CFW Documentation

Contents

[1 File Structure 2](#_Toc5288505)

[2 API 3](#_Toc5288506)

[3 Logging 3](#_Toc5288507)

[3.1 Example Code 3](#_Toc5288508)

[3.2 Sample Configuration 3](#_Toc5288509)

[3.3 Known Issues 4](#_Toc5288510)

[4 Resources 4](#_Toc5288511)

[4.1 JAR Resources 4](#_Toc5288512)

[5 Maintenance 4](#_Toc5288513)

[5.1 Font Awesome 4](#_Toc5288514)

# File Structure

* **/config**
  + **logging.properties:** Configuration file for java.util.logging.
  + **cfw.properties:** Configuration file which has to pe passed to CFWSetup.*initialize*("./config/cfw.properties");
  + **credentials.csv:** file containing username/password credentials when using authentication\_method=CSV
  + **keystore.jks:** Default keystore for testing purposes(Password: 123456)
* **/datastore:** Default folder for the database files.
* **/log:** Folder containing the log files.
* **/resources**
  + **/css**
    - **Custom.css:** Your custom css file which will be loaded by TemplateHTMLDefault.
  + **/js**
  + **/html**

# Logging

By default the logging uses a asynchronous logger, which writes incoming logs all 50 milliseconds.

## Example Code

**public** **static** Logger *logger* = CFWLog.*getLogger*(YourClass.**class**.getName());

CFWLog log = **new** CFWLog(*logger*).method("initialize").start();

log.end();

Explanations:

* Initalize a logger with CFWLog.getLogger().
* Create a new CFWLog instance passing the logger
* Use CFWLog.start() and CFWLog.stop() method to create duration logs.

## Sample Configuration

handlers = com.pengtoolbox.cfw.logging.AsyncLogHandler, java.util.logging.FileHandler, java.util.logging.ConsoleHandler

config =

com.pengtoolbox.pageanalyzer.level = INFO

com.pengtoolbox.cfw.level = INFO

com.pengtoolbox.cfw.logging.AsyncLogHandler.level = FINE

com.pengtoolbox.cfw.logging.AsyncLogHandler.limit = 10000000

com.pengtoolbox.cfw.logging.AsyncLogHandler.append = false

com.pengtoolbox.cfw.logging.AsyncLogHandler.count = 1

com.pengtoolbox.cfw.logging.AsyncLogHandler.pattern = ./log/applog\_%u\_%g.log

#java.util.logging.FileHandler.level = FINE

#java.util.logging.FileHandler.filter =

#java.util.logging.FileHandler.formatter = com.pengtoolbox.cfw.logging.CFWLogFormatterJSON

#java.util.logging.FileHandler.encoding =

#java.util.logging.FileHandler.limit = 10000000

#java.util.logging.FileHandler.count =

#java.util.logging.FileHandler.append = false

#java.util.logging.FileHandler.count = 2

#java.util.logging.FileHandler.pattern = ./log/trace\_%u.%g.log

java.util.logging.ConsoleHandler.level = FINE

java.util.logging.ConsoleHandler.filter =

java.util.logging.ConsoleHandler.formatter = com.pengtoolbox.cfw.logging.CFWLogFormatterJSON

java.util.logging.ConsoleHandler.encoding =

## Known Issues

The Logging will not be formatted correctly, if one of the following is the case:

* The config file ./config/logging.properties is not defined
* The package names have changed
* The class name of CFWLogFormatterJSON was modified
* Any java.util.logging class was used before the logging.properties was loaded.

# Resources

## JAR Resources

You can fetch resources from JAR files using the servlet “jarresource”.

Parameters

* **pkg:** The package where the resource is stored.
* **File:** The file to be fetched

Example:

*./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.eot*

# Application

## HTTP to HTTPs Redirect

Add a HTTPSRedirectHandler as the first handler in your handler collection.

handlerCollection.setHandlers(**new** Handler[] {**new** HTTPSRedirectHandler(), apiContext, rewriteHandler, pageanalyzerContext, CFWSetup.*createResourceHandler*(), CFWSetup.*createCFWHandler*(), **new** DefaultHandler() });

### Known Issues

* When both HTTP & HTTPS is enabled, and redirect\_http\_to\_https is disabled, and user switches from HTTPS to HTTP, he cannot login anymore without restarting the browser.

# Maintenance

## Font Awesome

When upgrading font awesome:

* Update the CSS and font files in the package ”com.pengtoolbox.cfw.resources”
* Update font-awesome.css, @font-face.src definitions have to load files using the jarresource servlet.

@font-face {

font-family: *'Font Awesome 5 Free'*;

font-style: *normal*;

font-weight: *900*;

font-display: *auto*;

src: *url("./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.eot")*;

src: *url("./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.eot?#iefix")* *format("embedded-opentype"),* *url("./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.woff2")* *format("woff2"),* *url("./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.woff")* *format("woff"),* *url("./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.ttf")* *format("truetype"),* *url("./jarresource?pkg=com.pengtoolbox.cfw.resources.fonts&file=fa-solid-900.svg#fontawesome")* *format("svg")*;

}