

Single Sign On - SSO

- [Joget SSO on G Suite](#)
- [Joget SSO with Keycloak using SAML](#)
- [Joget SSO with Azure Active Directory using SAML](#)
- [Joget SSO to Active Directory with Kerberos](#)
- [OpenID Connect](#)
- [Joget SharePoint SSO Integration](#)
- [Login an User Programmatically](#)

User logs in to external system / identity provider and implicitly gains access to Joget without being prompted to login again.

Joget SSO on G Suite

Please see [Joget Low Code Application Platform for G Suite](#)

Joget SSO with Keycloak using SAML

Please see [Joget SSO with Keycloak using SAML](#).

Joget SSO with Azure Active Directory using SAML

Please see [Joget SSO with Azure Active Directory using SAML](#).

Joget SSO to Active Directory with Kerberos

Please see [Joget SSO to Active Directory with Kerberos](#).

OpenID Connect

Please see [OpenID Connect Directory Manager Plugin](#).

Joget SharePoint SSO Integration

Please see [Joget SharePoint SSO Integration](#).

Login an User Programmatically

- You can build your own [Web Service Plugin](#) to perform custom SSO implementation.

```

import org.joget.apps.workflow.security.WorkflowUserDetails;
import org.joget.directory.model.service.DirectoryManager;
import org.joget.workflow.model.service.WorkflowUserManager;
import org.joget.apps.app.service.AppUtil;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.joget.directory.model.User;
import org.joget.workflow.util.WorkflowUtil;
import org.springframework.security.core.context.SecurityContextHolder;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpServletRequest;
import org.springframework.security.web.savedrequest.HttpSessionRequestCache;
import org.springframework.security.web.savedrequest.SavedRequest;

//Get service beans
DirectoryManager dm = (DirectoryManager) AppUtil.getApplicationContext().getBean("directoryManager");
WorkflowUserManager workflowUserManager = (WorkflowUserManager) AppUtil.getApplicationContext().getBean(
("workflowUserManager"));

//Login as "clark"
String username = "clark";
User user = dm.getUserByUsername(username);

if (user != null) {
    WorkflowUserDetails userDetails = new WorkflowUserDetails(user);

    //Generate an authentication token without a password
    UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(userDetails.getUsername(),
"", userDetails.getAuthorities());
    auth.setDetails(userDetails);
    //Login the user
    SecurityContextHolder.getContext().setAuthentication(auth);
    workflowUserManager.setCurrentThreadUser(user.getUsername());

    // generate new session to avoid session fixation vulnerability
    HttpServletRequest httpRequest = WorkflowUtil.getHttpServletRequest();
    HttpSession session = httpRequest.getSession(false);
    if (session != null) {
        SavedRequest savedRequest = (SavedRequest) session.getAttribute("SPRING_SECURITY_SAVED_REQUEST");
        session.invalidate();
        session = httpRequest.getSession(true);
        if (savedRequest != null) {
            session.setAttribute("SPRING_SECURITY_SAVED_REQUEST", savedRequest);
        }
    }
}
}

```

Please note that if you are adding these code in a filter, you will need to store the SecurityContext to session.

```

//Store SecurityContext to session to avoid spring security to clean it.
session.setAttribute("SPRING_SECURITY_CONTEXT", SecurityContextHolder.getContext());

```