《Java 企业级应用》实验报告

年级、专业、班级	2021 级计卓 1 班	姓名	李宽宇	学号	20215279
实验题目	单点登录系统的设计与实现				
实验时间	2024.4.20	**实验地点**	DS3401		
学年学期	2023-2024(2)	**实验性质**	**□验证性 ■设计性 □综合性**		

一、实验目的

1. 本次实验的目的是掌握 Java 企业级应用开发工具的使用方法；掌握 Web 编程技术，掌握 Servlet，JSP，Filter，Listener 等 Web 设计元素的使用；掌握 Tomcat 服务器的使用。
2. 设计开发完成一个基于 Web 的单点登录系统。熟练掌握单点登录系统的工作原理。能够规范的表述所设计的系统。
3. **不使用 SpringBoot**。抄袭计 0 分。

二、实验项目内容

设计开发完成“单点登录系统”软件。要求如下：

1、实现 2 个简单的、需要用户登录才能使用的应用系统，这 2 个应用系统称为 web1 和 web2；登录进去后，在主页面显示该系统已经登录的所有用户的信息和登录时间。并提供两个系统相互跳转的链接。一个应用系统可以登录多个用户。

2、设计实现一个单点登录系统 ssoserver；提供登录页面等功能。登录后，在主页面显示已经登录的所有用户的信息和登录时间。并提供跳转到 web1 和 web2 的链接，能够直接登录到 web1 或者 web2。

3、未登录状态下访问系统 web1 或者系统 web2，跳转到单点登录系统 ssoserver 进行登录验证，登录成功后，跳转到 web1 或者 Web2；

4、应用系统 web1、web2 和 ssoserver 提供登出注销功能。登出任何一个系统后，再次访问任何一个系统都需要重新登录。登出注销后，跳转到一个页面，显示登出注销成功等信息。

5、可以实现自定义功能。注意操作使用的方便性，注意设计思想的表达，注意优化代码结构，优化类的职责分工。代码有注释。

6、在报告中注明自己的创新点、特色等。

7、提交：（1）本实验报告，（2）源代码压缩文件 zip，（3）软件演示的 MP4 视频，视频大小不超过 40M，视频请在**搜狗浏览器或者 QQ 浏览器**测试能否正常播放。注意源代码加注释。注意文件名称的规范性。文件名：学号姓名 3.docx，学号姓名 3.zip, 学号姓名 3.mp4。**三个文件分别提交**。

三、实验过程或算法（写明创新点或特色、设计思想、设计原理、设计模式的使用、程序的结构、功能关系图、类的说明和类之间的关系图、程序主要执行流程图，最后是核心源代码，截图等）

1. 创新点或特色

(1) 单点登录（SSO 是一种机制，允许用户只需登录一次即可访问所有相关系统，无需单独登录到每个系统。这样做的好处是简化了登录流程，同时提供了额外的安全层。

(2) 同时运行了多个 Tomcat 服务器，模拟了真实的多系统情形；同时将 IP 地址映射到文字，例如：127.0.0.2->www.LoginSystem.com。

(3)使用了 mysql、JDBC、JavaBean、Servlet、JSP 等技术，遵循试图控制器模式（Model-View-Controler, MVC），使用一个或多个 Servlet 作为控制器。请求由前沿的 Servlet 接收并处理后，会重新定向到 JSP。更加明显地将显示与逻辑处理分离开。

(4)使用 maven 管理文件结构、git 管理项目进度，源代码链接：guoluguodong/SSOjava at master (github.com)

（5）使用了 Filter、Session 等技术

（6）使用了桥接模式、装饰器模式、静态工厂、单例模式等设计模式。

2. 设计思想

（1）程序设计重点使用 mysql、JDBC、JavaBean、Servlet、JSP 等技术，采用了多种设计元素，包括集合框架、日期类 LocalDate，格式化器、字节流、充分利用异常处理，使用 maven 管理项目结构，git 管理项目进度。

（2）从代码的规范角度，包名:全小写;类名:首字母大写,每个单词的首字母大写;方法名:小写字母开头,每个单词的首字母大写;变量名: 写字母开头,每个单词的首字母大写;常量名:基本类型的常量名全大写。

（3）前后端分离的设计思路，前端设计了网页类，后端构造关系型数据库，所有有关登录的 post 和 get 都放在 cas 中，共外部调用，使得 web1、2 与用户登录的具体操作独立开来，使得程序呈现出高内聚、低耦合的特点。

(4)设计模式方面，使用了桥接模式、装饰器模式、静态工厂、单例模式等设计模式，所有类满足单一职责原则。

（5）注重程序的实用性与操作使用的方便性。图形界面美观简洁

3. 设计原理

在用户访问单点登录系统的任何一个网站时，首先会检测 cookies 有没有 token，如果有 token 就去 CAS 验证 token，没有 token 就需要跳转到登录系统 CAS。如果登陆系统的 token 存在且正确，就再次重定向到目标网站，实现跨 session 的 token 验证。

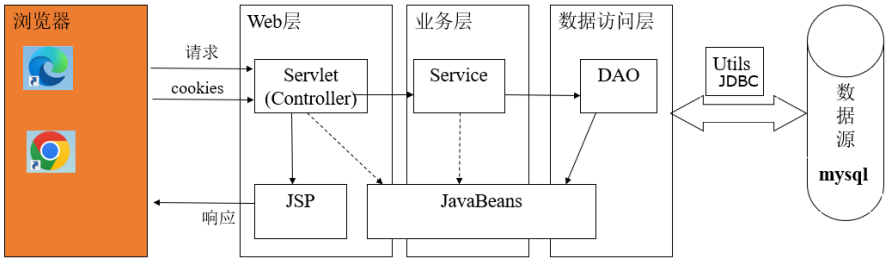

图 1:Servlet+JSP+JavaBean

（1）不同的 ip 地址不共享 cookie，登录了一个系统后，希望另一个系统刷新后也能登录,例如，在登录了 cas 后，希望 web1 输入域名自动登录

方案：所有页面在未登录状态下，都要先查 LoginSystem 的 token，因为这是登录的唯一入口，一定有 token，web1 获取并验证 token 后，写入本 ip 的 cookies，再重定向到 web1mainPage

（2）不同的 ip 地址不共享 cookie，退出登录了一个系统后，希望另一个系统刷新后也能退出登录并返回登录页面,例如，在退出登录了 cas 后，希望 web1 刷新页面后登录失效。

方案：所有页面在登录状态下，重新访问，都要先验证 token，如果有系统退出登录都修改数据库的 token 值为 null,使得其他页面验证 token 不通过.

（3）如何使得 login 后就不能访问登录页，会自动跳转主页。以及 web1 系统中所有页面都有登录的记录

方案：在 cookies 中存储 token，login 后，LoginSystem.com 域名的会话 cookies 中有 token 值且能通过验证，就会从登录页重定向到主页

（4）如何使得未登录状态下访问 web1，就会跳转到 login，登录后直接跳转到 web1，而非主页。

方案：在 web1 重定向到的 login 后面，加 post 参数 web=web1,这样登录后重定向到 web1，没有就默认到 mainPage.

4.设计模式的使用

(1)桥接模式，桥接模式（Bridge Pattern）：JDBC 规范中使用了桥接模式。这是一种结构型设计模式，基于类的最小设计原则。通过使用封装、聚合及继承等行为，让不同的类承担不同的职责，DriverManager 是 JDBC 中的一个关键类，它负责加载数据库驱动并管理数据库连接。它充当了桥接模式中的桥接角色，将应用程序与具体的数据库驱动隔离开来。JDBC 使用反射方法加载数据库驱动。每个数据库厂商的驱动类必须实现 java.sql.Driver 接口，并在加载时将自身注册到 DriverManager 中。例如，对于 MySQL，我们可以通过 Class.forName("com.mysql.jdbc.Driver") 来加载驱动。

(2)工厂模式：在 JDBC 中，我们使用 DriverManager.getConnection(...) 来获取数据库连接。这是一个静态工厂方法，它隐藏了具体的数据库驱动实现细节，让我们可以通过统一的接口获取连接。

(3)装饰器模式：装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。@WebServlet("/clearcookie")为 jsp 查找 servlet 提供了定位。

(4) 单例模式:userService 作为 service 的单例,userDao 作为 UserDao 的单例模式。单例模式（Singleton Pattern）这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。这种模式涉及到一个单一的类，该类负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

4. 程序的结构

(1) 对于 cas 服务器，其遵循遵循试图控制器模式（Model-View-Controler, MVC），使用一个或多个 Servlet 作为控制器。请求由前沿的 Servlet 接收并处理后，会重新定向到 JSP。如下图所示。

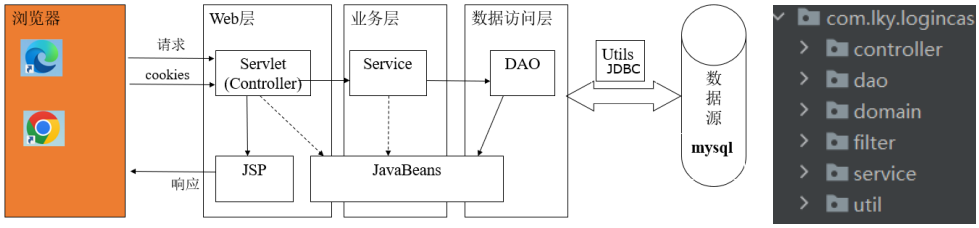

图 2:Servlet+JSP+JavaBean 程序结构图

其中 controller 使用多个多个 Servlet 作为控制器，目录下包括：

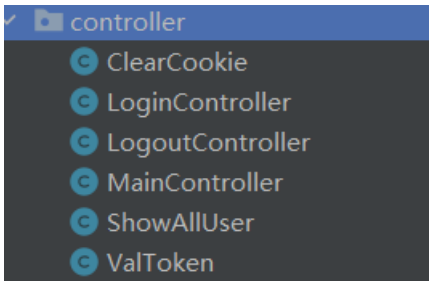

图 3:controller 目录

Dao 用于与数据库交互，包括：

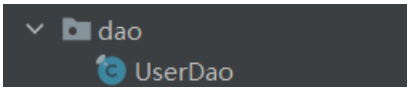

图 4:dao 目录

Domain 目录下存储实体类 user

图 5: Domain 目录

Filter 目录下存储加载页面的过滤器

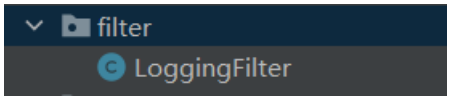

图 6: Filter 目录

Service 目录下存储 usersevice，作为业务层

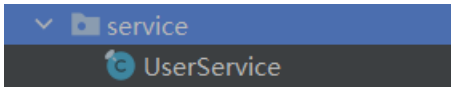

图 7: Service 目录

Util 目录下存储 JDBC 和 Token 的生成算法

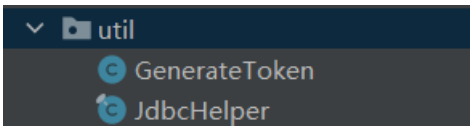

图 8: Util 目录

Webapp 存放 jsp 文件和资源文件

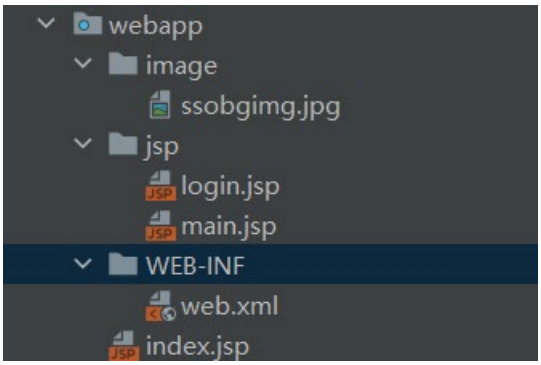

图 9:Webapp 目录

index.jsp 是主目录，重定向到 login.jsp 登录页面，登录完成跳转到 main.jsp 主页面。

（2）web1，只有 logoutController，其他 post 和 get 请求都要向 cas 发起。

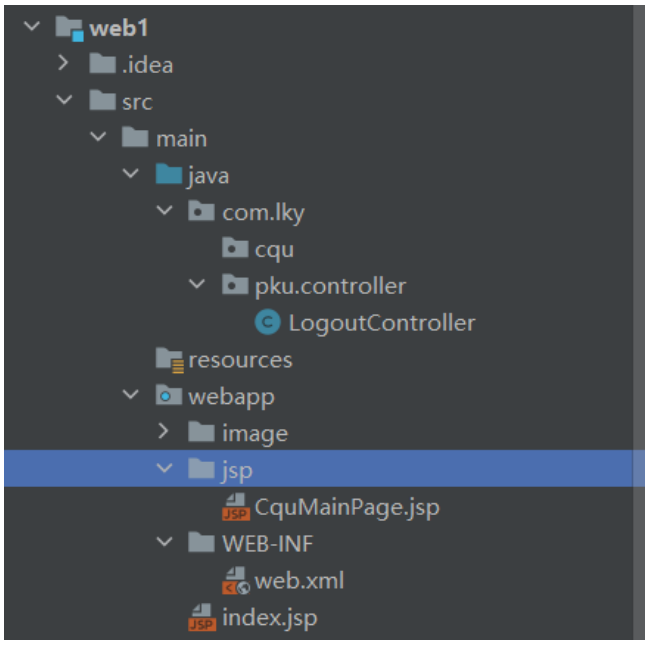

图 10:web1 目录

（3）web2 同 web1

5. 功能关系图

web 服务器，cas 服务器，客户端关系图

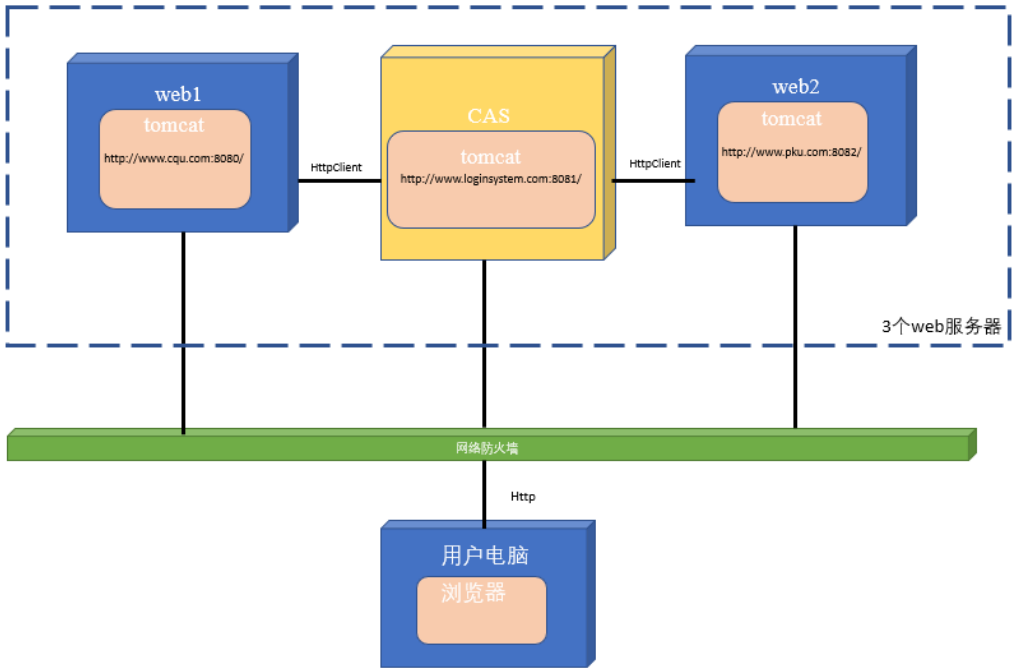

图 11: web 服务器，cas 服务器，客户端关系图

核心功能图,下图描述了 SSO 机制，允许用户只需登录一次即可访问所有相关系统，无需单独登录到每个系统。

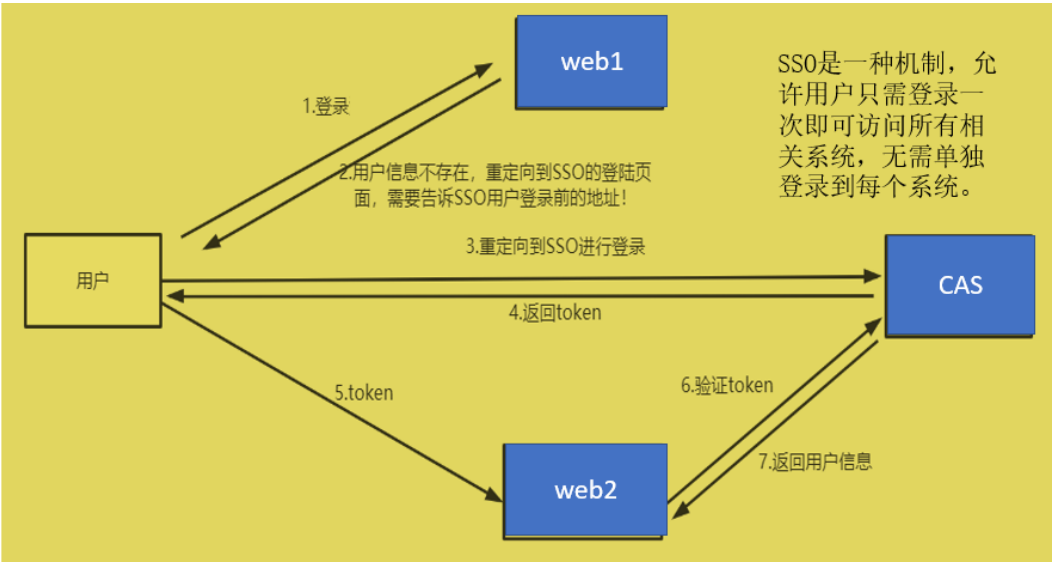

图 12: 核心功能图

6. 类的说明和类之间的关系图

（一）类的说明

（1）实体类

User:用户类，属性包括 id，用户名，密码，最近登录时间，token,如果 token 为 null 则处于退出登录状态

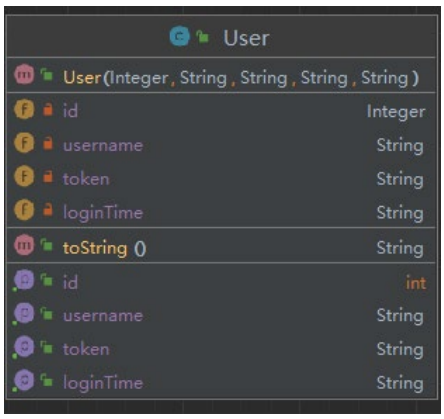

图 13:user 类

（2）控制类

数据访问层，userDao 类

图 14:userDao 类

Web 层，包括 MainController、LogoutController、LoggingFilter、LogoutController、ValToken、ClearCookie 等，这些类实现 HTTPSerevlet，为外部和内部 get 和 post 提供了接口。

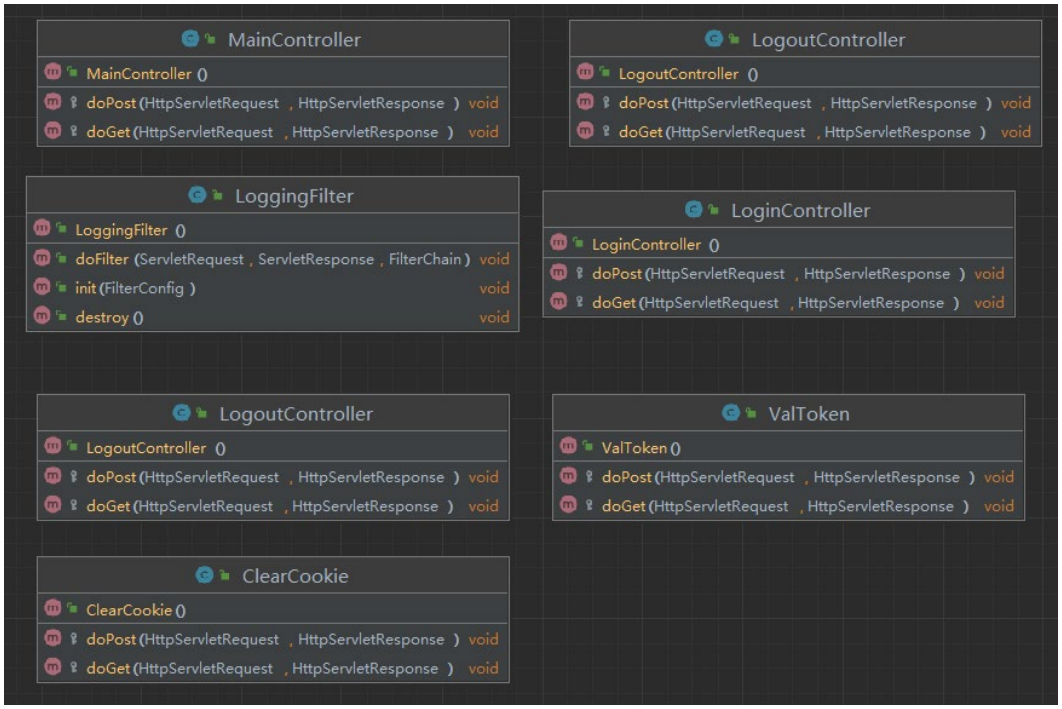

图 15:HttpSerevlet 的子类

（3）工具类

GenerateToken 和 JdbcHelper 为 token 生成和数据库连接的工具类

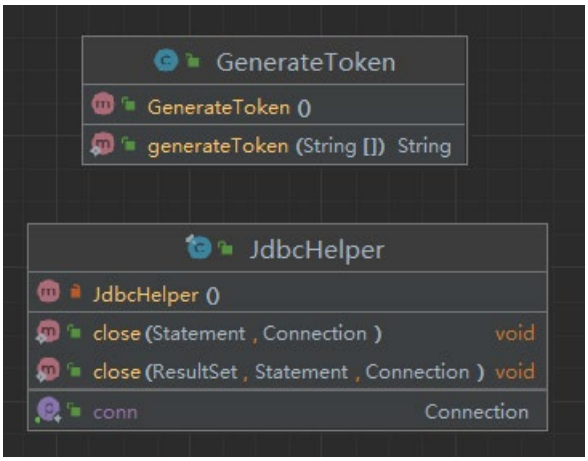

图 16: GenerateToken 和 JdbcHelper 类

（二）类关系图

图 17:类关系图

7. 程序主要执行流程图

时序图——访问登录页面

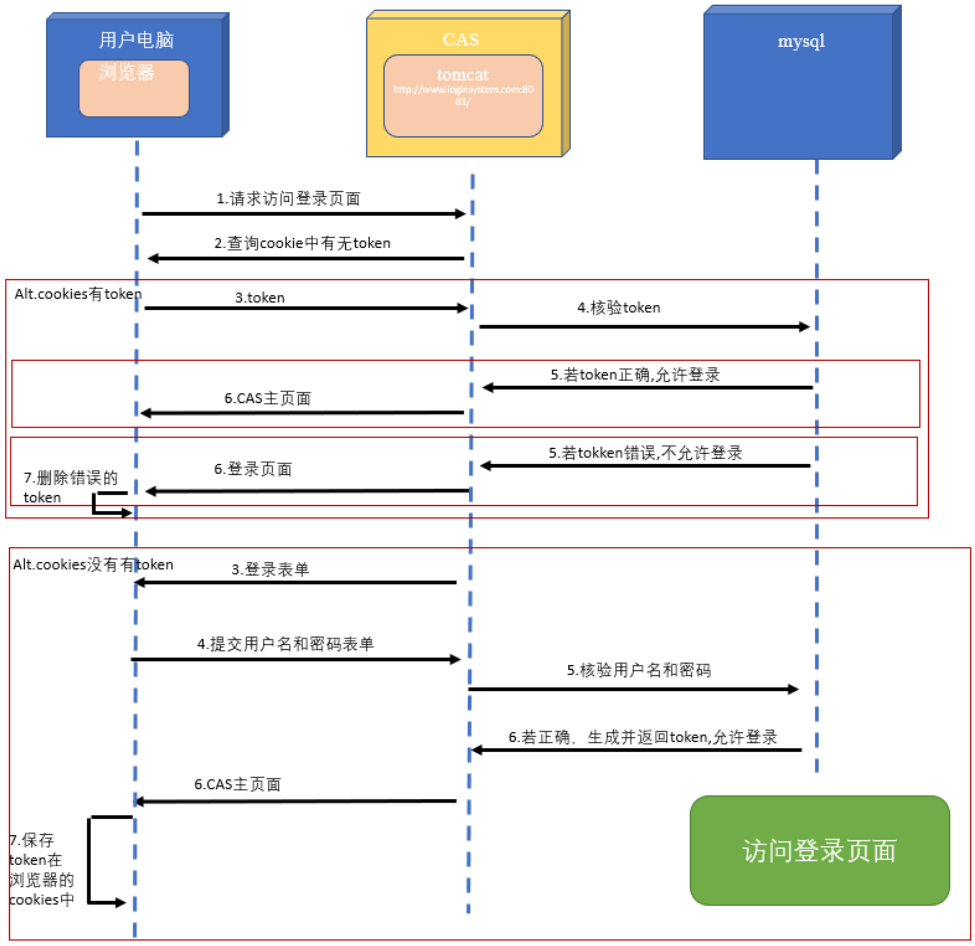

图 18:访问登录页面时序图

时序图——访问 web1

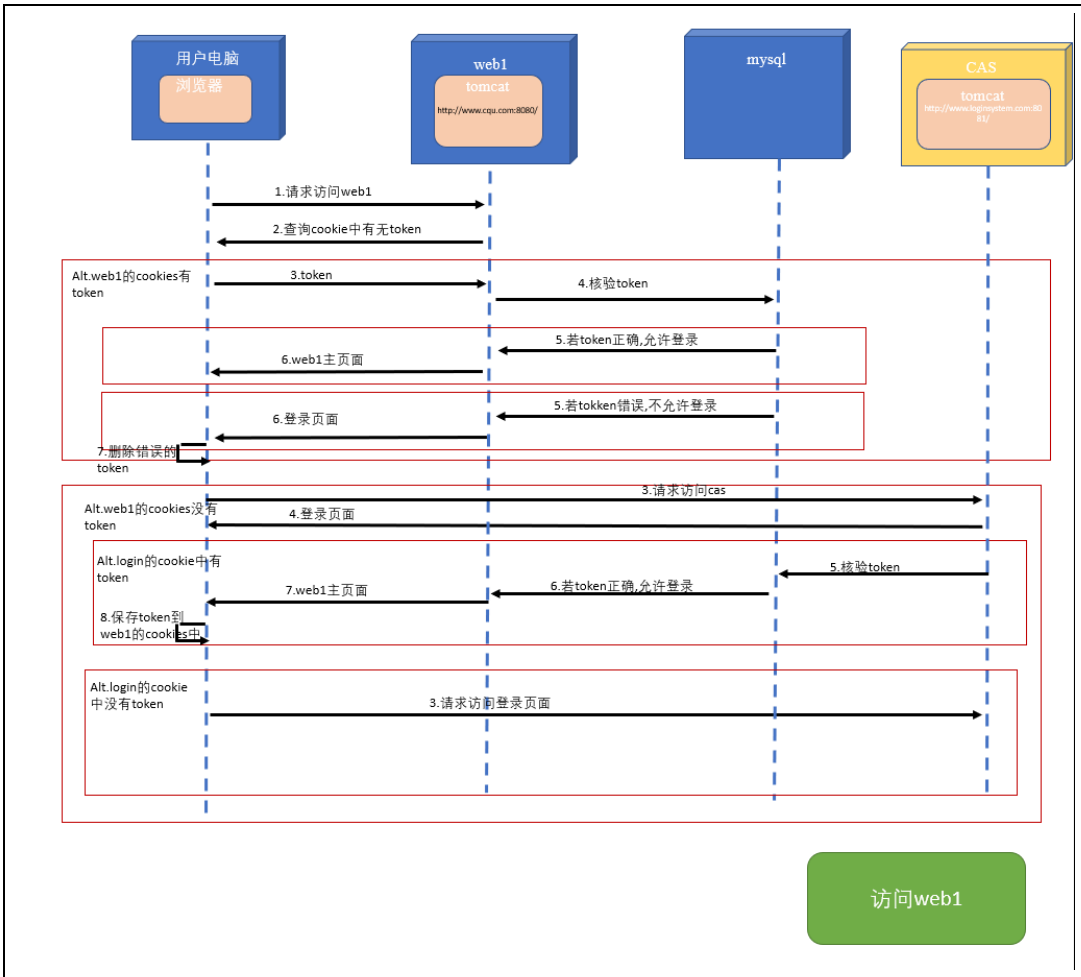

图 19:访问 web1 时序图

8. 核心源代码、截图

（1）如何在主页面显示已经登录的所有用户的信息和登录时间，首先要连接到调用数据库的 showAllUser 这一 post 请求，存到表格中

```
<%
    String targetUrl = "http://www.LoginSystem.com:8081/ShowAllUser";
    // 创建连接
    URL url = new URL(targetUrl);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    // 设置请求方法为 POST
    connection.setRequestMethod("POST");
    connection.setDoOutput(true);
    // 发送请求
    String[][] table;
    try (BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream(), "utf-8"))) {
        String response1 = br.readLine();
        System.out.println(response1);
        String[] records = response1.split(",");
        table = new String[records.length][3];
        for (int i = 0; i < records.length; ++i) {
            table[i][0] = records[i].split(" ")[2].split(" = ")[1];
            table[i][1] = records[i].split(" ")[4].split(" = ")[1];
            table[i][2] = records[i].split(" ")[8].split(" = ")[1];
        }
    }
    connection.disconnect();
```

Jsp 中绘制表头和表格，

```
<table border="1">
    <tr>
        <th>id</th>
        <th>用户名</th>
        <th>登录时间</th>
    </tr>
    for (int i = 0; i < table.length; i++) { %>
<tr>
    <td><%= table[i][0] %>
    </td>
    <td><%= table[i][1] %>
    </td>
    <td><%= table[i][2] %>
    </td>
</tr>
<% } %>
```

（2）提交登录的表单

```
<form action="/login?web=<%=web%>" method="post">
    <label for="username">用户名:</label><br>
    <input type="text" id="username" name="username"><br>
    <label for="password">密码:</label><br>
    <input type="password" id="password" name="password"><br>
    <input type="submit" value="登录">
</form>
```

处理该登录的请求，获取表单内容，

```
@WebServlet("/login")
public class LoginController extends HttpServlet {
    8 usages  guoluguodong
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String web = request.getParameter( s: "web");
        String username = request.getParameter( s: "username");
        String password = request.getParameter( s: "password");
        //创建JSON对象message，以便往前端响应信息
        HttpSession session = request.getSession(); // 创建session变量
        String message = null; // 定义message变量
```

尝试调用 service 查验用户名密码进行登录，

```
try{
    User loggedUser = UserService.getInstance().login(username, password);
```

Cookie 存储用户名和 token

```
if(loggedUser != null){
    Cookie cookieName = new Cookie( name: "username",loggedUser.getUsername());
    Cookie cookieToken = new Cookie( name: "token",loggedUser.getToken());
    response.addCookie(cookieName);
    response.addCookie(cookieToken);
```

如果该 post 还有参数 web,判断时跳转到 web1/2 还是主页

```
if(web==null)
    response.sendRedirect( s: "http://www.LoginSystem.com:8081/jsp/main.jsp");
else if(web.equals("web1"))
    response.sendRedirect( s: "http://www.CQU.com:8080?token="+loggedUser.getToken());
else if(web.equals("web2"))
    response.sendRedirect( s: "http://www.PKU.com:8082/?token="+loggedUser.getToken());
else{
    response.sendRedirect( s: "http://www.LoginSystem.com:8081/jsp/main.jsp");
}
return;
```

异常处理

```
        }else {
            message = "未登录 或者 用户名或密码错误！";
            session.setAttribute( s: "message", message);
            response.sendRedirect( s: "http://www.LoginSystem.com:8081/jsp/login.jsp");
        }
    }catch (SQLException e){
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json; charset=UTF-8");
        message="数据库操作异常";
        session.setAttribute( s: "message", message);
    }
}
```

（3）在 web1 发出退出登录请求到 cas

```
String token = request.getParameter( s: "token");
HttpSession session = request.getSession();
try {
    UserService.getInstance().logout(token);
    Cookie cookieName = new Cookie( name: "username",  value: null);
    Cookie cookieToken = new Cookie( name: "token",  value: null);
    response.addCookie(cookieName);
    response.addCookie(cookieToken);
    response.sendRedirect( s: "http://www.loginsystem.com:8081/jsp/login.jsp?web=main");
} catch (SQLException e) {
    throw new RuntimeException(e);
}
```

首先调用业务层的 service 退出登录，要将 web1 的 cookies 清理掉，再重定向到主页

由于 web1 发出的请求，删除的 cookie 是 web1 的，还要在 login.jsp 中加入如下代码，调用 clearcookie，清理 loginSystem 的 cookies

```
String logout = request.getParameter("logout");
String web = request.getParameter("web");
if (logout != null && logout.equals("true")) {
    response.sendRedirect("http://www.LoginSystem.com:8081/clearcookie");
} else {
```

（4）UserService 是处理用户的业务层，单例模式，userService 作为单例

```
public final class UserService {
    no usages
    private UserDao userDao = UserDao.getInstance();
    1 usage
    private static UserService userService = new UserService();
```

getInstance()用于获取单例

```
public static UserService getInstance() { return UserService.userService; }
```

核心业务包括下面四个，验证 token、登录、登出、查询所有登录用户

```
public User valToken(String token) throws SQLException {
    return UserDao.getInstance().valToken(token);
}

1 usage  guoluguodong
public User login(String username, String password) throws SQLException{
    return UserDao.getInstance().login(username, password);
}
1 usage  guoluguodong
public void logout(String token) throws SQLException{
    UserDao.getInstance().logout(token);
}
1 usage  guoluguodong
public ArrayList<User> showAllUser() throws SQLException {
    return UserDao.getInstance().showAllUser();
}
```

（5）UserDao 是数据获取层，以 valToken 为例，连接数据库

```
try{
    //获得数据库连接对象
    Connection connection = JdbcHelper.getConn();
```

构建 sql 语句

```
Statement stmt = connection.createStatement();
//执行SQL查询语句并获得结果集对象
String findByUsername_sql = "SELECT * FROM USER WHERE token=?";
//在该连接上创建预编译语句对象
PreparedStatement preparedStatement = connection.prepareStatement(fi
//为预编译参数赋值
preparedStatement.setString( parameterIndex: 1,token);
```

执行查询操作

```
ResultSet resultSet = preparedStatement.executeQuery();
```

返回结果并关闭连接

```
while (resultSet.next()) {
    user = new User(resultSet.getInt( columnLabel: "id"), resultSet.getS

}
connection.close();
```

四、实验结果及分析和（或）源程序调试过程（界面截图和文字）、实验总结与体会

（一）实验结果及分析

域名映射关系

127.0.0.2　　www.LoginSystem.com

127.0.0.3　　www.CQU.com

127.0.0.4　　www.PKU.com

(1) CAS 域名：http://www.loginsystem.com:8081/

CAS 登录页面 http://www.loginsystem.com:8081/jsp/login.jsp?web=main

CAS 主页面 http://www.loginsystem.com:8081/jsp/main.jsp

(2)web1 域名：http://www.cqu.com:8080/

web1 页面 http://www.cqu.com:8080/jsp/CquMainPage.jsp

(2)web2 域名：http://www.pku.com:8082/

Web2 页面 http://www.pku.com:8082/jsp/PkuMainPage.jsp

1.启动三个服务器

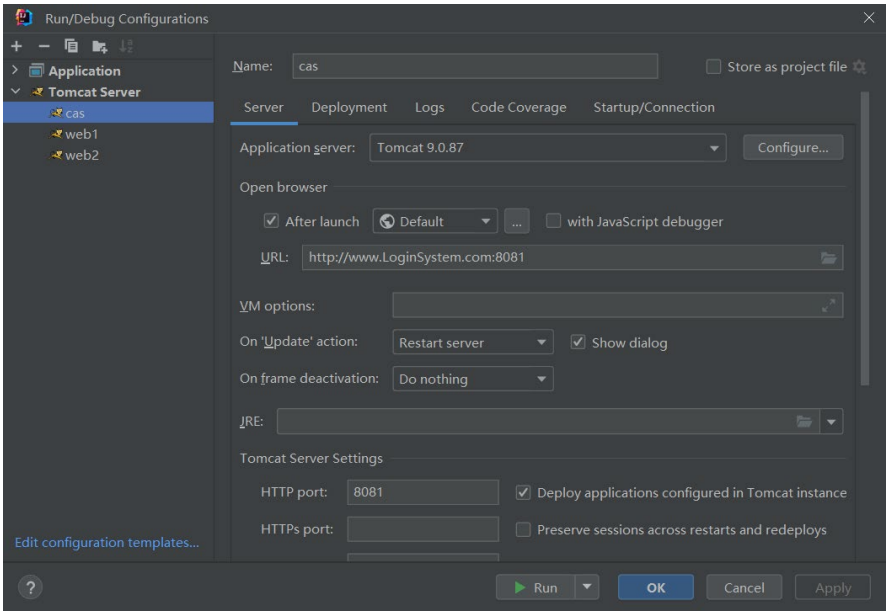

图 20:运行配置

2. 输入 http://www.loginsystem.com:8081/，会跳转到登录页面

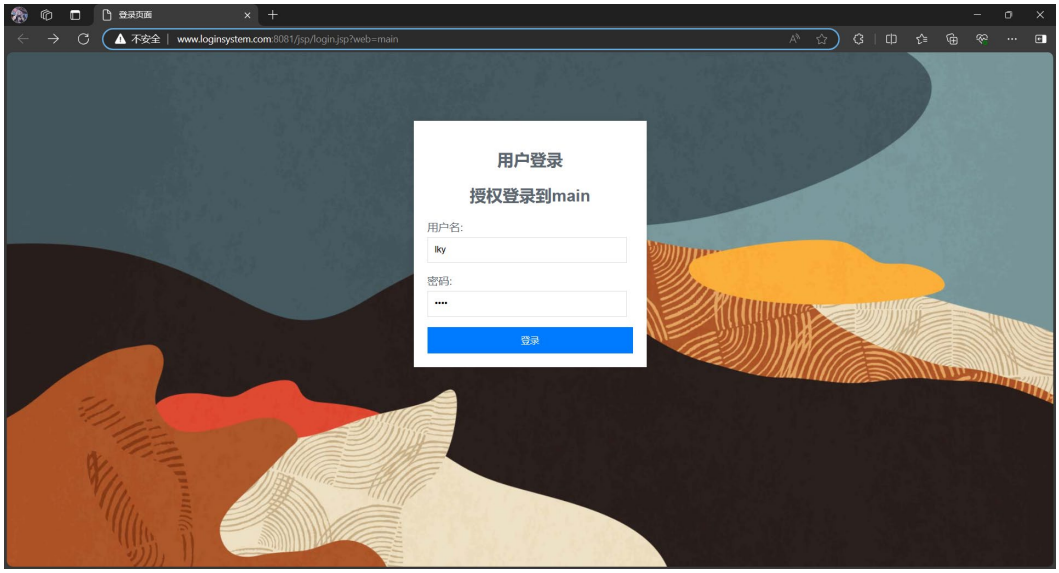

图 21:登录页面

3. 输入账号密码

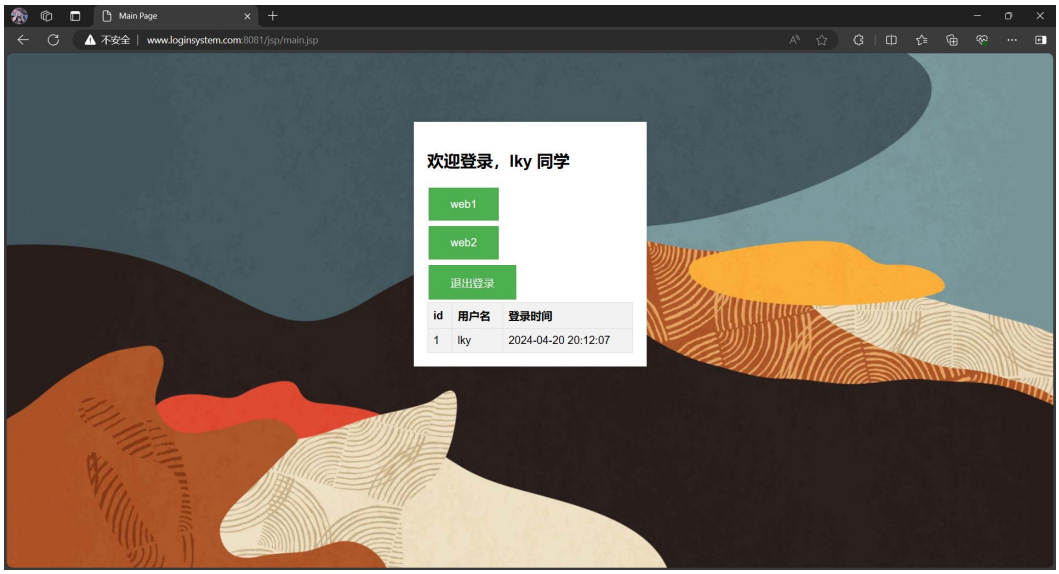

图 22：登录后主页面

4. 此时在 cas 已经登录，输入 http://www.cqu.com:8080/ 和 http://www.pku.com:8082/都已经登录

图 23:web1 系统和 web2 系统

5. 主页面分别点击 web1 和 web2 都能直接跳转，web1 或者 web2 也可以点击跳转

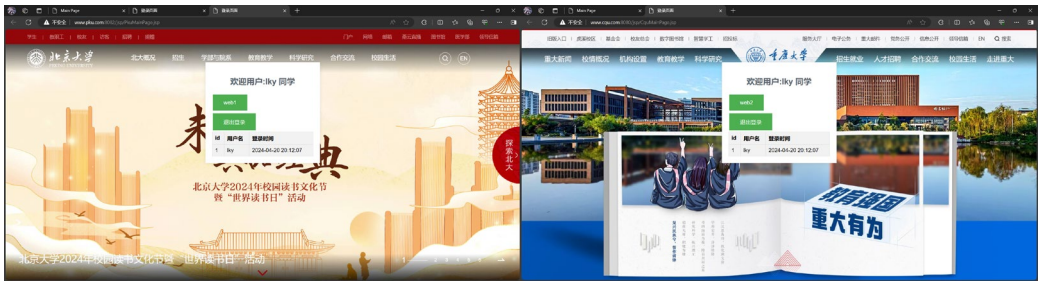

图 24:web1 系统和 web2 系统

6. 任意一个系统点击退出登录，其他页面 f5 刷新均会返回登录页面。

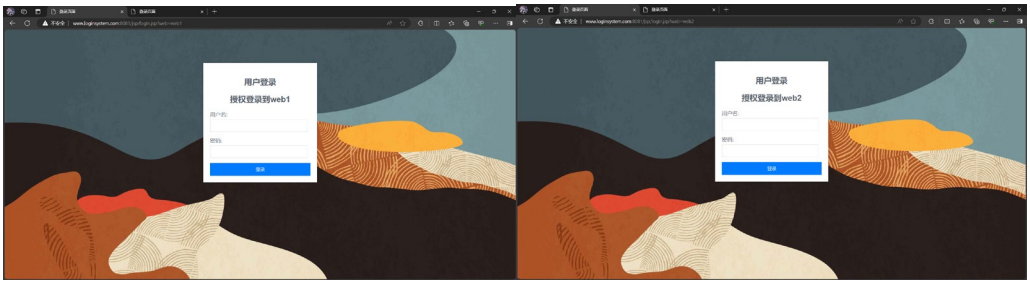

图 25：单点注销，页面 f5 刷新均会返回登录页面

7. 如果登录多个用户，就可以在每个页面都可以看到已登录用户，如果有新用户登录了，已登录用户刷新下就可以看到。

图 26：显示当前登录用户

8. 如果没有登录，就输入了 web1：http://www.cqu.com:8080/，会跳回登录页面，web=web1 作为参数，若是从 web1 来的，再次登录后会直接跳转到 web1

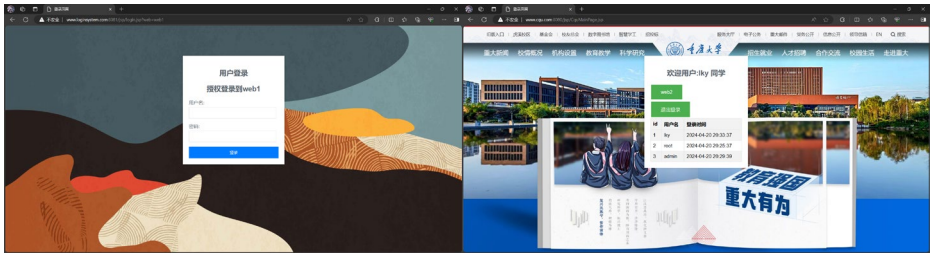

图 27：web1 请求 cas 授权

（二）实验总结与体会

总结和体会

学习新知识总是一次宝贵的经历。单点登录（SSO）可以提高用户体验并简化身份验证流程。通过项目，我不仅掌握了 SSO 的工作原理，还学到了如何设计和实现一个安全可靠的系统。

①身份验证和授权：SSO 不仅涉及用户的身份验证，还需要授权机制来确保用户只能访问他们有权限的资源。

②令牌管理：SSO 使用令牌来跟踪用户的登录状态。了解了不同类型的令牌（例如 JWT、OAuth2 等）以及它们的优缺点。

③安全性：SSO 系统必须具备高度的安全性，以防止恶意用户或攻击者利用漏洞。

④单点注销：学习单点注销用户体验。

遇到的问题和解决：

（1）项目出现二级.pom 时，tomcat 不能直接找到该运行的 webapp 的.war 包

解决：在 Project Structure 中的的 artifacts 中点击+，选择 web application exploded，选择 from modules 即可找到并创建 war

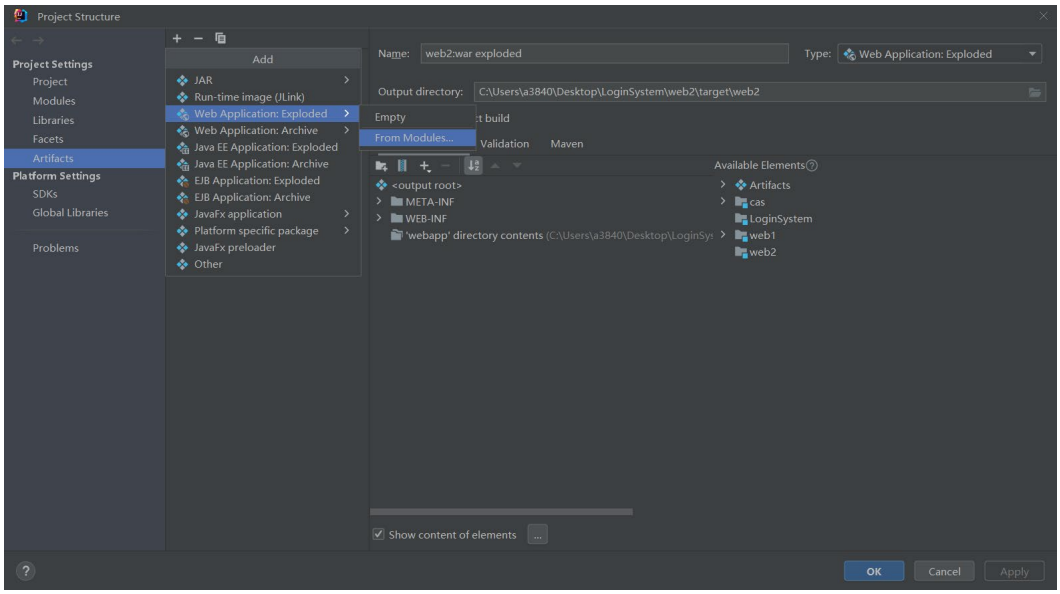

图 28：创建 war

再在 Deployment 中导入 war，即可在 tomcat 上运行

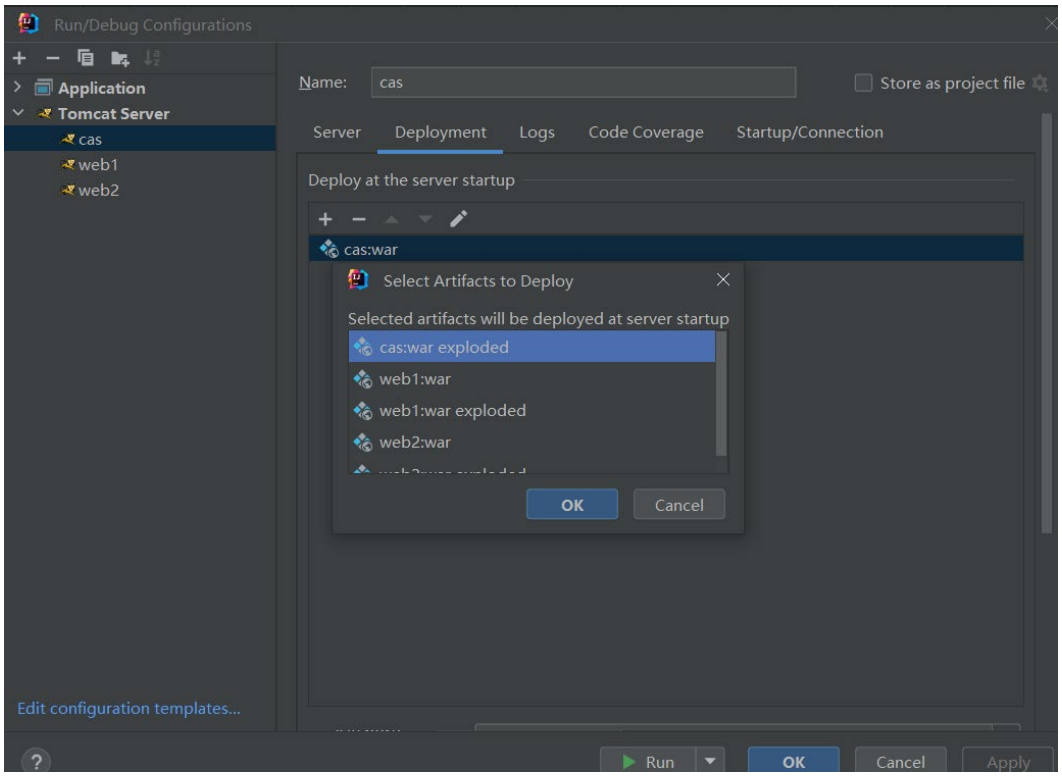

图 29:在服务器上部署 war

（2）域名将 ip 地址映射到文本

解决：以管理员模式运行记事本，打开 C:\Windows\System32\drivers\etc\hosts，在末尾添加映射关系即可

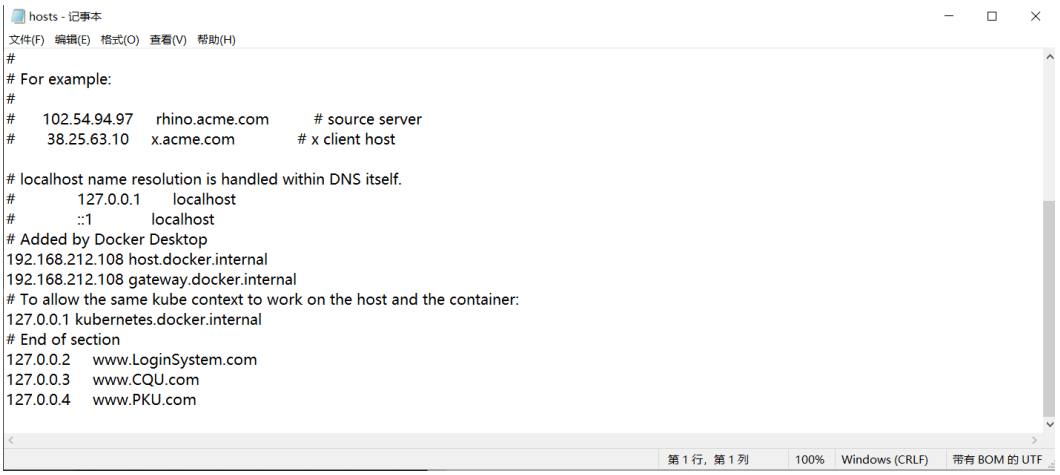

图 30：ip 地址映射到文本

（3）tomcat 不能同时运行多个 war，比方说，运行了 cas 后，再运行 web1，就会导致 cas 被终止。

解决方案：复制多个 tomcat 文件，并创建对应的系统环境变量，这样就可以用多个 tomcat 同时运行多个 war

复制多个 tomcat 文件

图 31：复制多个 tomcat 文件

创建对应的系统环境变量

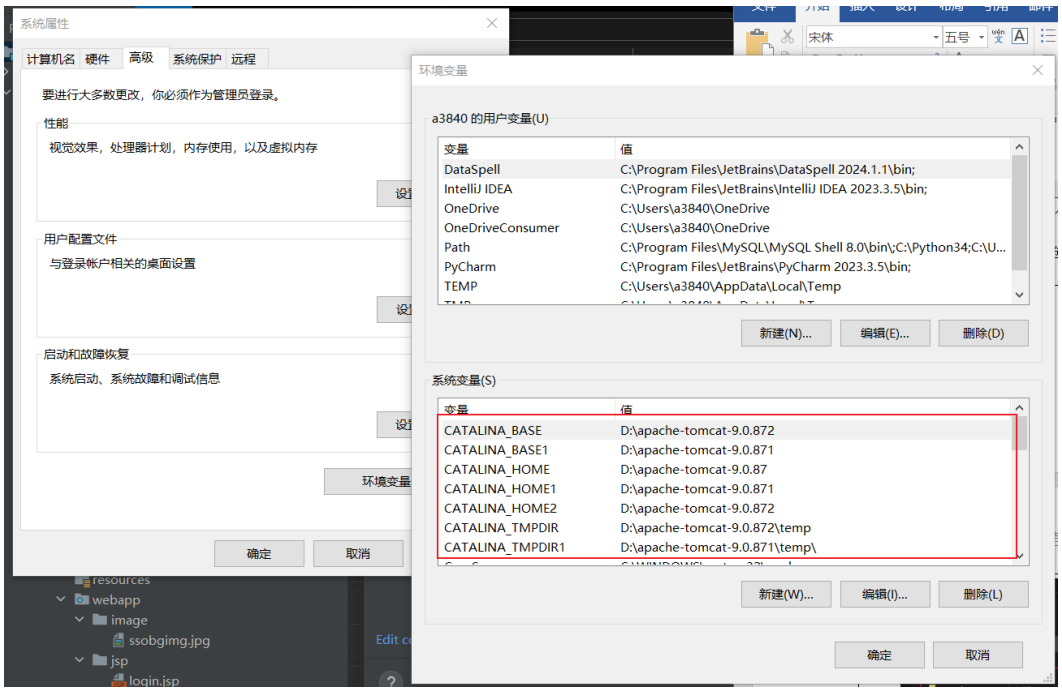

图 32：创建环境变量

到 tomcat 的 bin 路径下修改 catalina.bat 和 startup.bat 所有涉及到环境变量的地方，例如，将 CATALINA_BASE 改成 CATALINA_BASE1

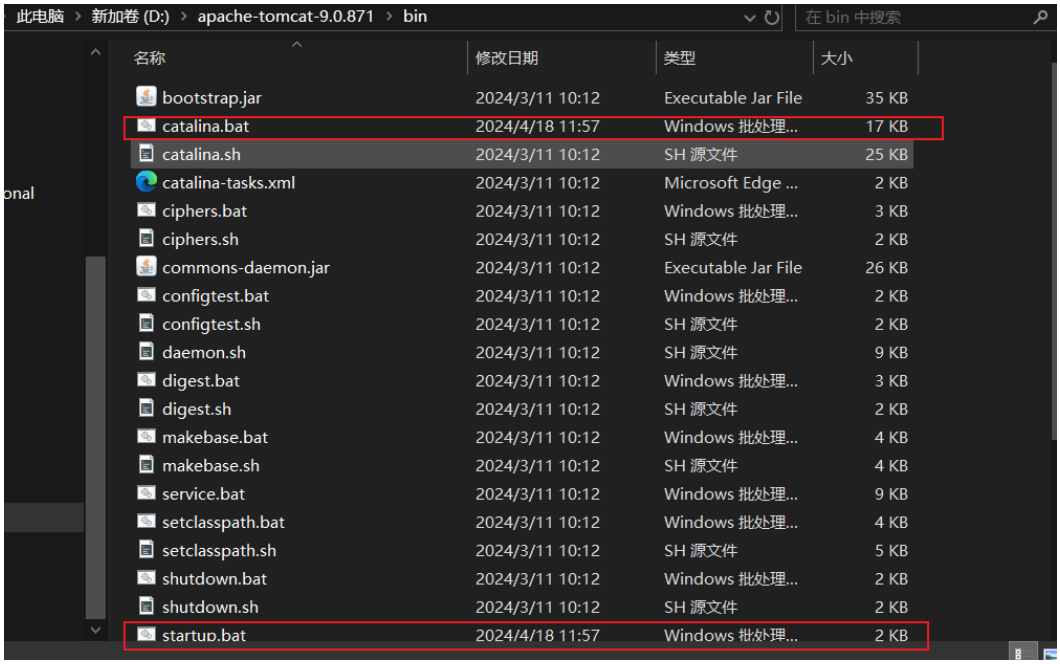

图 33:tomcat 待修改的文件

（4）不同的 ip 地址不共享 cookie，登录了一个系统后，希望另一个系统刷新后也能登录,例如，在登录了 cas 后，希望 web1 输入域名自动登录

解决方案：所有页面在未登录状态下，都要先查 LoginSystem 的 token，因为这是登录的唯一入口，一定有 token，web1 获取并验证 token 后，写入本 ip 的 cookies，再重定向到 web1mainPage

（5）不同的 ip 地址不共享 cookie，退出登录了一个系统后，希望另一个系统刷新后也能退出登录并返回登录页面,例如，在退出登录了 cas 后，希望 web1 刷新页面后登录失效。

解决方案：所有页面在登录状态下，重新访问，都要先验证 token，如果有系统退出登录都修改数据库的 token 值为 null，使得其他页面验证 token 不通过.

（6）如何使得 login 后就不能访问登录页，会自动跳转主页。以及 web1 系统中所有页面都有登录的记录

解决：在 cookies 中存储 token，login 后，LoginSystem.com 域名的会话 cookies 中有 token 值且能通过验证，就会从登录页重定向到主页

（7）如何使得未登录状态下访问 web1，就会跳转到 login，登录后直接跳转到 web1，而非主页。

解决方案：在 web1 重定向到的 login 后面，加 post 参数 web=web1,这样登录后重定向到 web1，没有就默认到 mainPage.