Balena.IO: Drones Tracker

Prerequisites

- (**required**) Docker daemon started
- (*optional*) Maven installed – if you choose to build Docker image from source

Docker image

Either the provided Docker image can be loaded, or new one built from source.

Building Docker image from source is fully automated, but requires Maven installed and internet connection to download any required artifacts into local Maven repository.

a) Load the Docker image provided

   cat release/balena-dronesim-app.docker-image.tar | docker load

b) Build Docker image from source

   mvn -P docker install

Running

```
docker run -p 8080:8080 --name balena-drones balena-dronesim-app
```

Once launched, the single page application can be accessed at: http://localhost:8080/drones/index.html

Configuration

Application comes pre-configured – no initial configuration is required for application to run.

However, if you wish to tweak any of the settings, drone's configuration file can be found in `dronesim-app` Maven module, in `src/main/docker` sub-directory, and it's called `balena-drones.properties`. It contains configuration settings for four drones, each having different IDs, start and end latitudes and longitudes, as well as altitudes, and speeds. Configuration for new drones can be easily added following the same syntax.

Technology stack

a) Back-end

- Java 8
- OSGi Release 7 Core and Enterprise specifications implementations, including
  - OSGi Configurator
  - OSGi Metatype Service
  - OSGi Configuration Admin Service
  - OSGi DTO
  - OSGi Declarative Services
  - OSGi Converter
  - OSGi HTTP Whiteboard
  - OSGi JAX-RS Whiteboard
  - OSGi Push Stream
  - OSGi Promises
- Equinox OSGi container
- Apache Felix Http Jetty
- "Java Geocalc" – lightweight open source library for Earth coordinates calculations

b) Front-end

- HTML5
- CSS3
- JavaScript
- jQuery
- jQuery SSE

c) Build

- "Kubernetes Maven Plugin" – for automating Docker image build

More information

- ”OSGi Enterprise R7 Push Stream Specification” https://osgi.org/specification/osgi.enterprise/7.0.0/util.pushstream.html
- ”OSGi Enterprise R7 Promises Specification” https://osgi.org/specification/osgi.enterprise/7.0.0/util.promise.html
- “OSGi Compendium R7 Declarative Services Specification” https://osgi.org/specification/osgi.cmpn/7.0.0/service.component.html
- ”OSGi Enterprise R7 HTTP Whiteboard Specification” https://osgi.org/specification/osgi.enterprise/7.0.0/service.http.whiteboard.html
- ”OSGi Enterprise R7 JAX-RS Whiteboard Specification” https://osgi.org/specification/osgi.enterprise/7.0.0/service.jaxrs.html
- ”OSGi Compendium R7 Converter Specification” https://osgi.org/specification/osgi.cmpn/7.0.0/util.converter.html
- “OSGi Compendium R7 Configurator Specification“ https://osgi.org/specification/osgi.cmpn/7.0.0/service.configurator.html
- “OSGi Compendium R7 Configuration Admin Service Specification“ https://osgi.org/specification/osgi.cmpn/7.0.0/service.cm.html
- “OSGi Compendium R7 Metatype Service Specification“ https://osgi.org/specification/osgi.cmpn/7.0.0/service.metatype.html
- “OSGi enRoute [Maven] Archetypes“ https://enroute.osgi.org/about/112-enRoute-Archetypes.html
- ”Java Geocalc” https://github.com/grumlimited/geocalc
- ”jQuery SSE - jQuery Plugin for Server-Sent Events (SSE) EventSource Polyfill” https://github.com/byjg/jquery-sse
- “Kubernetes Maven Plugin“ https://github.com/deanjameseverett/k8-maven-plugin

Also, articles I published some time ago on my blog, regarding a different application which utilizes similar technologies, go into more detail:

- Using Camel and RabbitMQ in an OSGi R7 application, including custom message types
- Implementing asynchronous processing with OSGi R7 Promises
- Implementing efficient monitoring of long running operations with OSGi R7 Push Stream and Server Sent Events

- OSGi R7 HTTP Whiteboard, JAX-RS Whiteboard and Converter services applied
- Painless Monolith breakup or how automation and efficient design enables smooth transitions
- Automating Kubernetes deployments

More concise versions of these articles I also published on DZone and JAXenter:

- DZone: Data Streaming in OSGi R7 applications With OSGi R7 Push Stream and Server Sent Events
- JAXenter: Automated build and deployment of Docker containerized OSGi applications on Kubernetes