



Using OpenDS as an Embedded Application

Neil A. Wilson

OpenDS Project Architect

Quick Overview of OpenDS

- Open source pure Java LDAPv3 directory server
- Intend to provide a high performance, highly-scalable, feature-rich, embeddable, extensible, user-friendly directory service (basically, better than every other server in every way)
- Virtually all information is publicly available online
 - > <https://opends.dev.java.net/> -- source repository, issue tracker, mailing lists, downloadable builds
 - > <https://www.opends.org/> -- documentation wiki, daily builds, source browser, QuickSetup installer

Embedding OpenDS

- As a pure Java application, OpenDS runs in a JVM
- It can run in the same JVM as another application
 - > Could be a Web application container
- It can be controlled (started, stopped, restarted, etc.) by another application in the same JVM.
- All functionality of the standalone server is also available when running in the same JVM as another application
 - > You also have the option to disable connection handlers so that it isn't externally accessible

Requirements for Embedding

- All necessary libraries need to be in the CLASSPATH.
 - > You should just need the lib/*.jar files.
- There needs to be an appropriate filesystem layout for OpenDS components.
 - > A simple approach is to just copy the OpenDS "standalone" filesystem layout to a subdirectory below your application.
 - > If you want to, you can explicitly specify the paths to certain components via properties (e.g., config file, schema directory, locks directory) or in the config file (e.g., DB directory, logs directory, etc.).

DirectoryEnvironmentConfig

- The DirectoryEnvironmentConfig class in the org.opens.server.types package lets you set a number of "environment" properties, including:
 - > Server root
 - > Config file and config handler class
 - > Schema directory
 - > Lock file directory
 - > Disable connection handlers
 - > Force daemon threads
 - > Install your own loggers outside of the configuration (can provide coverage for early startup operations)

EmbeddedUtils

- The EmbeddedUtils class in the org.opens.server.util package provides methods for controlling the server in an embedded manner:
 - > Determine whether the server is currently running
 - > Start the server (with a given DirectoryEnvironmentConfig)
 - > Stop the server (with a given reason)
 - > Restart the server (with a given reason and DirectoryEnvironmentConfig)
 - > Initialize OpenDS for client-side use

InternalClientConnection

- The InternalClientConnection class in the org.opens.server.protocols.internal package provides methods for performing internal operations
 - > You can get a root connection, or a connection authenticated as a given user.
 - > Add, compare, delete, extended, modify, and modify DN all work as expected.
 - > Searches can collect matching entries in a list or use a callback to provide entries as they're found
 - > Bind operations will work, but won't change the identity associated with the connection.
 - > Abandon and unbind aren't supported for internal operations and won't do anything

The @PublicAPI Annotation

- We have defined an @PublicAPI annotation (in the org.opens.server.types package) that can be used to denote what is contained in our public API
 - > Intended to be used to indicate which packages/classes/methods are available for third-party uses of OpenDS (e.g., embedding applications, OpenDS extensions like plugins and password validators).
 - > Defines a stability level to indicate how likely it is that the code will change in an incompatible way in the future.
 - > Defines mayInstantiate, mayExtend, and mayInvoke properties that indicate how the code can be used.
 - > Shows up in Javadoc documentation and is available via reflection for possible use by validation tools

Packages in the Public API

- The public API primarily consists of classes in the following packages:
 - > org.opens.server.api
 - > org.opens.server.api.plugin
 - > org.opens.server.config
 - > org.opens.server.interop
 - > org.opens.server.protocols.asn1
 - > org.opens.server.protocols.internal
 - > org.opens.server.types
 - > org.opens.server.types.operation
 - > org.opens.server.util

Possible Future Enhancements

- Provide improved documentation for third-party development (for developing server extensions as well as for embedded use)
- Provide an OpenDS Developer's Kit that can be used to facilitate third-party development using OpenDS (for developing server extensions as well as embedded use)
- Provide a mechanism for making internal configuration changes more convenient
- Provide a common API that can be used for either internal operations or external LDAP communication



Using OpenDS as an Embedded Application

Neil A. Wilson

neil.a.wilson@sun.com