Installer

Introduction

The security library supports a web installer in the web-installer module which can be used for writing your own. It take sets of files that live on the web and will write them to locations locally, creating directories as needed. This article is intended for developers who are using this. It is not intended to be user facing (though you can certainly feel free to boilerplate off sections as needed).

Basic use

The class is edu.uiuc.ncsa.security.installer.WebInstaller and it resides in the web-installer module of the NCSA security library. You can specify this as a maven dependency as

```
<dependency>
    <groupId>edu.uiuc.ncsa.security</groupId>
    <artifactId>web-installer</artifactId>
    <version>5.6</version>
</dependency>
```

(or whatever the correct version is). You can extend this class if you need to add customizations such as your own command line arguments, but as much as possible, this is configuration driven.

A typical example is to install the application with one version, then use the installer to update your local installation with subsequent versions.

Most of this article is just discussing the configuration files.

Operation of the installer

The current syntax for the installer is

```
java -jar installer.jar OPERATION [FLAG]* [SWITCH ARG]*
OPERATION : update | remove | notes | help | install | list | versions
   SWITCH : -dir | -version | -log
     FLAG : -v | -noPace | --help
```

The operation is required. Only known operations are allowed and unknown options result in an error message. Starting with no arguments prints the general help.

Operations

Name	Description
help	Print the general help for the installer. It includes a summary of command line options.
install	Install a given version to a given location. This requires the **`-dir`** switch and you

	may specify a version with the **-version** switch.
list	List all of the files for a given version. This implies using the **-version** switch if you want anything other than the latest. It also prints permissions and such.
notes	The release notes (if any) for a given version.
remove	Remove the distro. This implies the **-dir** switch. ***Nota bene:*** This will remove the directory and its contents completely!
update	Update an existing install with a given version. Note that
versions	List versions available for the installer.

Switches

Switches are of the form -name and take an argument.

Switch	Argument	Description
-dir	file path	The target directory for the operations install, update and remove.

Configurations

Main setup (required)

There is one basic configuration for the installer and that must reside in your local resources folder as `setup.yaml`.

Structure

```
app_name: STRING
versions: VERSION
(cleanup_on_fail: true | false)?
(fail_on_error: true | false)?
type: setup
help: HELP

VERSION:
         file : FILE
         name : STRING
  description : STRING
HELP :
       args : FILE
    success : FILE
  installer : FILE
   examples : FILE
      notes : FILE

FILE - Path relative to the resource directory.
STRING - a string
TRUE - logical
```

Simple attributes

Name	Req?	Default	Description
app_name	Y	“NCSA Sec-lib”	Name of the application being installed to display to the user.
fail_on_error	N	false	If there is an error, abort the install (true) or just log it (false).
cleanup_on_fail	N	false	If there is an error **and** fail_on_error is true, then remove the entire installation as it is.
type	Y	setup	Constant.

Versions

As you use the installer, the user may be allowed to install different versions. This block will allow for listing those and selecting them. The file listed here is a configuration file for all the files in your distribution. The value of this entry is a map whose values are

Name	Req?	Default	Description
file	Y	-	The path relative to the resource directory for this
name	N	(file)	Human readable name for this
description	N	(file)	A human readable description

Help

There are two ways to setup help. The first (described later) is to extend the class and override the help methods. The second is to set some files that contain it which are displayed at specific points by the system. This allows the system to display all the standard help it comes with plus it lets you augement or customize it.

Name	Req?	Default	Description
args	N	-	Prints help about custom arguments
examples	N	-	Additional examples, printed after the main help.
installer	N	-	Help about the application. This would replace the default help if present. (default is show when the user supplies --help as the command line argument)
notes	N	-	Release notes. These are displayed to the user when they use the **`notes`** option.
success	N	-	A final success message, printed after a successful installation. Typically this has additional step to take (e.g., telling the user to add a specific path to the PATH variable for their environment.)

Overrides

You may also extend the WebInstaller class and override the help directly. The advantage is that you can access information about the installer (such as configuration options passed in). The main methods are

`protected void showHelp()` - the main help for the installer. If not overridden displays installer and examples. Otherwise, shows examples after the main help.
`protected void printMoreArgHelp()` - printed after the basic arguments (such as -dir) if you have added your own to your class.
`protected void printMoreExamplesHelp()` - prints more examples after the installer main help.

Sample setup.yaml

```
---
- app_name: OA4MP
 help: {app: /siwtchHelp.txt, success: /success.txt}
 versions:
 - {file: /v5_5.yaml, name: v5.5, description: Version 5.5}
 - {file: v5.5, name: latest, description: latest release}
 cleanup_on_fail: false
 fail_on_error: false
 type: setup
```

File sets

Each version of the installer you specify has a set of files associated with it. These have a source (on the web) and a file name there. They are mapped to a local directory with a local file name. You may specify operations on the files, which are

- **execute** - the executable bit on the file is set true
- **update** - If this file should be overwritten on updates. Typically you specify files that are preprocessed and have their state set on installation, then in subsequent updates are not touched.
- **preprocess** - if templates should be applied to the file (must be a text file or very strange things may happen).

Structure

```
sourceURL : STRING
  type: file_set
  directories:
  -files :
   (- STRING | FILE_ENTRY)+
  target_dir:

FILE_ENTRY:FILE
preprocess: true | false
update: true | false
source: STRING
```

```
target: STRING
execute: true | false


FILE - the path relative to the target directory where this set of files will be
installed.
```

Simple Attributes

Name	Req?	Default	Description
sourceURL	Y	-	The URL on the web that is the source for every file in the file set.
type	Y	file_set	This is fixed

The Directories element

This contains lists of sets of files, each with a target directory. The files may be either a string that is simply the file name and no processing is done, or a FILE_ENTRY. The elements of the file entry are

Name	Req?	Default	Description
execute	N	false	If the file should be set executable after copying.
preprocess	N	false	If templates (See below) should be applied.
source	Y	file_set	The only requirement. The name of the source.
target	N	(source)	If the target file has a different name, specify it
update	N	true	

Example

This contains the source sets for two web locations, the OA4MP source tree for pdfs and the GutHub release page.

```
- sourceURL: https://github.com/ncsa/oa4mp/releases/download/v5.5/
  directories:
  - files:
    - oa4mp-derby.sql
    - oa4mp-mysql.sql
    - oa4mp-mariadb.sql
    - oa4mp-pg.sql
    - oa4mp-mariadb.sql
    - derby-migrate.sql
    - jwt-scripts.tar
    - {preprocess: true, update: false, source: cfg.xml}
    - oidc-cm-scripts.tar
    - {update: false, source: oa4mp-message.template}
    - {update: false, source: oa4mp-subject.template}
    target_dir: /etc
  - files:
```

```
    - {preprocess: true, source: cli, exec: true}
    - {preprocess: true, source: jwt, exec: true}
    - {preprocess: true, source: clc, exec: true}
    - {preprocess: true, source: migrate, exec: true}
    - qdl-installer.jar
    target_dir: /bin
  - files: [fs-migrate.jar, oauth2.war, jwt.jar, clc.jar, cli.jar]
    target_dir: /lib
  - {target_dir: /var/storage/server}
  - {target_dir: /log}
  - files: [version.txt]
    target_dir: /
  type: file_set
```

Templates

When the user starts the installation, they specify a target directory on the local machine with the **`-dir`** switch. In files that are preprocessed, this may be referenced as ${ROOT} and that means that any reference to it is replaced.

Example

If the user invokes the installer as

```
java -jar installer install -dir /opt/qdl -log /tmp/qdl.log
```

then let us say that there is this entry for the file

```
{preprocess: true, source: qdl-run, exec: true}
```

The result is that the file qdl-run is downloaded and set to be executable. When preprocessing, let us say the file contained

```
# The script to invoke the QDL interpreter.
QDL_HOME=${ROOT}
JAR_LIB="$QDL_HOME/lib/cp"
CFG_FILE="$QDL_HOME/etc/cfg-min.xml"
CFG_NAME="run-it"
QDL_JAR="$QDL_HOME/lib/qdl.jar"
java -cp "$JAR_LIB/*" -jar $QDL_JAR  -cfg  $CFG_FILE -name $CFG_NAME -home_dir $QDL_HOME -run
"$@"
```

then after preprocessing, it would read

```
QDL_HOME=/opt/qdl
JAR_LIB="$QDL_HOME/lib/cp"
CFG_FILE="$QDL_HOME/etc/cfg-min.xml"
CFG_NAME="run-it"
QDL_JAR="$QDL_HOME/lib/qdl.jar"
java -cp "$JAR_LIB/*" -jar $QDL_JAR  -cfg  $CFG_FILE -name $CFG_NAME -home_dir $QDL_HOME -run
"$@"
```

If you want to set your own templates, then you may override the method

```
protected Map<String,String> setupTemplates() throws IOException
```

Example of setting up templates

```
@Override
protected Map<String,String> setupTemplates() throws IOException {
    templates = super.setupTemplates();
    // get the root directory, slap the right separator on it and add it.
    String root = getArgMap().getRootDir().getAbsolutePath() + File.separator);
    templates.put("${QDL_LIB}", root + "lib";
    return templates;
}
```

Now there are two templates, ${ROOT} and ${QDL_LIB}