Plugins

Plugins everywhere! Joget Workflow provides an improved plugin architecture that supports both normal Java and dynamic OSGI plugins.

Plugin Types

Process Plugins

- **Process Tool** plugins to integrate with external systems
- **Process Participant** plugins for process-related participant mapping
- **Deadline** plugins for handling process-related deadlines

Form Plugins

- **Form Element** plugins to extend the types of fields available in a form
- **Form Validator** plugins to extend ways of validating form data
- **Form Data Binder** plugins to extend methods of loading and storing data in a form

Datalist Plugins

- **Datalist Binder** plugins to extend methods of loading data for a list (e.g., querying from an external SQL database)
- **Datalist Action** plugins to extend methods of executing an action on list items (e.g., deleting a record)
- **Datalist Format** plugins to extend ways of formatting column data

Userview Plugins

- **Userview Menu** plugins to extend the types of pages available in a userview
- **Userview Theme** plugins to change the UI design for a userview
- **Userview Permission** plugins to handle permissions and access rights in a userview

General Plugins

- **DirectoryManager** plugins to integrate users from external systems (e.g., Active Directory or LDAP)
- **Audit Trail** plugins for system-related functions (e.g., capture reporting data)
- **Hash Variable** plugins to extend support for processing hash variables

Development Approach

The method of developing a Joget plugin remains the same as described in Extending Functionality - Developing Plugins. However, you now have the option of using an additional feature: standard Java classes. Standard Java classes are read directly from the classpath.

The following table highlights the different approaches:

Dynamic OSGI Plugin	Standard Java Plugin
Build as an OSGI JAR bundle	Build as a standard Java JAR
Deploy JAR using the Manage Plugins in the Web Console	Make JAR available in the Java classpath (e.g., place it under WEB-INF/lib)
Supports dynamic loading/unloading/reloading without restarting	Requires restarting the JVM for deployment or changes
Runs in isolated mode, thus, preventing library version conflict with base libraries or other plugins	May cause library version conflicts with base libraries or other plugins
More difficult to develop and test due to OSGI configuration and isolation	Easier to develop and test using normal Java classes and libraries
	Java class path starts with **org.joget.**