Apache Felix File Install

Apache Felix File Install

File Install is a directory based OSGi management agent. It uses a directory in the file system to install and start a bundle when it is first placed there. It updates the bundle when you update the bundle file in the directory and, when the file is deleted, it will stop and uninstall the bundle.

File Install can do the same for configuration configuration files. This surprisingly simple bundle is very powerful because there are so many programs that work with the file system. For example:

- If you use Ant, you can just copy the resulting bundle to the watched directory.
- You can download bundles from the web and directly install them without any extra effort.
- You can easily drag and drop bundles in and out of the framework.

Setup

The bundle runs on any framework. For its configuration, it will use the following system properties:

Property	Default	Description
`felix.fileinstall.poll`	2000 ms	Number of milliseconds between 2 polls of the directory
`felix.fileinstall.dir`	./load	The name of the directory to watch. Several directories can be specified by using a comma separated list.
`felix.fileinstall.log.level`	1	Threshold of logging information
`felix.fileinstall.bundles.new.start`	true	Automatically start newly discovered bundles
`felix.fileinstall.filter`		A regular expression to be used to filter file names
`felix.fileinstall.tmpdir`		The name of the temporary directory to use with exploded or transformed bundles (defaults to the temporary dir for the JVM)
`felix.fileinstall.noInitialDelay`	false	Determines if File Install waits `felix.fileinstall.poll` milliseconds before doing an initial scan or not.
`felix.fileinstall.bundles.startTransient`	false	If `true`, File Install will start the bundles transiently.
`felix.fileinstall.bundles.startActivationPolicy`	true	Start bundles with their default activation policy or not
`felix.fileinstall.start.level`	0	If set to a value different from `0`, File Install will set the start level for deployed bundles to that value. If set to `0`, the default framework bundle start level will be used.
`felix.fileinstall.active.level`	0	FileInstall won't scan for files unless the active framework start level is greater than this value
`felix.fileinstall.enableConfigSave`	true	If `false`, File Install will not write back in the file configurations changes. Note that this setting is global and can not be modified per polled directory.
`felix.fileinstall.bundles.updateWithListeners`	false	If `true`, FileInstall will update managed bundles when a new version of an artifact handler is detected. This detection is performed based on the last modification date of the bundle registering the handler.

Once started, the values of these properties are printed to the console.

Configurations

Configuration files are plain property files (`java.util.Property`). The format is simple:

```
file ::= ( header | comment ) *
header ::= <header> ( ':' | '=' ) <value> ( '\<nl> <value> ) *
comment ::= '#' <any>
```

Notice that this model only supports string properties. For example:

```
# default port
ftp.port = 21
```

Configuration file names are related to the PID and factory PID. The structure of the file name is as follows:

```
filename ::= <pid> ( '-' <subname> )? '.cfg'
```

If the form is `<pid>.cfg`, the file contains the properties for a Managed Service. The `<pid>` is then the PID of the Managed Service. See the Configuration Admin service for details.

When a Managed Service Factory is used, the situation is different. The `<pid>` part then describes the PID of the Managed Service Factory. You can pick any `<subname>`, this bundle will then create an instance for the factory for each unique name. For example:

```
com.acme.xyz.cfg // configuration for Managed Service
// com.acme.xyz
com.acme.abc-default.cfg // Managed Service Factory,
// creates an instance for com.acme.abc
```

If a configuration managed by File Install is modified, File Install will write the configuration back to a property file to ensure peristence across restarts.

Property substitution in configuration files

It is possible to use system properties to specify the values of properties in configuration files. 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. Bundle context properties will take precedence over system properties if available.
Example:

```
ftp.port = ${system.ftp.port}
```

Watching multiple directories with File Install

Starting with version 3.1.0 it is possible to specify several directories to watch with the system property `felix.fileinstall.dir`; this property can either point to a single directory or a comma separated list of directories. In addition, Apache Felix File Install provides a ManagedServiceFactory to create multiple instances of File Install. Asuming you have a File Install bundle watching a `bundles` folder, creating a new instance is as simple as creating a new configuration file `org.apache.felix.fileinstall-<pid>.cfg` in that folder (substitute <pid> with a unique Id for the new service instance):

org.apache.felix.fileinstall-configDir.cfg

```
felix.fileinstall.poll=2000
felix.fileinstall.dir=/configDir
felix.fileinstall.debug=-1
felix.fileinstall.filter=.*\\.cfg
felix.fileinstall.bundles.new.start=false
```

Exploded bundles

Apache Felix File Install has the ability to watch for exploded bundles and automatically update such bundles if the content is changed in any way. If a watched directory contains a sub directory, its content will be jar'ed and deployed to the OSGi framework. Any change to a file in this directory or one of its subdirectories will result in the directory to be jar'ed again and the corresponding bundle to be updated.

Custom artifacts

Apache Felix File Install can support deployment of custom artifacts. By default, configurations and plain OSGi bundles are supported, but other kind of artifacts can be deployed through custom artifact handlers.

To add support for a custom artifact, a service implementing one of `org.apache.felix.fileinstall.ArtifactListener`'s sub interfaces must be registered in the OSGi registry.