北京郵電大學

Beijing University of Posts and Telecommunications

实　　验　　报　　告

实验题目：	实验二：中国旅游省份推荐平台
课　　程：	网络信息系统基础
学　　院：	电子工程学院
班　　级：	2022211202
学　　号：	2022212388
姓　　名：	俞霁航

本人郑重声明，所提交的论文，为本人独立完成，除已经注明引用的内容外，论文未剽窃，抄袭他人成果。

1.选题的主题问题定义

1.1 描述应用的主题和背景。

本次实验我选择了**“文化旅游展示平台”**作为主题，希望通过这个平台展示中国不同地区的文化和旅游资源。用户可以通过登录功能进入平台，而平台首页是一个 3d 的可以旋转的中国地图，用户在地图上点击对应省市就可以进入详细介绍的二级页面，了解各省的自然风光、历史文化和现代发展等内容。

整个应用最大的亮点是一个立体的三维互动地图，我在设计时特别注重让它看起来高端大气，通过 Three.js 技术让地图变得生动逼真。用户不仅可以看到地图的全貌，还能点击具体省份了解更多信息，比如历史文化、著名景点或者标志性建筑，让每次探索都充满惊喜。

现在人们越来越多地通过线上平台获取旅游信息，但传统平台的展示方式往往比较单调，不够吸引人。我希望通过这个平台，用更丰富的内容和互动效果，让用户对中国的文化和旅游资源有更直观的感受，同时也提升平台的趣味性和体验感。

1.2 说明应用需要解决的问题和实现的目标。

需要解决的问题

1. **用户登录和身份验证**:

在设计中，用户需要先登录才能访问平台的核心内容，确保数据安全和访问

控制。

2. **页面数据传递和管理**:

平台包含多个页面，不同页面之间需要共享信息，比如用户的登录状态、访问记录等。

3. **如何提升视觉体验**:

普通的旅游展示页面往往枯燥单调，如何通过动态效果和交互设计吸引用户是个重要的问题。

交互地图作为平台的核心，如何设计得既好看又实用，同时还能让用户操作起来非常流畅。

4. **易用性与直观性**:

页面需要结构清晰，功能易懂，确保用户可以快速找到自己感兴趣的内容。

实现的目标

1. **打造完整的用户登录功能**:

用户通过用户名和密码登录平台，登录后可以访问核心页面。

使用 Session 管理用户状态，支持短期会话；通过 Cookie 实现“记住我”功能，方便用户长期登录。

2. **设计一个立体交互地图**:

利用 Three.js 和 D3.js 技术打造一个三维互动地图，这是整个实验的亮点。

地图支持鼠标悬停显示省份信息，点击省份跳转到详细页面，让用户可以直观方便地探索中国各个省市的旅游资源。

3. **页面的动态效果**:

使用 JavaScript 和 CSS 动画，让页面充满活力，比如背景渐变、图片悬停放

大等效果。

同时还需要确保页面流畅性，提升用户在平台上的整体体验。

4. **展示丰富的内容**:

平台不仅有地图页面，还有多个详细页面，展示各省份的自然景观、历史遗迹、民俗文化等信息。

这些内容也清晰高端。

2.程序设计

2.1 登录设计:

登陆逻辑采用了实验要求的 cookie 和 session 技术。

2.1.1 功能描述:

1. **用户登录功能的设计目标**:

提供一个直观、便捷且安全的登录界面，用户只能通过输入正确的用户名和密码即可访问平台。

利用 Session 实现用户登录状态的管理，确保登录用户可以在输入密码后正常访问界面，不会因为刷新等掉出页面。

```
String userName = request.getParameter("userName");
String password = request.getParameter("password");
```

实现“记住我”功能，通过 Cookie 存储用户名和密码，让用户在 7 天内无需重复登录。

```
// 设置用户名和密码到 Cookie
Cookie userCookie = new Cookie("userName", userName);
Cookie passwordCookie = new Cookie("password", password);
userCookie.setMaxAge(60 * 60 * 24 * 7); // Cookie 有效期为 7 天
passwordCookie.setMaxAge(60 * 60 * 24 * 7);
response.addCookie(userCookie);
response.addCookie(passwordCookie);
```

2. **用户身份验证的实现过程**:

用户在 index.jsp 页面输入用户名和密码，并提交到 processLogin.jsp 进行验证。

```
<input type="text" name="userName" class="input" placeholder="请输入您的旅行代号" required>
<input type="password" name="password" class="input" placeholder="请输入您的密码" required>
```

后端模拟用户数据（用户名和密码）用于身份验证（实验中使用固定的用户名和密码）。

```
if (userName != null && password != null && !userName.isEmpty() && !password.isEmpty())
```

验证通过:

登录信息（用户名和密码）保存在 Session 和 Cookie 中。

```
if (VALID_USERNAME.equals(userName) && VALID_PASSWORD.equals(password)) {
    // 保存用户名和密码到 Session
    session.setAttribute("userName", userName);
    session.setAttribute("password", password);

    // 设置用户名和密码到 Cookie
    Cookie userCookie = new Cookie("userName", userName);
    Cookie passwordCookie = new Cookie("password", password);
    userCookie.setMaxAge(60 * 60 * 24 * 7); // Cookie 有效期为 7 天
    passwordCookie.setMaxAge(60 * 60 * 24 * 7);
    response.addCookie(userCookie);
    response.addCookie(passwordCookie);

    // 重定向到主页面 main.jsp
    response.sendRedirect("main.jsp");
```

重定向到主页面 main.jsp。

验证失败:

重定向回登录页面 index.jsp，并通过 URL 参数附带错误信息。

```
response.sendRedirect("index.jsp?error=invalid");
```

2.1.2 数据设计：

1. **数据的含义与作用**：

用户名（userName**）**：

表示用户的唯一标识。

用于验证用户身份和记录登录状态。

密码（password**）**：

用于身份验证，确保用户输入正确密码。

登录状态（Session **数据）**：

标识用户是否已登录，确保用户在不同页面访问时不需重复验证。

记住我功能（Cookie **数据）**：

存储用户的用户名和密码，简化后续登录操作。

2. **数据的存储范围**：

用户名和登录状态：

存储在 Session 中，与当前会话绑定，作用范围为整个用户会话。

记住我功能数据：

存储在客户端的 Cookie 中，作用范围为用户浏览器，跨会话有效，持续 7 天。

表单提交的数据：

临时存储在 Request 对象中，作用范围仅限于登录页面提交到处理页面。

3. **使用的技术**：

JSP 内置对象：

Request：接收用户输入的数据。

```
// 获取用户输入的用户名和密码
String userName = request.getParameter("userName");
String password = request.getParameter("password");
```

Response：设置 Cookie 和跳转页面。

```
response.addCookie(userCookie);
response.addCookie(passwordCookie);
```

Session：管理用户登录状态。

```
session.setAttribute("userName", userName);
session.setAttribute("password", password);
```

Cookie：实现记住用户名和密码的功能，简化用户体验。

```
// 设置用户名和密码到 Cookie
Cookie userCookie = new Cookie("userName", userName);
Cookie passwordCookie = new Cookie("password", password);
userCookie.setMaxAge(60 * 60 * 24 * 7); // Cookie 有效期为 7 天
passwordCookie.setMaxAge(60 * 60 * 24 * 7);
```

2.1.3 实现细节：

页面和逻辑的具体实现：

登录页面（index.jsp**）**：

提供登录表单，用户输入用户名和密码。

检查 Session 和 Cookie，如果已存在有效的用户信息，直接跳转到 main.jsp。

```
if ((userName != null && password != null) || (cookieUserName != null && cookiePassword != null)) {
    response.sendRedirect("main.jsp");
    return; // 防止后续代码执行
}
```

登录逻辑处理页面（processLogin.jsp**）**:

接收用户输入的用户名和密码。

验证用户名和密码的正确性。

设置 Session 和 Cookie。

根据验证结果，重定向到主页面或返回登录页面。

```
// 获取用户输入的用户名和密码
String userName = request.getParameter("userName");
String password = request.getParameter("password");

if (userName != null && password != null && !userName.isEmpty() && !password.isEmpty()) {
    // 验证用户名和密码
    if (VALID_USERNAME.equals(userName) && VALID_PASSWORD.equals(password)) {
        // 保存用户名和密码到 Session

        session.setAttribute("userName", userName);
        session.setAttribute("password", password);

        // 设置用户名和密码到 Cookie
        Cookie userCookie = new Cookie("userName", userName);
        Cookie passwordCookie = new Cookie("password", password);
        userCookie.setMaxAge(60 * 60 * 24 * 7); // Cookie 有效期为 7 天
        passwordCookie.setMaxAge(60 * 60 * 24 * 7);

        response.addCookie(userCookie);
        response.addCookie(passwordCookie);

        // 重定向到主页面 main.jsp
        response.sendRedirect("main.jsp");
    } else {
        // 验证失败，重定向回登录页面并附加错误信息
        response.sendRedirect("index.jsp?error=invalid");
    }
} else {
    // 如果用户名或密码为空，重定向回登录页面并附加错误信息
    response.sendRedirect("index.jsp?error=empty");
}
```

JSP 内置对象的使用方式:

Request 对象:

使用 request.getParameter("userName")和 request.getParameter("password")获取用户提交的表单数据。

Session 对象:

使用 session.setAttribute()函数保存用户的登录信息。

Response 对象:

使用 response.addCookie(cookie) 函数将用户名和密码存储在客户端。

使用 response.sendRedirect("main.jsp")函数实现页面跳转。

Cookie 对象:

创建 Cookie 存储用户名和密码，并设置有效期为 7 天。

数据的传递与校验:

表单提交:

用户输入的数据通过 POST 请求传递到 processLogin.jsp。

后端验证:

通过预定义的用户名和密码（traveler 和 securepass）验证用户身份。

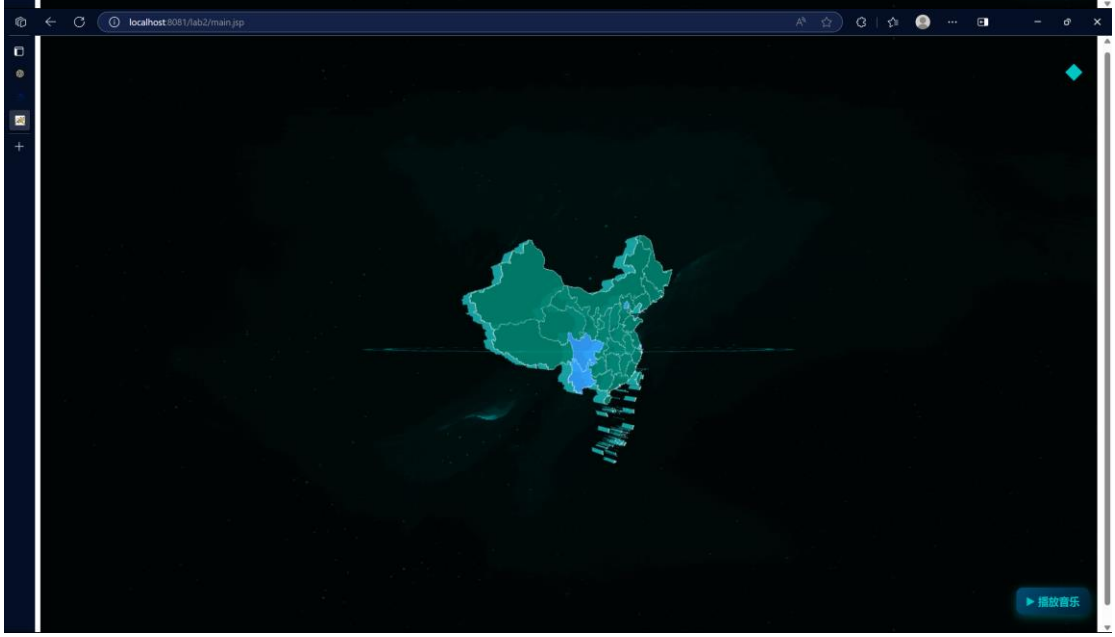
验证通过:

设置 Session 和 Cookie，并跳转到主页面。

验证失败:

重定向回登录页面，并附带错误信息（index.jsp?error=empty）。

2.2 视觉效果设计：



2.2.1 功能描述：

1. **页面动态效果的目标**：

提供用户沉浸式的视觉体验，页面不仅静态呈现内容，还要通过动态效果提升用户的交互感和参与感。

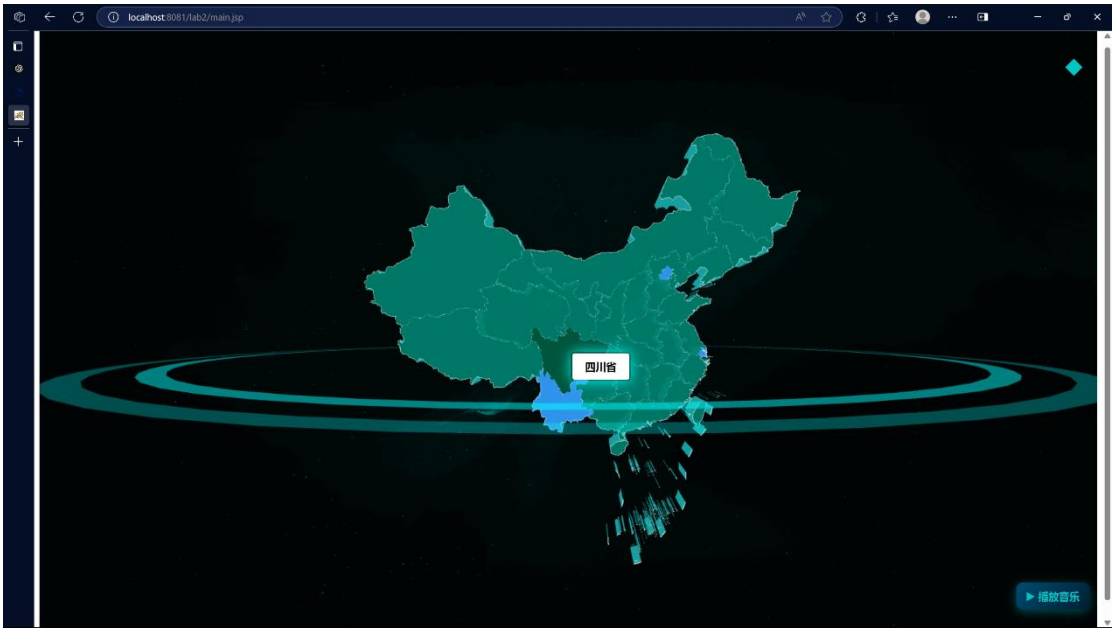
使用动态样式切换、动画效果以及三维交互地图，丰富页面展示内容，让用户能够直观感受到中国文化和旅游的魅力。

2. **页面间的信息传递设计**:

用户可以通过点击地图中的省份跳转到具体页面，浏览该省份的具体旅游信息，由于精力有限，我只做了四个省份的具体信息，这部分实现了页面间的信息传递。

使用 JavaScript 在地图元素上绑定事件，通过 userData 记录目标页面 URL，点击后跳转到对应的详细页面。

提供鼠标悬停提示功能，当用户鼠标移到某个省份时，显示省份的名称，并通过颜色变化提示用户可以点击。



注：鼠标被截图软件隐藏

2.2.2 技术实现：

使用的技术：

Three.js：构建三维地图模型，并通过相机和光线效果增强立体感。

JavaScript：实现地图交互（点击、悬停）、页面跳转以及动态效果控制。

CSS 动画：为页面元素（如背景、标题、按钮等）添加渐变、缩放等动态效果，提升视觉吸引力。

HTML5 Audio：实现音乐播放功能，增强用户体验。

页面布局设计与动态元素的实现细节：

地图布局：

使用 Three.js 构建地图模型，加载 JSON 地图数据。

通过 d3.geoMercator 投影将地理坐标转换为平面坐标，并用 Three.js 创建几何形状。

为可以点击和不可点击的省份设置不同颜色，通过鼠标交互改变颜色。

动态元素：

标题栏、内容框使用 CSS 动画实现动态加载效果。

鼠标悬停地图时显示省份信息，点击后跳转到详细页面。

代码讲解

1. **Three.js 地图初始化**： 在 init()函数中初始化地图场景、相机和渲染器：

```
function init() {
    // 创建场景
    scene = new THREE.Scene();

    // 创建相机
    camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
    camera.position.set(cameraInitialPosition.x, cameraInitialPosition.y, cameraInitialPosition.z);

    // 创建渲染器
    renderer = new THREE.WebGLRenderer({antialias: true});
    renderer.setSize(window.innerWidth, window.innerHeight);
    container.appendChild(renderer.domElement);

    // 控制器
    controls = new THREE.OrbitControls(camera, renderer.domElement);

    // 添加背景图
    const textureLoader = new THREE.TextureLoader();
    textureLoader.load('index/pageBg.png', function (texture) {
        scene.background = texture;
    });

    // 加载地图数据
    loadMapData();

    // 添加圆环
    addRings();

    // 监听窗口大小变化
    window.addEventListener('resize', onWindowResize, false);

    // 添加鼠标点击事件监听器
    container.addEventListener('click', onMouseClick, false);

    // 开始动画循环
    animate();
}
```

2. **加载地图数据**：使用 FileLoader 读取地图 JSON 数据，并生成三维几何图形：

```
function loadMapData() {
    const loader = new THREE.FileLoader();
    loader.load('index/ChinaMap.json', function (data) {
        const jsonData = JSON.parse(data);
        createMap(jsonData);
    });
}
```

```
function createMap(data) {
    chinaMap = new THREE.Object3D();
    const projection = d3.geoMercator()
        .center([104.0, 37.5])
        .scale(80)
        .translate([0, 0]);

    data.features.forEach(function (feature) {
        const province = new THREE.Object3D();
        const coordinates = feature.geometry.coordinates;
        coordinates.forEach(function (multiPolygon) {
            multiPolygon.forEach(function (polygon) {
                const shape = new THREE.Shape();
                const pointsArray = [];
                for (let i = 0; i < polygon.length; i++) {
                    const [x, y] = projection(polygon[i]);
                    if (i === 0) {
                        shape.moveTo(x, -y);
                    } else {
                        shape.lineTo(x, -y);
                    }
                    pointsArray.push(new THREE.Vector3(x, -y, mapThickness));
                }
                const lineGeometry = new THREE.BufferGeometry().setFromPoints(pointsArray);

                const extrudeSettings = {
                    depth: mapThickness,
                    bevelEnabled: false
                };
                const geometry = new THREE.ExtrudeGeometry(shape, extrudeSettings);

                // 根据省份名称设置颜色
                let provinceColor = mapDefaultColor;
                if (provinceColors[feature.properties.name]) {
                    provinceColor = provinceColors[feature.properties.name];
                }
```

```
            const materialTop = new THREE.MeshBasicMaterial({
                color: provinceColor,
                transparent: true,
                opacity: 0.92
            });
            const materialSide = new THREE.MeshBasicMaterial({
                color: mapEdgeColor,
                transparent: true,
                opacity: 0.92
            });

            const mesh = new THREE.Mesh(geometry, [materialTop, materialSide]);
            const lineMaterial = new THREE.LineBasicMaterial({color: "white"});
            const line = new THREE.Line(lineGeometry, lineMaterial);

            province.properties = feature.properties;
            province.add(mesh);
            province.add(line);

            // 如果是需要跳转的省份，给mesh添加userData
            if (provinceToPage[feature.properties.name]) {
                mesh.userData.url = provinceToPage[feature.properties.name];
                // 可选：改变鼠标样式
                mesh.cursor = 'pointer';
            }
        });
    });
    chinaMap.add(province);
});

chinaMap.position.set(mapInitialPosition.x, mapInitialPosition.y, mapInitialPosition.z);
scene.add(chinaMap);
}
```

3. **交互实现**：

鼠标悬停高亮省份： 使用射线检测用户鼠标指向的地图区域，并改变省份颜色：

```
function animate() {
    requestAnimationFrame(animate);

    // 旋转圆环
    if (ring1 && ring2) {
        ring1.rotation.z += 0.005;
        ring2.rotation.z -= 0.005;
    }

    // 更新射线
    if (chinaMap) {
        if (lastIntersected) {
            // 恢复省份的颜色
            let provinceName = lastIntersected.object.parent.properties.name;
            let provinceColor = mapDefaultColor;
            if (provinceColors[provinceName]) {
                provinceColor = provinceColors[provinceName];
            }
            lastIntersected.object.material[0].color.set(provinceColor);
            lastIntersected.object.material[1].color.set(mapEdgeColor);

            const tooltip = document.getElementById('tooltip');
            if (tooltip) {
                tooltip.style.visibility = 'hidden';
            }
        }
        lastIntersected = null;

        raycaster.setFromCamera(mouse, camera);
        const intersects = raycaster.intersectObjects(chinaMap.children, true);
        lastIntersected = intersects.find(function (item) {
            return item.object.material && item.object.material.length === 2;
        });

        if (lastIntersected && lastIntersected.object.parent.properties.name) {
            lastIntersected.object.material[0].color.set(mapHoverColor);
            lastIntersected.object.material[1].color.set(mapHoverColor);
            showTooltip();
```

```
            // 如果是可点击的省份，改变鼠标样式
            if (lastIntersected.object.userData.url) {
                document.body.style.cursor = 'pointer';
            } else {
                document.body.style.cursor = 'default';
            }
        } else {
            document.body.style.cursor = 'default';
        }
    }

    controls.update();
    renderer.render(scene, camera);
}
```

点击跳转到详细页面： 在 onMouseClick 中检查省份的 userData，并跳转到对应页面：

```
function onMouseClick(event) {
    if (lastIntersected && lastIntersected.object.userData.url) {
        window.location.href = lastIntersected.object.userData.url;
    }
}
```

3.问题及解决

3.1 实验过程中遇到的问题。

1. **地图数据加载问题**：

   在 Three.js 中加载 ChinaMap.json 数据时，页面报错显示无法正确解析地图数据。

2. **鼠标悬停和点击的交互问题**：

   实现鼠标悬停高亮省份和点击跳转时，偶尔会出现射线检测不到省份的

问题，导致交互失效。

3. **动态背景动画卡顿**:

测试时，发现页面背景的渐变动画在某些浏览器（如旧版 Chrome）中运行时会出现卡顿现象，影响用户体验。

4. **跨页面信息传递不一致**:

点击地图跳转到 page1.jsp 等详细页面后，无法保留用户的登录状态。

3.2 问题产生的原因。

1. **地图数据加载问题**:

地图数据格式与代码中的解析逻辑不匹配，导致 JSON.parse()解析失败。

文件路径错误或数据文件未正确加载。

2. **鼠标悬停和点击的交互问题**:

Three.js 射线检测逻辑没有针对多层嵌套的几何体进行优化，导致一些省份的鼠标事件无法触发。

在鼠标快速移动或切换时，raycaster 的更新速度跟不上。

3. **动态背景动画卡顿**:

动画使用了过大的渐变范围（background-size: 300%），浏览器渲染时占用大量资源。

卡顿可能与设备性能和浏览器优化程度有关。

4. **跨页面信息传递不一致**:

用户登录状态只存储在 Session 中，而跳转到新页面时没有从 Session 恢复状态。

Cookie 数据未正确同步到后续页面。

3.3 具体解决方法和步骤。

1. **地图数据加载问题的解决方法**：

   确保 ChinaMap.json 的路径正确，并验证文件的 JSON 格式是否符合预期。

2. **鼠标悬停和点击的交互问题的解决方法**：

   优化 raycaster 的逻辑，增加对嵌套对象的检测，并限制检测层级：

```
const loader = new THREE.FileLoader();
loader.load(
    'index/ChinaMap.json',
    (data) => {
        try {
            const jsonData = JSON.parse(data);
            createMap(jsonData);
        } catch (error) {
            console.error("地图数据解析失败：", error);
        }
    },
    undefined,
    (error) => {
        console.error("地图数据加载失败：", error);
    }
);
```

3. **动态背景动画卡顿的解决方法**：

   减少渐变范围，并降低动画的刷新频率：

```css
body {
    background: linear-gradient(to bottom, #001f3f, #003d66, #0073e6);
    background-size: 200% 200%;
    animation: bgGradient 15s infinite alternate ease-in-out;
}

@keyframes bgGradient {
    0% {
        background-position: 0% 50%;
    }
    100% {
        background-position: 100% 50%;
    }
}
```

优化浏览器渲染性能，通过减少 DOM 重绘和重排。

4. **跨页面信息传递不一致的解决方法**：

使用 Cookie 保存用户登录状态，确保在所有页面中都能验证用户身份：

```
Cookie sessionCookie = new Cookie("sessionKey", session.getId());
sessionCookie.setMaxAge(7 * 24 * 60 * 60);  // 7 天
response.addCookie(sessionCookie);
```

在每个页面检查 Cookie 并恢复登录状态：

```
<%
    Cookie[] cookies = request.getCookies();
    String sessionKey = null;

    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if ("sessionKey".equals(cookie.getName())) {
                sessionKey = cookie.getValue();
            }
        }
    }

    if (sessionKey == null || !sessionKey.equals(session.getId())) {
        response.sendRedirect("index.jsp?error=sessionExpired");
    }
%>
```

4.感想

4.1 总结实验的收获与心得。

通过这次实验，我对 JSP 和前端技术的实际应用有了更加深刻的理解。从用户登录到页面动态效果再到跨页面交互，每个功能模块的实现让我切身体会到了 Web 应用开发的完整流程。在实践中，我发现了一些自己之前忽略的细节问题，比如跨页面信息传递的稳定性、动态效果与性能的平衡等，这些都是在开发过程中需要不断优化和完善的地方。

本次实验中最大的亮点是通过 Three.js 构建了一个立体的中国地图。这个模块让我学会了如何加载外部数据、渲染三维模型，并与用户交互。尽管遇到了一些挑战，但最终实现了省份悬停高亮、点击跳转以及页面间的逻辑传递等功

能。

此外，通过设置登录功能，我也对 Session 和 Cookie 在状态管理中的作用有了更清晰的理解。这些技术不仅提升了用户体验，还提高了平台的安全性。实验中遇到的 bug 和问题，也让我学会了如何从错误中寻找原因，并通过查阅文档或尝试不同方法进行调试和解决。

4.2 对所学技术的体会。

1. **JSP 的灵活性**： JSP 提供的内置对象（如 Request、Response 和 Session）让数据传递和状态管理变得非常方便。同时，JSP 的动态内容生成能力使得前后端的融合更加紧密，但也需要小心管理代码结构，以免逻辑过于混乱。
2. **Three.js 的强大功能**： 作为一个专注于三维图形渲染的库，Three.js 的功能非常全面。从创建几何形状到处理复杂交互，再到控制相机和光照，每一步都有丰富的 API 支持。这次实验让我深刻体会到三维交互的魅力，同时也意识到优化三维渲染性能的重要性。
3. **JavaScript 的实用性**： JavaScript 在动态效果和交互逻辑中扮演了重要角色。通过 JS，我实现了动画控制、鼠标交互、页面跳转等功能。它灵活的事件绑定机制和 DOM 操作能力让我可以快速开发用户友好的交互功能。
4. **Session 和 Cookie 的结合应用**： Session 适合管理短期的用户状态，而 Cookie 则适合存储持久化的数据。通过实验，我对这两者的作用范围和结合使用有了更深刻的理解，并能根据需求选择合适的方式存储数据。

5. **CSS 动画的视觉提升**： 动态背景和渐变效果是这次实验中的一大亮点。CSS 动画简单高效，不仅让页面更具吸引力，也让我对 Web 动画的实现方式有了更多的理解。

总结

本次实验不仅是对所学知识的应用，更让我感受到技术整合的力量。从 JSP 到前端技术，再到三维渲染和动画，每一种技术在解决特定问题时都有独特的优势。在实验中，我既学会了如何运用这些技术，也积累了处理实际问题的经验，为今后的开发打下了坚实的基础。

如果有机会，我希望能将实验进一步拓展，比如增加更多省份的详细页面，优化地图的性能，或者加入用户个性化推荐功能，使平台更加智能化和多样化。这次实验让我对 Web 开发产生了更浓厚的兴趣，也期待在未来能运用这些技术开发出更加实用和有趣的项目！