Single Sign On (SSO)

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

IMPORTANT NOTICE
The sample code provided below using Javascript that exposes the user credential information is not a good security best practice. Please do not put this into practice.

Using JSON API

- Using '/web/json/directory/user/sso' JSON API.
- You are allowed to call this method using JSON API Authentication or
- Directly passes the username and password with "username" and "password" parameters respectively shown in following example.

```
<script>
    $(document).ready(function(){
                $.ajax({
            type: "POST",
            url: 'http://localhost:8080/jw/web/json/directory/user/sso?callback=callbackFunction',
            data: {
                username: 'admin',
                password: 'admin'
            },
            success: function(res) {
                console.log("username (" + res.username + ") is " + ((res.isAdmin !== undefined && res.isAdmin
=== "true")?"admin":"not an admin"));
            },
            dataType: "json"
        });
    });
</script>
```

Using Basic Http Authentication with JSON API

- Since V4, Joget Workflow is supported Basic HTTP Authentication in JSON API authentication, you can passing the credentials in the header.
- **Example**: Assuming the username and password required is "user1" and "password1" respectively, we can set the Basic Auth header to the JSON API using following jQuery script.

```
<script>
    $(document).ready(function(){
                $.ajax({
            type: "POST",
            url: 'http://localhost:8080/jw/web/json/directory/user/sso',
            beforeSend: function (xhr) {
                xhr.setRequestHeader ("Authorization", "Basic dXNlcjE6cGFzc3dvcmQx");
            },
            success: function(res) {
                console.log("username (" + res.username + ") is " + ((res.isAdmin !== undefined && res.isAdmin
=== "true")?"admin":"not an admin"));
            },
            dataType: "json"
        });
    });
</script>
```

Using Javascript API

- Includes the jQuery & util.js libraries.

- Using the AssignmentManager.login method for SSO.
- Perform actions in callback of successful login.

```
<script type="text/javascript" src="http://localhost:8080/jw/js/jquery/jquery-1.9.1.min.js"></script>
<script type="text/javascript" src="http://localhost:8080/jw/js/json/util.js" ></script>

<script type="text/javascript" >
$(document).ready(function(){
    var loginCallback = {
        success : function(response){
            if(response.username != "roleAnonymous"){
                alert("login successfully");
            }else{
                alert("login fail");
            }
        }
    };
    AssignmentManager.login('http://localhost:8080/jw', 'admin', 'admin', loginCallback);
});
</script>
```

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 userDetail = new WorkflowUserDetails(user);

    //Generate an authentication token without a password
    UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(userDetail.getUsername(),
"", userDetail.getAuthorities());
    auth.setDetails(userDetail);
    //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_KEY");
        session.invalidate();
        session = httpRequest.getSession(true);
        if (savedRequest != null) {
            session.setAttribute("SPRING_SECURITY_SAVED_REQUEST_KEY", 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());
```