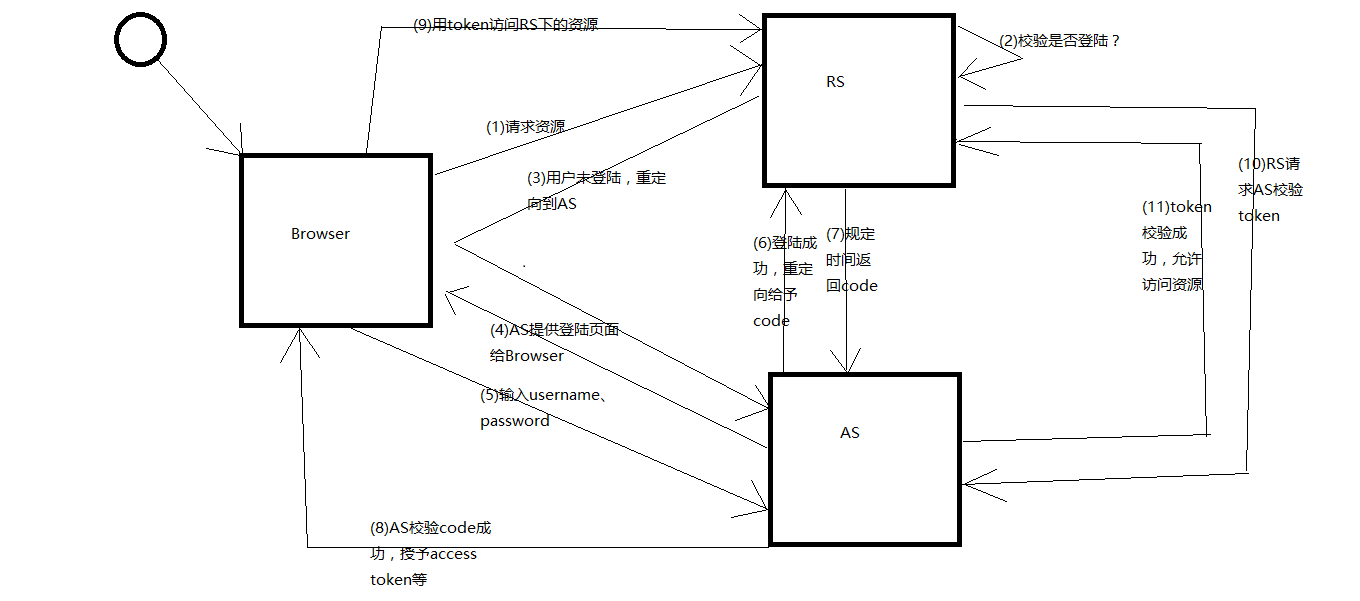
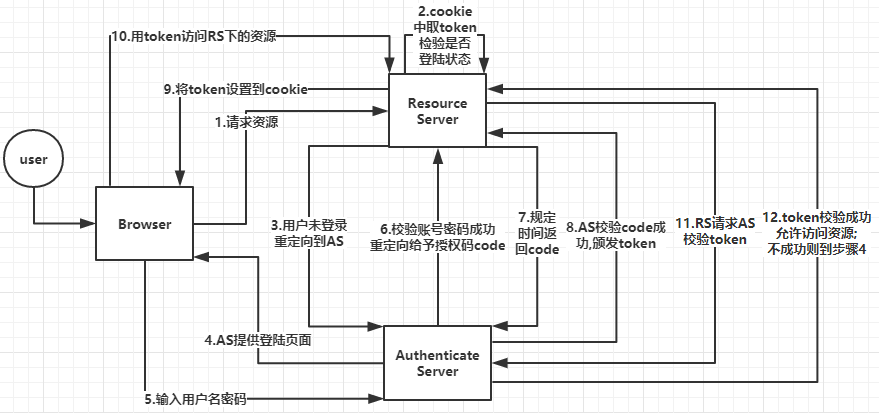
单点登录系统

V1.0

# OAuth2.0基本思路





客户端必须得到用户的授权（authorization grant），才能获得令牌（access token）。OAuth 2.0定义了四种授权方式。

授权码模式

简化模式

密码模式

客户端模式

RS：资源服务器

AS：授权服务器

code：授权码

Access Token：访问令牌

Refresh Token：刷新令牌

ClientId：客户端id

ClientSecret：客户端密钥

RedirectURL：重定向地址

ReturnURL：目标地址

# 申请接入

|  |  |
| --- | --- |
| 描述 | 向SSO服务器申请接入第三方应用 |
| URL | http://IP地址:端口号/sso-server/common/client/switch-on-sso.jsp |
| 示例 | http://127.0.0.1:8080/sso-server/common/client/switch-on-sso.jsp |
| 返回值 | 客户端ID |



# 审核并同意接入

|  |  |
| --- | --- |
| 描述 | SSO服务器管理人员审核并同意第三方系统接入 |
| URL | http://IP地址:端口号/sso-server/common/login.jsp |
| 示例 | http:// 127.0.0.1:8080/sso-server/common/login.jsp |
| 参数 | username //admin  password //admin |



# 搭建工程

## 步骤

1. 导入sso-client包（并Add to Build Path）
2. 编写servlet继承并重写servlet中的方法
3. 配置filter和servlet

## 示例：

### 解决cookie跨域

配置filter（不用编写，只需在web.xml中配置）

**<filter>**

**<filter-name>P3PFilter</filter-name>**

**<filter-class>com.kongbig.web.filter.P3PFilter</filter-class>**

**</filter>**

**<filter-mapping>**

**<filter-name>P3PFilter</filter-name>**

**<url-pattern>/\*</url-pattern>**

**</filter-mapping>**

### 实现单点的servlet（OAuthServlet）

编写一个servlet继承OAuthServlet实现单点登陆功能：

**重写认证成功后的方法，将用户信息共享到session**

**public class ClientOauthServlet extends OAuthServlet {**

**@Override**

**public void loginSuccess(HttpServletRequest request, HttpServletResponse response,**

**AccessTokenModel accessTokenModel) {**

**User user = new User("username", "password");**

**request.getSession().setAttribute("user", user);**

**System.out.println("SSO登陆验证成功后的操作...");**

**}**

**@Override**

**public void loginError(HttpServletRequest request, HttpServletResponse response) {**

**System.out.println("SSO登陆验证失败后的操作...");**

**}**

**}**

配置servlet：（pattern不可改）

**<servlet>**

**<servlet-name>OAuthServlet</servlet-name>**

**<servlet-class>com.kongbig.web.servlet.ClientOauthServlet</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>OAuthServlet</servlet-name>**

**<url-pattern>/sso/oauth</url-pattern>**

**</servlet-mapping>**

### 实现退出的servlet（LogoutServlet）

编写一个servlet继承LogoutServlet实现退出功能：

**退出成功后，将session中的用户信息重新设置为null**

**public class ClientLogoutServlet extends LogoutServlet {**

**@Override**

**public void logoutError(HttpServletRequest request, HttpServletResponse response) {**

**request.getSession().setAttribute("user", null);**

**System.out.println("退出失败后的操作...");**

**}**

**@Override**

**public void logoutSuccess(HttpServletRequest request, HttpServletResponse response) {**

**System.out.println("退出成功后的操作...");**

**}**

**}**

配置servlet：（pattern不可改）

**<servlet>**

**<servlet-name>LogoutServlet</servlet-name>**

**<servlet-class>com.kongbig.web.servlet.ClientLogoutServlet</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>LogoutServlet</servlet-name>**

**<url-pattern>/sso/logout</url-pattern>**

**</servlet-mapping>**

### 安全拦截器例子

1．/sso/oauth表示认证的情况，不拦截。

2．要判断session中是否有用户信息才放行（可个性化）

**public class SecurityFilter implements Filter {**

**private String ignorePattern = "";**

**@Override**

**public void init(FilterConfig filterConfig) throws ServletException {**

**ignorePattern = filterConfig.getInitParameter("ignorePattern");**

**}**

**@Override**

**public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)**

**throws IOException, ServletException {**

**HttpServletRequest request = (HttpServletRequest) req;**

**HttpServletResponse response = (HttpServletResponse) resp;**

**if (isIgnore(request)) {**

**chain.doFilter(req, resp);**

**} else if (request.getRequestURI().contains("/sso/oauth")) {**

**// 需要认证的情况，不拦截。**

**chain.doFilter(req, resp);**

**} else if (request.getSession().getAttribute("user") != null) {**

**// 登陆成功才转发到目标地址**

**chain.doFilter(req, resp);**

**} else {**

**// 请求认证**

**OAuthUtil.*oauth*(request, response);**

**}**

**}**

**@Override**

**public void destroy() {**

**}**

**}**

**<filter>**

**<filter-name>SecurityFilter</filter-name>**

**<filter-class>com.kongbig.web.filter.SecurityFilter</filter-class>**

**<init-param>**

**<param-name>ignorePattern</param-name>**

**<param-value>\*.css|\*.js|\*.png|\*.jpg|\*.ttf|\*.woff|\*.gif</param-value>**

**</init-param>**

**</filter>**

**<filter-mapping>**

**<filter-name>SecurityFilter</filter-name>**

**<url-pattern>/\*</url-pattern>**

**</filter-mapping>**

### 请求sso-client进行认证的工具类

**public class OAuthUtil {**

**public static void oauth(HttpServletRequest request, HttpServletResponse response)**

**throws ServletException, IOException {**

**String uri = request.getServletPath();**

**StringBuilder sb = new StringBuilder();**

**sb.append("/sso/oauth?returnURI=" + uri);**

**request.getRequestDispatcher(sb.toString()).forward(request, response);**

**}**

**}**