Runtime

Table of Contents

A Jakarta EE 10 Core profile runtime.

General

The product has several ways of operation

- standalone; Start up the runtime with one or more applications.
- domain; Start up the runtime and remotely administer the environment.
- clustered; Remote administration of several instances all running the same applications(s)
- embedded; Start the runtime as part of your application or test.

Applications are internally identified by the contextroot they are available (must be unique on the runtime) and when performing remote administration, applications are indicated by their deployment name.

Modules

Core module

Reacts on : Configuration changes

Exposes some crucial data and services

- RunData: All essential data of the current run of the Runtime
- WatcherService : A service to perform some logging (with JFR events if activated) and storing some POJO that might be exposed through JMX (if activated)

Config module

Dependent on : Core Module

Responsible for reading the configuration file, create them if needed, including the profile.

It exposes the following objects

- RuntimeConfiguration : The entire configuration of the current runtime instance
- PersistedDeployments : The Deployment information on the archives that are stored during the previous run.

Logging Module

Dependent on : Config Module

Based on the configuration information, it starts the logging. It also ends the 'temporary logging'.

It does not expose any Objects.

Jetty module

Dependent on : Config Module

It starts up the Jetty server based on the configuration information. When there is no `health` module active, it makes sure there is a basic handler for */health* so that system that inquiry if the runtime is healthy, behave properly.

It exposes

- HandlerCollection : So that other modules can add their own support for specifications or endpoints.

The module is also responsible for deploying applications that only use HTML or Servlet resources.

Jersey module

Dependent on : Jetty Module

The module is responsible for deploying applications that use JAX-RS resources.

Configuration

Logging

The runtime controls the entire logging subsystem.

The configuration is performed through a properties file that configures the Java Util Logging system. The file is expected at `<instance>/logging.properties` and is created from a default when not found.

Within the Runtime itself, the logging is performed through SLF4J API. A bridge to Java Util Logging is included. The `be.atbash.runtime.logging.handler.LogFileHandler` handles the logging to file.

By default, no messages are written to the console and if needed, the `--logToConsole` is required to view logging entries also on the console.

Early logging

Since the logging module requires the Configuration module to be started (so that configuration is known) the logging cannot be configured immediately after the Java program starts. The initial logging is captured by an `EarlyLogHandler` that stores the LogRecords within memory until the logging module is ready.

The Early Logging is configured through this call:

```
LoggingManager.getInstance().initializeEarlyLogging(logToConsole);
```

Performed steps are

- Define RuntimeLogManager as Log manager.
- Set logToConsole value as System property (`LoggingUtil.SYSTEM_PROPERTY_LOGGING_CONSOLE = "runtime.logging.console"`)
- Remove all handlers from th Root Logger.
- Add the Early Logging Handler.

Command line option

Runtime

java -jar atbash-runtime.jar <options>

-r|--root: Location of 'domain' directory, by default current directory.

-n|--configname: Configuration name within the domain, by default `default`.

-v|--verbose: Are all log messages shown on the console.

--watcher: Defines the type of the diagnostics, valid values are
`MINIMAL`: (default) only the events from the Core module are sent to JFR.
`OFF`: JFR and JMX are disabled
`JFR`: Events are send to JFR system and Recording is started which is dump to file at JVM exit.
`JMX`: Some data is available within the JMX system.
`ALL`: Combination of `JFR` and `JMX`

zero, one or more WAR files can be added to the command line that needs to be deployd. Also the applications that are already 'deployed' within the configuration are started.

CLI

java -jar atbash-runtime.jar <command> <options>

create-config

Create a configuration within the domain.

xx To be documented xx

Options for all remote commands

-h | --host (default - localhost): Host name or IP address of the machine running the Atbash runtime in Domain mode.

-p | --port (default - 8080): Port number assigned the process running the Atbash runtime in Domain mode

-f | --format (default - TEXT): Format output of the Remote CLI commands. Support values are TEXT and JSON.

status (remote)

Returns the status and information of the runtime with as output.

version - version number of the runtime. modules - the active modules of the runtime. uptime - The time the runtime is already running.

deploy (remote)

Deploys the application on the runtime, and optionally define the context root.

```
deploy <file>
```

```
deploy --contextroot <root> <file>
```

```
deploy --contextroot <root1,root2> <file1> <file2>
```

list-applications (remote)

List all applications running on the runtime.

undeploy (remote)

Undeploy an application

```
undeploy <name>
```

Integrations

Java Flight Recorder

Atbash Runtime has support for the Java Flight recorder functionality. When defining the option `--watcher JFR` or `--watcher ALL` for the instance/domain, a JFR Recording is started automatically and written out when the JVM exits. Several messages are also created as JFR events and allow the analysis of the JVM behaviour.

The JFR file is created in the current directory.

Remote CLI domain API

Overview of the endpoints when the domain mode is active and can be used to interact with the runtime remotely.

This API is also used by the `runtime-cli` tool.

Each API returns a JSON result.

GET - domain/status

Returns the status and information of the runtime.

POST - domain/deploy

Deploy application.

GET - domain/list-applications

List all applications running on the runtime.

POST - domain/undeploy

Undeploy an application

Messages

Core module

- DE-101(INFO): Start of the Deployment of the application.
- DE-102(INFO): End of the Deployment of the application.

Config module

- CONFIG-101(INFO): Start of the Module setup
- CONFIG-102(INFO): End of the Module setup

Logging Module

- LOG-101(INFO): Start of the Module setup
- LOG-102(INFO): End of the Module setup

Jetty module

- JETTY-101(INFO): Start of the Module setup
- JETTY-102(INFO): End of the Module setup

Jersey module

- JERSEY-101(INFO): Start of the Module setup
- JERSEY-102(INFO): End of the Module setup