Getting Sessions

Using org.openntf.domino.utils.Factory

The standard way to get a Session is using `Factory.getSession(SessionType)`. `Factory.getSession()` still works for getting the current session, but is discouraged in favour of more explicit usage of SessionTypes. There are a number of standard SessionTypes, the most common developers use are `SessionType.CURRENT` and `SessionType.NATIVE` (a session running as the server). XPages developers will be used to `SessionType.SIGNER` and `SessionTyp.SIGNER_FULL_ACCESS`, but in personal experience the server typically has relevant access in order for replication etc to work effectively.

The full list of SessionTypes is:

- CURRENT - A named Session for the current user
- CURRENT_FULL_ACCESS - A named Session for the current user but with full access
- SIGNER - A named Session for whichever Notes ID is defined as the current application"s signer
- SIGNER_FULL_ACCESS - A named Session for whichever Notes ID is defined as the current application"s signer with full access
- NATIVE - A Session based on the current server"s access
- TRUSTED - Returns a Trusted Session, not yet implemented
- FULL_ACCESS - A Session equivalent to Full Access Administration setting in Domino Administrator
- PASSWORD - A Session based on a specific Notes Client user ID and password

XPages Factory Setup

The Factory is automatically initialised with various current, signer and native SessionTypes. So you can just use the various `Factory.getSession(SessionType)` methods. In addition, global implicit variables are set up for every request. You will already be aware of some of these implicit variables that are set up by the XPages runtime - `session`, `sessionAsSigner`, `sessionAsSignerWithFullAccess` and `database`. If the "godmode" switch is enabled, ODA overrides these implicit variables to replace them with the corresponding ODA session and database objects. Otherwise, ODA creates `openSession`, `openSessionAsSigner`, `openSessionAsSignerWithFullAccess` and `openDatabase`.

ODA Starter Servlet

ODA Starter Servlet has an ODADataServlet class which manages all the setup and tear down of the Factory, adding `SessionType.CURRENT`. `SessionType.NATIVE` is automatically available. If other session types are required, you will need to set them up for each request here.

CrossWorlds

CrossWorlds uses a Filter to filter all HTTP requests for any web application, see `org.openntf.xworlds.core` bundle's `org.openntf.xworlds.appservers.webapp.XWorldsRequestFilter`. This uses configuration to set up Sessions for the current user and the application signer, see `org.openntf.xworlds.appservers.webapp.config.DefaultXWorldsApplicationConfig`. You can extend this class in your web application to change the base functionality.

OsgiWorlds

If you want to use ODA in a web servlet with Vaadin development, OsgiWorlds can be used. This uses an approach specifically for Vaadin development, extending the VaadinServlet class and its service method, see `org.openntf.osgiworlds.ODA_VaadinServlet`. This uses configuration to set up Sessions for the current user and the application signer, see `org.openntf.osgiworlds.DefaultDominoApplicationConfig`. Remember to register the servlet in the normal Vaadin way (the older method is in web.xml, the newer method is using @Annotations in the application's main UI class.

OSGi Web Applications

For a web application, if it uses a REST servlet as the entry point, you can use `ODADataServlet` from the ODA Starter Servlet as a basis to initialise the NotesThread and set up the Factory and ODA sessions. If it uses an HttpServlet entry point, you can use the `ODA_VaadinServlet` from OsgiWorlds as a basis (the VaadinServlet class extends `javax.servlet.Http.Servlet`).

DAS Servlet

The ODA graph REST API uses an extension to `com.ibm.domino.das.service.RestService`, implementing the REST service extension c`om.ibm.domino.das.service.IRestServiceExt` , see `org.openntf.domino.rest.service.ODAGraphService` class. This class's `beforeDoService` method initialises the thread for ODA and creates a Session as the current user using the `org.openntf.domino.xsp.`

`session.DasCurrentSessionFactory` class to retrieve the current user and current database from the Domino browser session. The `beforeDoService` method terminates the thread for ODA. The same classes can be used for custom REST servlets and was a basis for ODA Starter Servlet.