Table of contents

Adds support for Dropwizard

Dropwizard Plugin - Reference Documentation

Authors: Burt Beckwith

Version: 0.1

Table of Contents

1 Introduction to the Dropwizard Plugin

The Dropwizard plugin integrates Dropwizard with Grails to make it easy to create REST services using th

Dropwizard is very opinionated, so several framework and library choices are fixed and cannot be changed

- you must use the Jetty server, Tomcat isn't an option
- Jackson
- Jersey
- Logback (this plugin depends on the Grails logback plugin to provide this)

In addition, the plugin enables the use of Freemarker and Mustache templates for response rendering, al JSON support, or other text-based syntaxes such as XML

Note that since Dropwizard requires Jetty and the current version of Jetty in the Grails Jetty plugin is to `BuildConfig.groovy`. Use the plugin's `run-dropwizard` script to start your application.

1.1 History

History

- March 5, 2013
  - initial 0.1 release

2 Getting Started

The first step is to add a dependency for the plugin in `BuildConfig.groovy`:

```
plugins {
   …
   compile ':dropwizard:0.1'
}
```

Next, create a YAML file that defines the Dropwizard configuration as described in the section on con checks, etc.) as described in the general usage section.

3 Configuration

Much of the Dropwizard configuration is done in a YAML file. This includes the server and admin HT done when using this plugin, but it's best to leverage what is already available in Grails where `DataSource.groovy`, but can also extend to using GORM or a NoSQL plugin for persistence, and support for Hibernate and Liquibase, but since these are trivial to use in Grails it makes more sense to use

Application classes

In a typical Dropwizard application, you define a service class that extends `com.yammer.dropwizard` to the application. It configures other helper classes including resources (REST endpoints), health checks, can customize most options.

There are three ways to register application classes. The easiest is to use the typical Grails convent `grails-app/dropwizard`. These can include resource classes, which must have a class name e "HealthCheck", and task classes ending in "DropwizardTask". These classes must all have default no-arg c

For more flexibility, you can register classes as Spring beans in `grails-app/conf/spring/res` property as described below to have the plugin find them and register them in Dropwizard for `Config.groovy` under the `grails.plugin.dropwizard.initializer` key that can do additi

YAML

See the Dropwizard documentation for the supported syntax for the configuration file, but you can u uncomment lines):

```
http:
   port: 8080

#   adminPort: 8081

#   2-1000000
#   maxThreads: 1024

#   1-1000000
#   minThreads: 8

#   rootPath: /*

#   one of blocking, legacy, legacy+ssl, nonblocking, nonblocking+ssl
#   connectorType: blocking

#   maxIdleTime: 200s

#   1-128
#   acceptorThreads: 1

#   -Thread.NORM_PRIORITY-Thread.NORM_PRIORITY)
#   acceptorThreadPriorityOffset: 0

#   min -1
#   acceptQueueSize: -1

#   min 1
#   maxBufferCount: 1024

#   requestBufferSize: 16KB
#   requestHeaderBufferSize: 6KB
#   responseBufferSize: 32KB
#   responseHeaderBufferSize: 6KB
```

```
#   reuseAddress: true
#   soLingerTime: null
#   lowResourcesConnectionThreshold: 0
#   lowResourcesMaxIdleTime: 0s
#   shutdownGracePeriod: 2s
#   useServerHeader: false
#   useDateHeader: true
#   useForwardedHeaders: true
#   useDirectBuffers: true
#   bindHost: null
#   adminUsername: null
#   adminPassword: null

#   requestLog:
#      console:
#         enabled: true
#         threshold: Level.ALL
#         timeZone: UTC
#         logFormat: null
#      file:
#         enabled: false
#         threshold: Level.ALL
#         currentLogFilename: null
#         archive: true
#         archivedLogFilenamePattern: null
#         # 1-50
#         archivedFileCount: 5
#         timeZone: UTC
#         logFormat: null
#
#      syslog
#         enabled: false
#         threshold: Level.ALL
#         host: localhost
#         # auth, authpriv, daemon, cron, ftp, lpr, kern, mail, news, syslog, use
local5, local6, local7
#         facility: local0
#         timeZone: UTC
#         logFormat: null

#      timeZone: UTC

#   gzip:
#      enabled: true
#      minimumEntitySize: 256B
#      bufferSize: 8KB
#      excludedUserAgents: ImmutableSet<String>
#      compressedMimeTypes: ImmutableSet<String>

#   ssl:
#      keyStore: /path/to/file
#      keyStorePassword: null
#      keyManagerPassword: null
#      keyStoreType: JKS
#      trustStore: /path/to/file
#      trustStorePassword
#      trustStoreType: "JKS"
#      needClientAuth: true/false
#      wantClientAuth: true/false
#      certAlias: null
#      allowRenegotiate: true/false
#      crlPath: /path/to/file
#      crldpEnabled: true/false
#      ocspEnabled: true/false
#      maxCertPathLength:
#      ocspResponderUrl:
#      jceProvider:
#      validatePeers
#      supportedProtocols:
#         - SSLv3
#         - TLSv1
#         - TLSv1.1
#         - TLSv1.2

#   contextParameters:
#      ImmutableMap<String, String>
```

Config.groovy

There are a few configuration options for the plugin that are defined in Config.groovy:

Property	Default	Meaning
grails.plugin.dropwizard. banner	none; look for a file named `banner.txt` in the classpath	the string t
grails.plugin.dropwizard. dropwizardContext	"dropwizard"	the string t
grails.plugin.dropwizard. autoRegisterResources	`false`	whether to those with
grails.plugin.dropwizard. autoRegisterHealthChecks	`false`	whether to those that
grails.plugin.dropwizard. autoRegisterManaged	`false`	whether to those `com.yam`
grails.plugin.dropwizard. autoRegisterLifeCycle	`false`	whether to those `org.ecl`
grails.plugin.dropwizard. autoRegisterAnnotatedProviders	`false`	whether to those with
grails.plugin.dropwizard. autoRegisterInjectableProviders	`false`	whether to beans `com.sun`
grails.plugin.dropwizard. autoRegisterTasks	`false`	whether to that extend
grails.plugin.dropwizard. yamlPath	`"classpath:dropwizard.yml"`	the locatio syntax
grails.plugin.dropwizard. assets	none	a List of I list can co resource p elements, there are t pattern, an value 'inde
grails.plugin.dropwizard. serviceClassName	`"grails.plugin.dropwizard.GrailsService"`	the name constructo pointing at
grails.plugin.dropwizard. initializer	none	optional cl `com.yam` `com.yam` `grails.` , and @org

4 General Usage

Usage

Start the application with

```
$ grails run-dropwizard
```

Use CTRL-C to stop.

URLs

As a hybrid application, the Grails URLs and Dropwizard URLs use different context paths. Fo
http://www.servername.com:8080/app/ and the Dropwizard endpoints under http://www.servername.c
sub-context, e.g. http://www.servername.com:8080/app/ and http://www.servername.com:8080/app/dw/
resource handlers, this isn't possible.

The context for your Grails controllers is determined the same way as when not using this plugin, i.e. it d
Dropwizard context with the `grails.plugin.dropwizard.dropwizardContext` attribute in `Co`

Admin URIs

Dropwizard has an admin servlet with some convenient URIs to monitor your application. By default this p

- http://server:8081/dropwizard/metrics
    - displays extensive runtime and usage information; append ?pretty=true to pretty-print the JSON
- http://server:8081/dropwizard/healthcheck
    - runs all health checks and displays their statuses
- http://server:8081/dropwizard/threads
    - displays a thread dump
- http://server:8081/dropwizard/ping
    - responds with "pong" as a simple test that something is there

Utility methods

Dropwizard displays all known endpoints at startup, but this information is also available at runtime. Depe

```
def dropwizardService
...
def endpoints = dropwizardService.findEndpoints()
```

This will be a `List` of `grails.plugin.dropwizard.util.EndpointData`

You can also retrieve information from Dropwizard via the `dropwizard` Spring bean. This is s
`com.yammer.dropwizard.config.Configuration`), `environment` (the `com.yammer.d`
`grails.plugin.dropwizard.GrailsService`)

Domain Object serialization

You can call GORM and use domain classes from your Dropwizard REST resources. Note however tha
automatic marshalling since there's no way to exclude non-serializable attributes. Use simple data classes
Groovy) instead.

Reloading

Reloading doesn't currently work. I'm looking into supporting runtime reloading in development mode, an