieName = movieName;
    }

    @Override
    public String toString() {
        return "History{" +
                "movieName='" + movieName + '\'' +
                ", userId=" + userId +
                ", addTime=" + addTime +
                '}';
    }
}
```

其对应的 DAO 类 HistoryDao.java 代码如下：

```
package dao;

import domain.History;
import domain.Movie;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.ArrayListHandler;
import org.apache.commons.dbutils.handlers.BeanListHandler;
```

```
import utils.DataSourceUtils;

import java.sql.SQLException;
import java.util.List;

public class HistoryDao {



    /**
     * @param userId    用户 id
     * @param movieName 电影名
     * @throws SQLException
     * @Description: 添加一条浏览记录，如果存在则忽略
     */
    public void addRecord(int userId, String movieName) throws SQLException {
        String sql = "INSERT IGNORE INTO history (movieName, userid) values(?,?)
";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        runner.update(sql, movieName, userId);

    }



    public List<History> getRecords(int userId) throws SQLException {
        String sql = "select * from history where userid = ? ORDER BY addTime desc";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        return runner.query(sql, new BeanListHandler<>(History.class), userId);
    }
}
```

3.2.3 Filter 实现—登录权限控制机制

在项目中实现了对于登录请求的过滤器 LoginFilter，代码如下：

```
/**
 * 专门拦截登录请求的过滤器
 */
```

```
@WebFilter(filterName = "LoginFilter", urlPatterns = "/login.do")
public class LoginFilter implements Filter {
    static Logger logger = Logger.getLogger(LoginFilter.class);
    static String flag = "LoginFilter.............";

    @Override
    public void destroy() {
    }

    @Override
    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain
chain) throws ServletException, IOException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) resp;

        String username = request.getParameter("username");
        String password = request.getParameter("password");
        UserService service = new UserService();

        logger.warn(flag + username + " " + password);
        try {
            if (username != null && !"".equals(username)) {
                User user = service.login(username, password);
                if(user != null){
                    logger.warn(flag + " " + user.getId());
                    // 获得用户 id 对象的 session 对象
                    HttpSession                    session                    =
MySessionContext.getSession(String.valueOf(user.getId()));
                    // 如果 session 对象不为空，则代表已经存在 session 对象，即已经有人
登录过了
                    if (session != null) {
                        logger.warn(flag + "删除 seesion");
                        // 将这个 session 对象删除
                        MySessionContext.delSession(session);
```

```
                }
            }
        }
    } catch (LoginException e) {
        e.printStackTrace();
    }
    // 放行
    chain.doFilter(request, response);
}

@Override
public void init(FilterConfig config) throws ServletException {

}

}
```

然后,在 web.xml 文件中配置 LoginFilter:

```
<filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>filter.LoginFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>LoginFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

图 3-6 配置 filter

3.2.4 users 表和 collection 表的增删查改

（1）users 表的操作代码如下:

```
package dao;

import domain.User;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanHandler;
import utils.DataSourceUtils;
```

```
import java.sql.SQLException;

/**
 * @ClassName: UserDao.java
 * @Description: 用于操纵数据库，对 user 表进行增删改查
 */
public class UserDao {

    /**
     * @return User
     * @throws SQLException 参数
     * @throws SQLException
     * @Description: 根据用户名和密码查找用户
     */
    public User findUserByUsernameAndPassword(String username, String password)
throws SQLException {
        String sql = "select * from users where username=? and password=?";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        return runner.query(sql, new BeanHandler<User>(User.class), username,
password);
    }

    /**
     * @param user 被添加的用户对象
     * @throws SQLException 参数
     * @Description: 添加用户
     */
    public void addUser(User user) throws SQLException {
        String sql = "insert into users(username,password,email) values(?,?,?)";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        int row = runner.update(sql, user.getUsername(), user.getPassword(),
user.getEmail());
        if (row == 0) {
```

```
            throw new RuntimeException();
        }
    }

    /**
     * @param user 更新的用户信息
     * @Description:更新用户信息
     */
    public void updateUser(User user) throws SQLException {
        String sql = "UPDATE users SET introduce = ?, password = ? WHERE id = ?";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        runner.update(sql,       user.getIntroduce(),       user.getPassword(),
user.getId());
    }

    /**
     * 根据用户名查找用户
     *
     * @param userName 用户名
     * @throws SQLException
     */
    public User findUserByUserName(String userName) throws SQLException {
        String sql = "select * from users where userName = ?";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        return runner.query(sql, new BeanHandler<>(User.class), userName);
    }

}
```

在注册页面添加一个新用户 users_test:

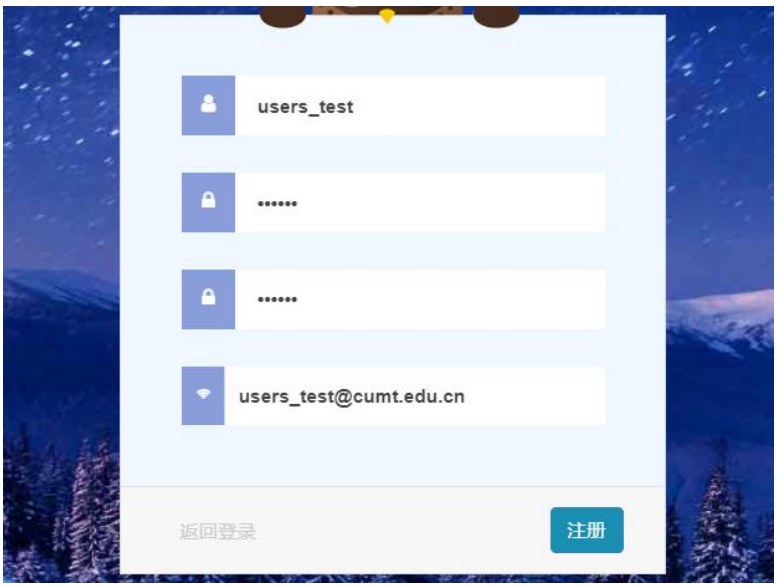

图 3-7 添加新用户 users_test

查看数据库，已添加了该用户：

id	username	password	gender	email	telephone	introduce
1	admin	123456	男	1013579926@qq.com	17851146833	管理员
12	袁孝健	yuan123	null	123@qq.com	null	这家伙很懒，还没有添加任何描述
13	yuanxiaojian	yuan123	null	yuanxiaojian@cumt,edu.cn	null	这家伙很懒，还没有添加任何描述
18	users_test	123456	null	users_test@cumt.edu.cn	null	这家伙很懒，还没有添加任何描述

图 3-8 成功添加用户

登录账号后，在个人界面可以修改简介和密码，也就是数据库中的 introduce 字段值，如下：

图 3-9 修改用户简介

该页面 Servlet 代码如下：

```
public class AlterUserInfoServlet extends HttpServlet {
    static Logger logger = Logger.getLogger(AlterUserInfoServlet.class);
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        /*request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        response.setCharacterEncoding("utf-8");*/

        // 获得要设置的新密码
        String introduce = request.getParameter("introduce");
        String password = request.getParameter("password");

        logger.warn(introduce);
        User user = (User) request.getSession().getAttribute("user");
        if (password != null && !"".equals(password)) {
            // 将这个新密码存到 User 对象中
            user.setPassword(password);
        }
        if (introduce != null) {
            user.setIntroduce(introduce);
        }
        UserService service = new UserService();
        PrintWriter writer = response.getWriter();
        try {
            service.updateUser(user);
            writer.write("ok");

        } catch (UpdateUserException e) {
            e.printStackTrace();
            writer.write("false");
        }
    }
```

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

修改后，查看数据库变化，如下：

id	username	password	gender	email	telephone	introduce
1	admin	123456	男	1013579926@qq.com	17851146833	管理员
12	袁孝健	yuan123	null	123@qq.com	null	这家伙很懒，还没有添加任何描述
13	yuanxiaojian	yuan123	null	yuanxiaojian@cumt,edu.cn	null	这家伙很懒，还没有添加任何描述
18	users_test	123456	null	users_test@cumt.edu.cn	null	修改users表中的个人简介。

图 3-10 introduce 字段修改

（2）collection 表

实现用户收藏电影页面操作的 Servlet 代码如下：

```
public class CollectionServlet extends HttpServlet {
    private Logger logger = Logger.getLogger(CollectionServlet.class);

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String movieName = request.getParameter("movieName");
        String userName = request.getParameter("userName");
        logger.warn(movieName + "   " + userName);

        CollectionService service = new CollectionService();
        // 如果只传递过来用户名,表示获取用户的收藏电影记录
        if (movieName == null || "".equals(movieName)) {
            try {
                List<Collection>                    list                    =
service.findAllRecordByUserName(userName);
                logger.warn("collection-->" + list.size());
                JSONArray                       jsonArray                   =
JSONArray.parseArray(JSON.toJSONString(list));
                String result = jsonArray.toJSONString();
                // 获取输出流
                PrintWriter writer = response.getWriter();
```

```
                writer.write(result);
                writer.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        } else {
            // 如果电影名和用户名都传递过来，则表示添加一条用户收藏记录
            try {
                service.addUserCollectionMovie(userName, movieName);
            } catch (SQLException e) {
                e.printStackTrace();
                logger.warn(e.getMessage() + "添加收藏出错");
            }
        }
    }

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

collection 表存储用户对于电影的收藏对应关系，其数据库操作代码如下：

```
package dao;

import domain.Collection;
import org.apache.commons.dbutils.QueryRunner;
import org.apache.commons.dbutils.handlers.BeanListHandler;
import utils.DataSourceUtils;

import java.sql.SQLException;
import java.util.List;

/**
```

```
 * @ClassName: CollectionDao.java
 * @Description: TODO(处理与 collection 表相关的操作)
 */
public class CollectionDao {

    /**
     * @param userName 用户名
     * @Description: 根据用户名查找记录
     */
    public List<Collection> findAllRecordByUserName(String userName) throws
SQLException {
        String sql = "select * from collection where userName = ? ORDER BY addTime
desc";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        return runner.query(sql, new BeanListHandler<>(Collection.class),
userName);
    }

    /**
     * 向收藏表中添加一条用户喜欢电影的记录
     *
     * @param userName  用户名
     * @param movieName 电影名
     */
    public void addUserCollectionMovie(String userName, String movieName) throws
SQLException {
        String sql = "insert into collection(userName, movieName) values(?,?)";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        runner.update(sql, userName, movieName);
    }



    /**
     * 删除用户对某个电影的收藏记录
```

```
     *
     * @param userName  用户名
     * @param movieName 电影名
     * @return void
     */
    public void cancelUserCollectionMovie(String userName, String movieName)
throws SQLException {
        String sql = "DELETE FROM collection WHERE userName=? and movieName=?";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        runner.update(sql, userName, movieName);
    }
}
```

以上述 users_test 用户为例，假设该用户想要收藏“奇幻”分类中的“哪吒之魔童降世”、“沉睡魔咒 2”、“罗小黑战记”3 部电影，只需要在电影名称下面点击“爱心”图标，将其变成红色，说明收藏成功，如下图：

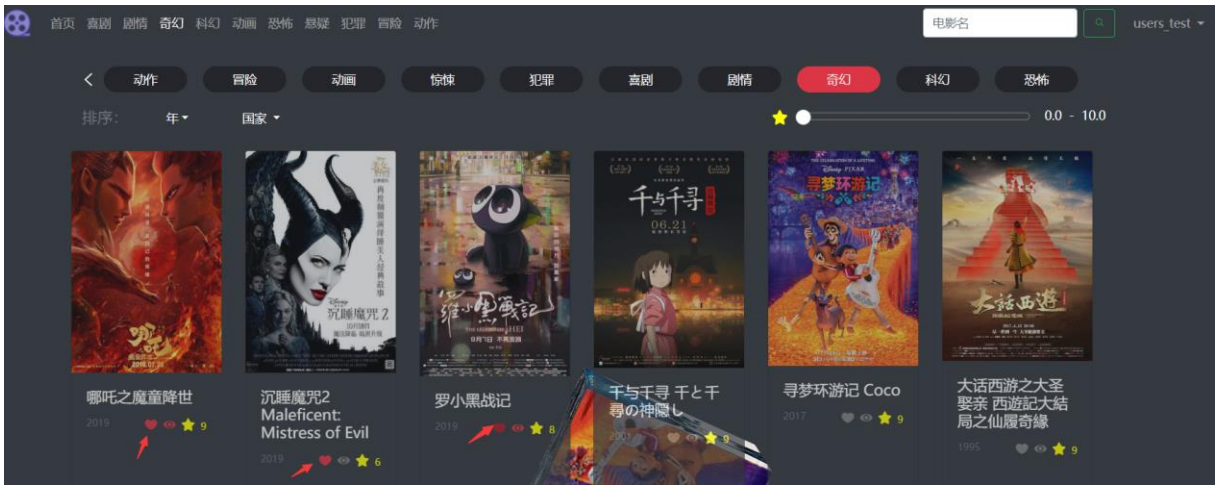

图 3-11 收藏电影

查看 collction 数据库，成功添加了用户与电影的收藏关系：

userName	movieName	addTime
users_test	哪吒之魔童降世	2020-10-31 17:56:23
users_test	沉睡魔咒2 Maleficent: Mis	2020-10-31 17:56:24
users_test	罗小黑战记	2020-10-31 17:56:25
yuanxiaojian	双子杀手 Gemini Man	2020-10-31 11:32:51
yuanxiaojian	哪吒之魔童降世	2020-10-31 11:32:52

图 3-11 收藏电影数据表

3.3 其它展示

在该网站的设计中，为了方便用户表达对于电影的看法与见解，添加了用户对电影的评论功能，其 Servlet 操作代码如下：

```
public class CommentServlet extends HttpServlet {
    static Logger logger = Logger.getLogger(CommentServlet.class);
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        /*request.setCharacterEncoding("utf-8");
        response.setCharacterEncoding("utf-8");*/

        String description = request.getParameter("description");
        String movieName = request.getParameter("movieName");
        User user = (User) request.getSession().getAttribute("user");
        PrintWriter writer = response.getWriter();
        logger.warn(description + " " + movieName);
        Comment comment = new Comment();
        comment.setMovieName(movieName);
        comment.setDescription(description);
        comment.setUserName(user.getUsername());

        CommentService service = new CommentService();
        try {
            service.addComment(comment);
            writer.write("ok");
        } catch (SQLException e) {
            e.printStackTrace();
        }

    }

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

```
        doPost(request, response);
    }
}
```

其数据库操作代码如下：

```
public class CommentDao {
    /**
     * @param userId
     * @param movieId
     * @Description: TODO(根据用户 Id 和电影 id 查找评论)
     */
    public String findCommentByUserIdAndMovieId(int userId, int movieId) {
        return null;
    }
    /**
     * @param movieName 电影名
     * @return 返回评论集合
     * @Description: 根据电影名，查找该电影的所有评论
     */
    public List<Comment> findCommentsByMovieName(String movieName) throws
SQLException {
        String sql = "select * from comments where movieName = ?  ORDER BY addTime
DESC";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
        return runner.query(sql, new BeanListHandler<>(Comment.class),
movieName);
    }
    /**
     * 添加一条评论
     * @param comment 评论对象
     */
    public void addComment(Comment comment) throws SQLException {
        String sql = "insert into comments(userName,movieName,description)
values(?,?,?)";
        QueryRunner runner = new QueryRunner(DataSourceUtils.getDataSource());
```

```
        runner.update(sql,    comment.getUserName(),    comment.getMovieName(),
comment.getDescription());
    }
}
```

具体进行评论操作时，首先选择一部电影，进入该电影详情页面，然后点击“评论”，输入想要评论的内容后，点击“提交评论”：

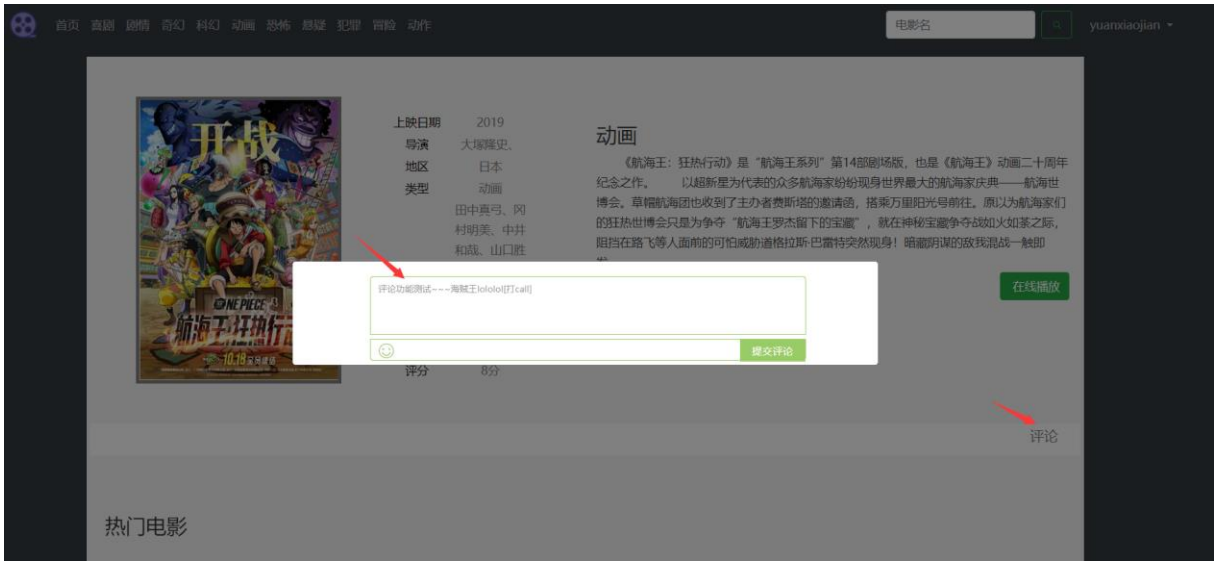

图 3-12 评论电影

我们评论的用户使 yuanxiaojian，评论后效果如下：

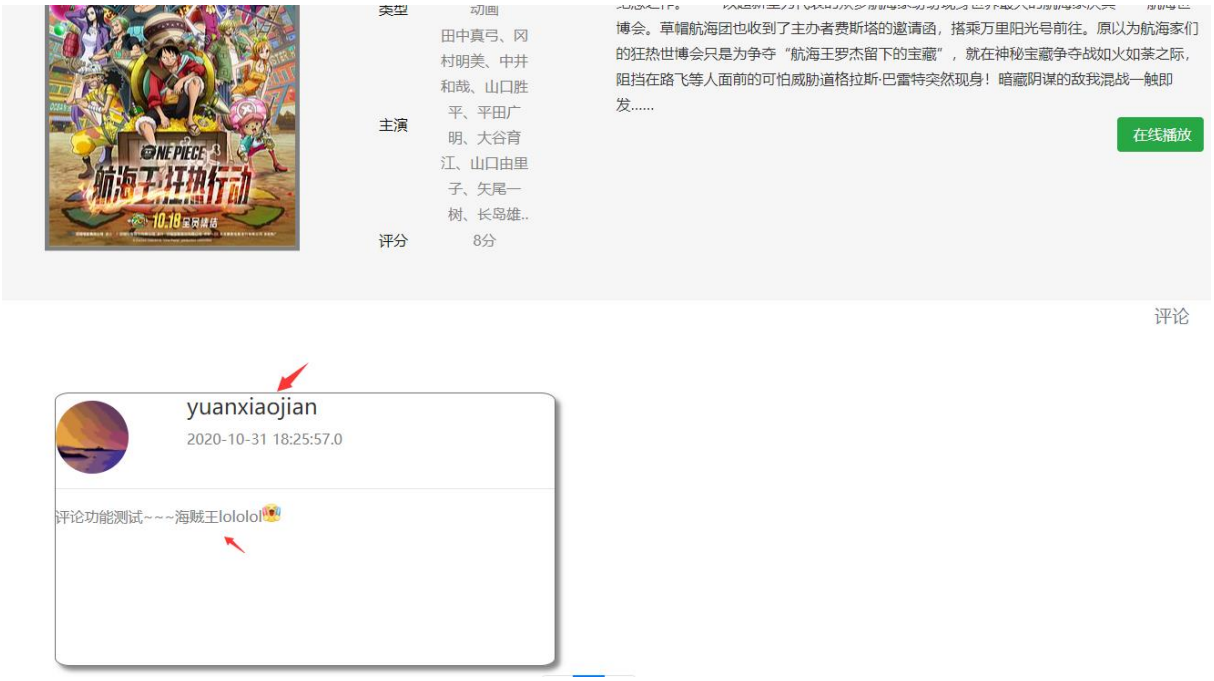

图 3-13 评论效果

评论后查看数据库中 comments 表的效果，如下图：

userName	movieName	description	addTime
yuanxiaojian	航海王：狂热行动 ONE PIECE STAMPEDE	评论功能测试~~~海贼王lololol[打call]	2020-10-31 18:25:57

图 3-14 comments 数据库添加评论

3.4 设计心得

经过完成本次实验，真正将网站完善成了一个具有实际功能的站点，并通过多数据的增删改查操作，使 web 开发知识与之前所学的数据库知识进行了结合，形成了一个更加连贯全面的知识体系。为了使网站具有更加丰富的功能，我掌握了 Servlet 的开发与配置，并且为了网站的安全性限制，熟悉了 Filter 的开发与配置。最后为了进一步的丰富网站功能，更是加深了对 MVC 模式的理解，熟悉了数据库连接操作和 Java 类的编写，增强了编程能力，提高了处理问题的能力。总的来说，通过这最后一次实验，以及与前两次实验的关联，使我真正学会了一个 Java Web 网站的开发，但目前功能还不算太复杂，在之后的学习过程中，将尝试使用 Spring 等框架进行更加深入的开发。