Apache Felix Framework Usage Documentation

Apache Felix Framework Usage Documentation

Downloading the Framework

Go to the downloads page and download the latest Felix framework distribution.

Starting the Framework

Start the framework from the installation directory by typing:

```
java -jar bin/felix.jar
```

The framework launcher starts the framework and installs and starts all bundles contained in the `bundle` directory of the current directory. By default, the bundle directory contains shell-related bundles providing a textual user interface to interact with the framework. Bundles installed into the framework are copied into a bundle cache directory for subsequent executions. By default, the framework creates a cache directory, called `felix-cache`, in your current working directory. If you want to start the framework using a different bundle cache directory, you can do so like this:

```
java -jar bin/felix.jar <cache-path>
```

Where `<cache-path>` is the path you want to use as the bundle cache. If you specify a relative cache path, then it will be treated as relative to the current working directory.

Useful Information
Previous versions of the framework prompted for a profile name when executed. The profile name was used to create a directory inside `.felix/` in the user home directory. This approach allowed users to have different sets of bundles for different purposes, e.g., testing, production, etc. If this behavior is still desired, it is very easy to mimic. Modify `conf/config.properties` to include "`felix.cache.rootdir=${user.home}/.felix`". Now, if you start Felix with something like "`java -jar bin/felix.jar foo`", it will use "`${user.home}/.felix/foo/`" as the bundle cache directory, where "`${user.home}`" is automatically substituted with the appropriate system property by the launcher.

Framework Shell

The main way to interact with the framework is via the supplied Apache Felix Gogo shell. After starting the framework, type `help` into the shell to see the list of the available commands and `help <command-name>` to get help for a specific command.

Useful Information
In Gogo, command names are made up of two parts: `<scope>:<name>`. This is similar to a fully qualified class name in Java and is used to avoid naming collisions. If the `<name>` portion of the command is unique, then you only need to type it. If not, then you must

either type the full `<scope>:<name>` or arrange the scope search path accordingly.

To install bundles, use the `felix:install` command, which is described in more detail in the next sub-section. To list installed bundles, use the `felix:lb` command. To stop the framework type `stop 0` to stop the System Bundle; any installed bundles will automatically be reloaded (and potentially restarted) the next time you launch with the associated cache.

Installing Bundles

A bundle is the OSGi term for a component for the OSGi framework. A bundle is simply a JAR file containing a manifest and some combination of Java classes, embedded JAR files, native code, and resources. A bundle may provide some specific functionality for the user or it may implement a service that other bundles can use; bundles can only use functionality from other bundles through shared services and packages.

The Felix framework distribution comes with three bundles, which are located in the `bundle/` directory of the framework distribution installation directory. These bundles include the Gogo Runtime (core command processing functionality), Gogo Shell (text-based shell user interface), Gogo Command (basic set of commands), and Bundle Repository (a bundle repository service). In addition to these bundles, the bundle repository provides access to other bundles for easy installation. The bundle repository service provides a set of shell commands in the `obr:*` scope; refer to the Apache Felix OSGi Bundle Repository for more information.

Before installing any bundles, it is important to understand how bundles are manually deployed into the framework. Bundles are deployed in two stages; first they are installed, then they are started. To install a bundle use the `felix:install` shell command followed by a bundle URL. For example, to install a `bundle.jar` bundle you type:

```
felix:install file:/path/to/bundle/bundle.jar
```

Once a bundle is installed, it can then be started by using the `felix:start` command and the bundle identifier of the desired bundle. The `felix:lb` command is used to list installed bundles and to obtain the bundle's identifier. The following Felix shell session illustrates how to start the `bundle.jar` bundle:

```
g! install file:/path/to/bundle/bundle.jar
g! lb
START LEVEL 1
   ID|State      |Level|Name
    0|Active     |    0|System Bundle (3.0.0)
    1|Active     |    1|Apache Felix Bundle Repository (1.6.2)
    2|Active     |    1|Apache Felix Gogo Command (0.6.0)
    3|Active     |    1|Apache Felix Gogo Runtime (0.6.0)
    4|Active     |    1|Apache Felix Gogo Shell (0.6.0)
    5|Installed  |    1|Example Bundle (1.0.0)
g! start 5
Hello from Bundle 5.
g!
```

The `felix:stop` command is used to stop a bundle and the `felix:uninstall` command is used to remove a bundle from the bundle cache. As an alternative to using the `felix:install` and `felix:start` commands explicitly, it is also possible to install and start a bundle in one step by using the `felix:start` command with a bundle URL.

Bundles can be updated using the `felix:update` command. The update command allows you to specify an URL from which to retrieve the updated bundle, but if one is not specified it will try to update the bundle from the bundle's `Bundle-UpdateLocation` manifest attribute, if present, or the bundle's original location URL.

Useful Information
When you use `felix:update` or `felix:uninstall`, the changes appear to take effect immediately, but in reality the changes are only partially enacted. If a bundle is updated or uninstalled and it was exporting packages, these packages are not removed until the framework is refreshed using the `PackageAdmin` service. The Felix shell offers a convenient `refresh` command for this purpose. This is the correct behavior as defined by the OSGi specification.

For an introduction to writing bundles and services, refer to the Felix bundle tutorial.

Web Proxy Issues when Installing Bundles

If you use a proxy for Web access, then you may run into difficulty using the Gogo shell to install bundles from remote URLs. To remedy this situation, certain system properties must be set to make the framework work with your proxy. These properties are:

- `http.proxyHost` - the name of the proxy host.
- `http.proxyPort` - the port of the proxy host.
- `http.proxyAuth` - the user name and password to use when connecting to the proxy; this string should be the user name and password separated by a colon (e.g., `rickhall:mypassword`).

These system properties can be set directly on the command line when starting the JVM using the standard "`-D<prop>=<value>`" syntax or you can put them in the `lib/system.properties` file of your Felix installation; see the next section on configuring Felix for more information.

Bundle Auto-Deploy

To minimize the amount of configuration necessary to install bundles when you launch the framework, the Felix launcher uses the concept of an "auto-deploy" directory. The Felix launcher deploys all bundles in the auto-deploy directory into the framework instance during startup. By default, the auto-deploy directory is "`bundle`" in the current directory, but it can be specified on the command line like this:

```
java -jar bin/felix.jar -b /path/to/dir
```

Specifying an auto-deploy directory replaces the default directory, it does not augment it. The framework distribution is configured to install and start bundles in the auto-deploy directory. Both the location of the auto-deploy directory and the deployment actions performed can be controlled by the following configuration properties, respectively:

- `felix.auto.deploy.dir` - Specifies the auto-deploy directory from which bundles are automatically deploy at framework startup. The default is the `bundle/` directory of the current directory.
- `felix.auto.deploy.action` - Specifies the auto-deploy actions to be performed on the bundle JAR files found in the auto-deploy directory. The possible actions are `install`, `update`, `start`, and `uninstall`. If no actions are specified, then the auto-deploy directory is not processed (i.e., it is disabled). There is no default value for this property, but the default `config.properties` file installed with the Felix framework distribution sets the value to: `install, start`

The next section describes how to set and use configuration properties.

Configuring the Framework

Both the Felix framework and the launcher use configuration properties to alter their default behavior. The framework can only be configured by passing properties into its constructor, but the launcher provides a mechanism to configure the framework via a property file. The framework launcher reads configuration properties from `conf/config.properties`. This file uses standard Java property file syntax.

The launcher also supports setting system properties via the `conf/system.properties` file. This file is purely for convenience when you need to repeatedly set system properties when running the framework. While the framework itself does not look at system properties, the launcher does copy any framework configuration properties found in the system properties into the framework configuration map, also for your convenience.

It is possible to specify different locations for these property files using the `felix.config.properties` and `felix.system.properties` system properties when executing the framework. For example:

```
java -Dfelix.config.properties=file:/home/rickhall/config.properties -jar
bin/felix.jar
```

Configuration and system properties are accessible at run time via `BundleContext.getProperty()`, but configuration properties override system properties. For more information about available configuration properties, refer to the Apache Felix Framework Configuration Properties document. The Felix framework distribution contains a default `conf/config.properties`.

System Property Substitution

It is possible to use system properties to specify the values of properties in the `conf/config.properties` file. This is achieved through system property substitution, which is instigated by using `${<property>}` syntax, where `<property>` is the name of a system property to substitute. When the properties file is read, any such property values are substituted as appropriate. It is possible to have nested system property substitution, in which case the inner-most property is substituted first, then the next inner most, until reaching the outer most.

Configuring Bundles

Some bundles use properties to configure certain aspects of their behavior. It is a good idea, when implementing bundles, to parameterize them with properties where appropriate. To learn about the configuration options for specific bundles, refer to the documentation that accompanies them.

Bundle properties may also be defined in the `conf/config.properties` property file. Any property placed in this file will be accessible via `BundleContext.getProperty()` at run time. The property file uses the standard Java property file syntax (i.e., attribute-value pairs). For information on changing the default location of this file, refer to the section on configuring Felix.

Feedback

Subscribe to the Felix users mailing list by sending a message to users-subscribe@felix.apache.org; after subscribing, email questions or feedback to users@felix.apache.org.