Session Management

In This Topic

In XperienCentral there are three different sessions that play a role in Session Management:

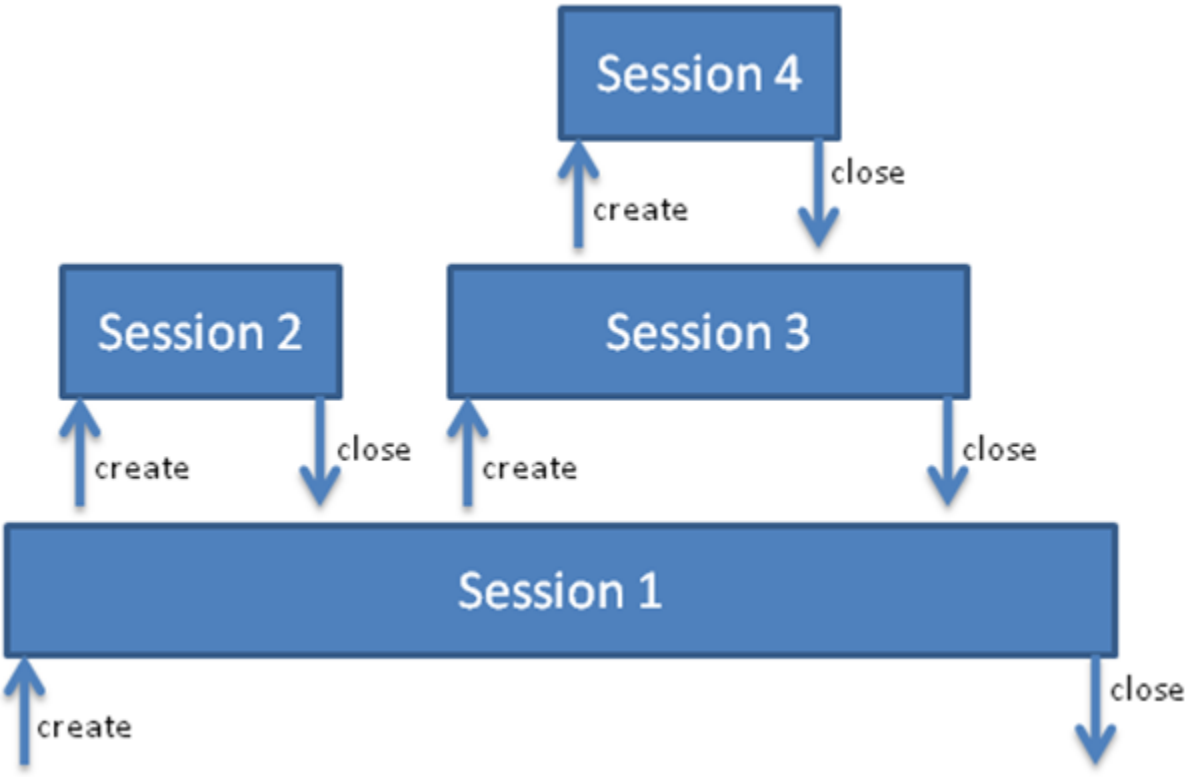
Session Interface	Description
`nl.gx.webmanager.foundation.Session`	XperienCentral specific session. Acts as a wrapper of XperienCentral specific context information and the JCR and HTTP sessions.
`javax.jcr.Session`	JCR session, needed to read and write from and to the JCR.
`javax.servlet.http.HttpSession`	Website visitor session (see Java API documentation)

Session Management is handled by the Session Manager service which only provides getters and instantiation methods for the first session type; the `nl.gx.webmanager.foundation.Session`. Because this is a wrapper around the other two sessions, you always have access to all three sessions if you have access to this session. The major purpose of the session is to define authorization. Authorization depends on the roles associated with the user stored in the session.

Session Stack

When an editor is working in the Workspace, the sessions are created automatically by the XperienCentral framework. For each request, XperienCentral identifies the user according to the cookie sent, together with the request, and creates a new XperienCentral session. This XperienCentral session is put on the top of the “session stack”. When the response is sent back to the client, this session is removed from the stack, therefore this session exists during the complete lifetime of the request.

During this lifetime, a second session can be created, which is again put on top of the session stack. The component that created the session is also responsible for closing the session. The purpose of using nested sessions is that actions performed within a specific session can be undone separately. The image below shows an example of a session stack:



- Session 1 is created and closed automatically by the framework
- Plugin 2 creates its own session 2 and closes it afterwards
- Plugin 3 creates its own session 3 and invokes a service from plugin 4
- The service in plugin 4 creates its own session 4 and closes it afterwards
- The session 3 is closed by plugin 3
- The framework automatically closes session 1

Back to Top

Retrieving and Creating Sessions

In some cases you can use an active session from the top of the stack created by another plugin or by the framework. In that case, simply invoke `SessionManager.getActiveSession()` to retrieve that active session. You should never close this session yourself, since you are not the one who created it. Leave this up to the creator of the session.

Active sessions will usually be available in element, panel and media item components since the controllers of these components are triggered by an editor who is logged in to XperienCentral. In other use cases, like testbundles or scheduled jobs, no active session will be available.

To create a session with login credentials, the best way is to define the login credentials in configuration entries managed by the Configuration Management service. The administrator can tune the authorization that is actually needed by that particular user.

To create the session, the `SessionManager.createSession(string, string)` method can be used. The code example below shows an example of creating a session for user USERNAMEKEY on webinitiative with ID WEBSITEKEY:

```
private Session login() {
   String website = myConfigService.getParameter(WEBSITEKEY);
   Session session = mySessionManager.createSession(website, "username");
   return session;
}
```

Since R36 an alternative option is available. In R35 and earlier a common code structure used to ensure the presence of a Session was somewhere along the lines of:

```
boolean sessionCreated = false;
Session session = mySessionManager.getActiveSession();

if (session == null) {
        session = mySessionManager.createSession();
        sessionCreated = true;
}

try {
// your logic here
...
} catch (Exception e) {
        // Handle the exception
} finally {
        if (sessionCreated) {
                session.close();
        }
}
```

The example above leads to a of duplicate code, both in XperienCentral and custom plugins. In R36 the SessionWrapper was introduced, which takes care of the Session creation for you. The example above can be changed to this:

```
try (SessionWrapper sessionWrapper = new SessionWrapper(mySessionManager)) {
        Session session = sessionWrapper.getSession();
        // your logic here
} catch (IOException e) {
    // Handle the exception
}
```

Please note that the getSession() should be called outside of the try-with-resources statement. This is because the SessionWrapper should be in control of closing the session. If you place the sessionWrapper.getSession() within the try-with-resources the session might be auto-closed, leading to errors.

Back to Top