# Atbash Configuration server

Rudy De Busscher

Version 0.9.1, ??/??/2018

# Table of Contents

User manual for Atbash configuration server.

# Release Notes

## 0.9.1

1. Support for Payara micro
2. update to latest Atbash config version

## 0.9.0

1. First release, just serves config values as Base64 encoded JSON.

# Introduction

Instead of having the MicroProfile configuration values within each application, they are now supplied by a server and centrally maintained.

Within the configuration of the server, we can define which client applications are known(supported) and the *root* directory where the configuration files are stored.

# Configuration

Define within a *config-server.properties* or *config-server.yaml* file the following configuration information.

- rootDirectory : Root directory containing the configuration files for the application. See further for a directory structure.
- applications : List of known applications.

example (as yaml)

```
rootDirectory : /Users/rubus/atbash/config-server/server/src/demo
applications : ["app1", "app2"]
```

Possible directory structure

```
src/demo
 ├──── /app1
 │   ├──── app1.properties
 │   └──── app1-test.properties
 └──── /app2
    └──── app2.properties
```

A subdirectory for each application so that the configuration values are nicely separated for each application. All the features of Atbash config (like stage, yaml support, date support, etc) are available.

# Creating the server

The Configuration server is built using the MicroProfile config 1.2 API. There are 3 possibilities to run your own configuration server

## Java EE 7 based

You can add an MicroProfile 1.2 config implementation, like Geronimo config, and deploy it to any Java EE 7 server running on Java 8 as a WAR file.

The WAR file can be created by the following command

```
mvn clean package -Pee7-server
```

## WildFly Swarm runnable jar

WildFly Swarm 2018.1.0 and later implement the MicroProfile 1.2 config implementation. An executable jar based on WildFly Swarm is thus also capable of running the Configuration server.

Building

```
mvn clean package -Pwildfly-swarm
```

Running

```
java -Dswarm.http.port=8181 -jar config-server-swarm.jar
```

## OpenLiberty runnable jar

Open Liberty 18.0.0.1 and later implement the MicroProfile 1.2 config implementation. An executable jar based on Open Liberty is thus also capable of running the Configuration server.

Building

```
mvn clean package -Pliberty
```

Running

```
java -jar config-server-swarm.jar
```

The port can be changed by the *httpPort* within the Maven Profile.

## Payara micro runnable jar

Payara micro 5.182 and later implement the MicroProfile 1.2 config implementation. An executable jar based on Payara micro is thus also capable of running the Configuration server.

Building

```
mvn clean package -Ppayara-micro
```

Running

```
java -jar config-server-microbundle.jar
```

# Client connector

This project also has a connector (ConfigSource) which connects with the Config server and integrates the call to the server within the Config.

It can be used from within Java EE, Java SE and MicroProfile.

The maven artefact of the client configuration is

```
    <dependency>
        <groupId>be.atbash.config</groupId>
        <artifactId>atbash-config-client</artifactId>
        <version>${atbash-config.version}</version>
    </dependency>
```

More info can be found at https://github.com/atbashEE/atbash-config-server/blob/master/client/src/main/doc/manual.adoc