Dropwizard

Java “framework” for developing ops-friendly, high-performance, RESTful web services.

Production-ready, out of the box.

Main reference site: http://dropwizard.io/
TextAnalyzer sample app: https://github.com/cdavidson825/TextAnalyzer

What is Dropwizard?

- Developed/open-sourced at Yammer by CodaHale
- Pulls together stable, mature libraries from the Java ecosystem into a simple, light-weight web-enabled package that lets you focus on getting things done.
- Out-of-the-box support for sophisticated configuration, application metrics, logging, operational tools, allowing you to ship a production-quality web service quickly.
- Deployed as single “fat” JAR allowing for trivial/consistent deployment across all environments. Environment specific info is stored in config files provided at runtime

Why Dropwizard?

- Dropwizard is a collection of industry standard Java libraries for developing web services.
  - It is not an “opinionated" framework
  - If you’re developing web services, you would include most (if not all) of these libraries anyways.
- Promotes a consistent, trivial, scaleable container-less deployments using self-contained artifacts (“fat” jars).
- Dropwizard does not include “magic” methods nor “inject” functionality. Very simple to trace the flow.
- Built-in production-quality metrics, health checks, logging.

Libraries included w/ Dropwizard

- **Jetty** — Embedded web server
- **Jersey** — RESTful web services
- **Jackson** — JSON
- **Codahale Metrics** — production-quality metrics
- **Google Guava** — Java goodness
- **UI templating** — Mustache / Freemaker engines
- **Hibernate Validator** — Declariative user input validation
- **Apache HTTP/Jersey Client** — Lower-level web service interaction.
- **Additional** — Logback, JodaTime, JDBI, Liquidbase, etc…
- **@ 9 MB of JARs provided by default.**

Dropwizard Components

- **Configuration** (io.dropwizard.Configuation) — environment-specific parameters
- **Application** (io.dropwizard.Application) — application entry point.
- **Resource/Service** classes — Jersey web service endpoints
- **Representation** classes — serialized JSON response objects
- **HealthCheck** (com.codahale.metrics.health.HealthCheck) — runtime tests

Configuration class

```
public class TextAnalyzerConfig extends io.dropwizard.Configuration
```

- Used to handle reading the environment-specific parameters for your application - specified via a YAML/.yml file.
- Uses annotations:
  - Hibernate Validators -- @NotEmpty, @NotNull, @Valid, etc for startup/initialization
  - Jackson deserialzation -- @JsonProperty, @JsonIgnore, etc for mapping to/from YAML and your application via Jackson

Configuration class code example

```java
package cwd.ta.app;

import org.hibernate.validator.constraints.NotEmpty;
import com.fasterxml.jackson.annotation.JsonProperty;
import io.dropwizard.Configuration;

public class TextAnalyzerConfig extends Configuration
{

    @NotEmpty
    private String defaultText;

    @JsonProperty
    public String getDefaultText()
    {
        return defaultText;
    }

    @JsonProperty
    public void setDefaultText(String text)
    {
        this.defaultText = text;
    }

}
```

text-analyzer.yml

```yaml

##############################################################################
# application config:                                                        #
##############################################################################

defaultText: On Wednesday, we will be eating pumpkin and talking on the telephone.


##############################################################################
# System configs (overrides)                                                 #
# resource: https://dropwizard.github.io/dropwizard/manual/configuration.html #
##############################################################################

logging:
  level: INFO
  loggers:
    io.dropwizard: INFO
  appenders:
    - type: console
```

Application class

```
public class TextAnalyzerApp extends io.dropwizard.Application<MyConfiguration>
```

- Parameterized with the application's configuration
- Contains the static main method for the application’s entry point.
- Registration of Web service resources, asset bundles, health checks, etc.

Application class code example

```java
package cwd.ta.app;

import io.dropwizard.Application;
import io.dropwizard.assets.AssetsBundle;
import io.dropwizard.setup.Bootstrap;
import io.dropwizard.setup.Environment;
import io.dropwizard.views.ViewBundle;
import cwd.ta.app.health.TextAnalyzerHealthCheck;
import cwd.ta.app.resource.TextAnalyzerResource;

public class TextAnalyzerApp extends Application<TextAnalyzerConfig>
{
    public static void main(String[] args) throws Exception
    {
        new TextAnalyzerApp().run(args);
    }

    @Override
    public void initialize(Bootstrap<TextAnalyzerConfig> bootstrap)
    {
        bootstrap.addBundle(new ViewBundle());
        bootstrap.addBundle(new AssetsBundle());
    }

    @Override
    public void run(TextAnalyzerConfig configuration, Environment environment)
            throws Exception
    {
        final TextAnalyzerResource resource = new TextAnalyzerResource(configuration);
        final TextAnalyzerHealthCheck healthCheck = new TextAnalyzerHealthCheck(configuration);
        environment.healthChecks().register("AppHealthCheck", healthCheck);
        environment.jersey().register(resource);
    }
}
```

Resource class (Jersey web service endpoints)

```
@Path("/text-analyzer")
@Produces(MediaType.APPLICATION_JSON)
public class TextAnalyzerResource
```

- Each Resource class is associated with a URI template and has methods to handle the various GET/POST/PUT/DELETE requests.
- Resources should be stateless/immutable.
- Uses javax.ws.rs annotations:
  - Application-level -- @Path("/text-analyzer") and @Produces(javax.ws.rs.core.MediaType.APPLICATION_JSON)
  - Method invocation-level -- @GET / @POST / @PUT / @DELETE, @Timed
  - Method parameter-level -- @QueryParam("text")

Resource class code example *

* ERROR HANDLING REMOVED TO FIT ON SLIDE

```
27
28  @Path("/text-analyzer")
29  @Produces(MediaType.APPLICATION_JSON)
30  public class TextAnalyzerResource
31  {
32      private final TextAnalyzerConfig config;
33      private final static Logger logger = LoggerFactory.getLogger(TextAnalyzerResource.class);
34
35      public TextAnalyzerResource(TextAnalyzerConfig config)
36      {
37          this.config = config;
38      }
39
40      @POST
41      @Timed
42      public List<Analysis> analyzeViaPost(@FormParam("text") String text)
43      {
44          logger.info("Inside analyzeViaPost");
45          return (analyze(text));
46      }
47
48      @POST
49      @Path("/html")
50      @Produces(MediaType.TEXT_HTML)
51      public AnalysisView analyzeViaPostWithView(@FormParam("text") String text)
52      {
53          logger.info("Inside analyzeViaPostWithView");
54          List<Analysis> analysisList = analyze(text);
55          return (new AnalysisView("analysis.mustache", analysisList));
56      }
57
58      private List<Analysis> analyze(String text)
59      {
60          List<Analysis> analysisList = new ArrayList<Analysis>();
61          analysisList.add(new IdentityAnalyzer().analyze(text));
62          analysisList.add(new SummaryAnalyzer().analyze(text));
63          analysisList.add(new DollarWordAnalyzer().analyze(text));
64
65          return (analysisList);
66      }
```

Response (Representation) classes

- These are the Response objects from your Resources (web-service end points)
- Simple immutable JavaBean objects used by Jackson to serialize into JSON as a response to client.
- Can be “raw” JSON or transformed by templating engines (e.g. Mustache or Freemarker) to HTML.

Response class code example

```java
package cwd.ta.app.analyzer;

import java.util.Map;

public class Analysis
{
    final private String analyzerName;
    final private Map<String, String> analysisMap;

    public Analysis(String analyzerName, Map<String, String> analysisMap)
    {
        this.analyzerName = analyzerName;
        this.analysisMap = analysisMap;
    }

    public String getAnalyzerName()
    {
        return analyzerName;
    }

    public Map<String, String> getAnalysisMap()
    {
        return analysisMap;
    }

    public String getAnalysisFor(String key)
    {
        return analysisMap.get(key);
    }
}
```

Putting it all together: TextAnalyzer

https://github.com/cdavidson825/TextAnalyzer

TextAnalyzer

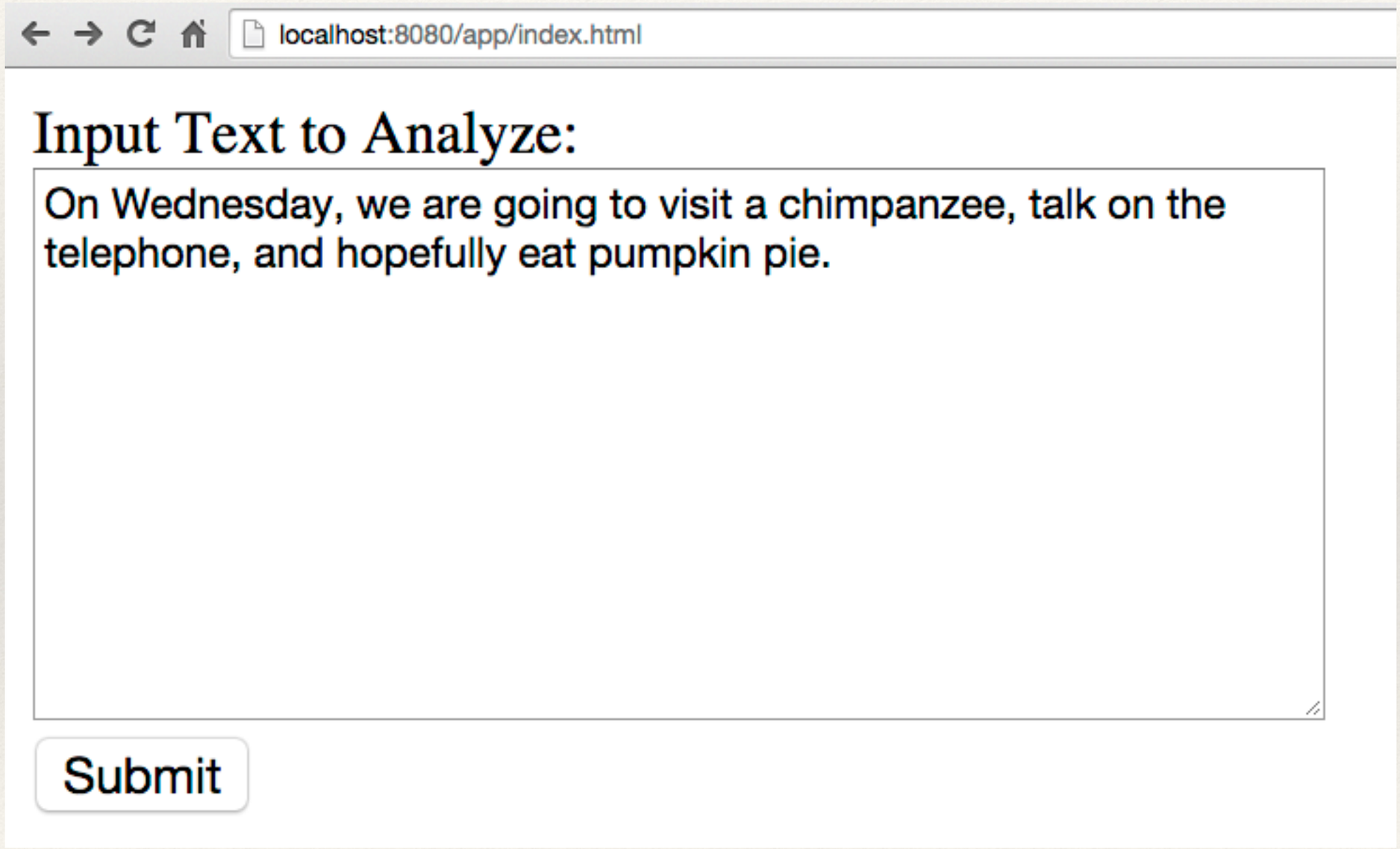
- Web app that performs analysis of input text using a series of VERY simplistic/notional "Analyzers".
  - IdentityAnalyzer: returns input unchanged
  - SummaryAnalyzer: counts characters and words
  - DollarWordAnalyzer: A=1,B=2,…Z=26, words that equal 100 ($1.00). Also returns the TOTAL_COST of the text input.

TextAnalyzer (cont.)

CLONE: git clone git@github.com:cdavidson825/TextAnalyzer.git
cd TextAnalyzer
BUILD: mvn
RUN: java -jar target/cwd-textanalyzer-1.0-SNAPSHOT.jar server text-analyzer.yml
— or just run ./startServer.sh

```
 ____ ______  ______     _    _   _    _    _  __   __________ ____
|_   _| ___\ \/ /_   _|   / \  | \ | |  / \  | | \ \ / /__  / ___|  _ \
  | | |  _|  \  /  | |    / _ \ |  \| | / _ \ | |  \ V /  / /|  _| | |_) |
  | | | |___ /  \  | |   / ___ \| |\  |/ ___ \| |___| |  / /_| |___|  _ <
  |_| |____/_/\_\ |_|  /_/   \_\_| \_/_/   \_\_____|_| /____|_____|_| \_\
```

TextAnalyzer Input



TextAnalyzer Output

localhost:8080/text-analyzer/html

TextAnalyzer Output:

IdentityAnalyzer : {ORIGINAL_TEXT=On Wednesday, we are going to visit a chimpanzee, talk on the telephone, and hopefully eat pumpkin pie.}

SummaryAnalyzer : {TOTAL_WORDS=18, TOTAL_CHARACTERS=86}

DollarWordAnalyzer : {TOTAL_COST=$9.49, DOLLAR_WORDS=[Wednesday, chimpanzee, telephone, pumpkin]}

—OR JSON —

localhost:8080/text-analyzer?text=On%20Wednesday,%20we%20are%20going%20to%20visit...


```
[{"analyzerName":"IdentityAnalyzer","analysisMap":{"ORIGINAL_TEXT":"On Wednesday,
we are going to visit a chimpanzee, talk on the telephone, and hopefully eat
pumpkin pie."}},{"analyzerName":"SummaryAnalyzer","analysisMap":
{"TOTAL_WORDS":"18","TOTAL_CHARACTERS":"86"}},
{"analyzerName":"DollarWordAnalyzer","analysisMap":
{"TOTAL_COST":"$9.49","DOLLAR_WORDS":"[Wednesday, chimpanzee, telephone,
pumpkin]"}}]
```

Dropwizard Administration

- Accessible via separate port (default 8081)
- Codahale Metrics (JSON) for view or integration with Ganglia / Graphite.
- (manual) HealthChecks
- ThreadDumps (i.e jstack)

Codahale Metrics:
http://localhost:8081/metrics?pretty=true


```
"cwd.ta.app.resource.TextAnalyzerResource.analyzeViaPost" : {
  "count" : 10,
  "max" : 0.0018424130000000002,
  "mean" : 9.862941E-4,
  "min" : 7.41297E-4,
  "p50" : 8.041165000000001E-4,
  "p75" : 0.00120337425,
  "p95" : 0.0018424130000000002,
  "p98" : 0.0018424130000000002,
  "p99" : 0.0018424130000000002,
  "p999" : 0.0018424130000000002,
  "stddev" : 3.536807846658874E-4,
  "m15_rate" : 0.002209922141215539,
  "m1_rate" : 0.030703655021877174,
  "m5_rate" : 0.006556779903598819 5,
  "mean_rate" : 0.017563063039269418,
  "duration_units" : "seconds",
  "rate_units" : "calls/second"
},
"cwd.ta.app.view.AnalysisView.rendering" : {
  "count" : 2,
  "max" : 0.048121916,
  "mean" : 0.0303746485,
  "min" : 0.012627381,
  "p50" : 0.0303746485,
  "p75" : 0.048121916,
  "p95" : 0.048121916,
  "p98" : 0.048121916,
  "p99" : 0.048121916,
  "p999" : 0.048121916,
  "stddev" : 0.025098426393563255,
  "m15_rate" : 0.11608479584547371,
  "m1_rate" : 6.68337227293087E-5,
  "m5_rate" : 0.03922515958534239,
  "mean_rate" : 0.003992787384564553,
  "duration_units" : "seconds",
  "rate_units" : "calls/second"
},
```

```
"org.eclipse.jetty.server.HttpConnectionFactory.8080.connections" : {
  "count" : 7,
  "max" : 60.312134687000004,
  "mean" : 39.11900014071429,
  "min" : 9.890533834000001,
  "p50" : 60.020624440000006,
  "p75" : 60.216379717,
  "p95" : 60.312134687000004,
  "p98" : 60.312134687000004,
  "p99" : 60.312134687000004,
  "p999" : 60.312134687000004,
  "stddev" : 26.274882951278688,
  "m15_rate" : 0.004963899487284789,
  "m1_rate" : 4.19010693464698E-4,
  "m5_rate" : 0.006294428255261951,
  "mean_rate" : 0.012283638084497537,
  "duration_units" : "seconds",
  "rate_units" : "calls/second"
},
"org.eclipse.jetty.server.HttpConnectionFactory.8081.connections" : {
  "count" : 3,
  "max" : 87.287836008,
  "mean" : 37.653916697,
  "min" : 9.999281773,
  "p50" : 15.674632310000002,
  "p75" : 87.287836008,
  "p95" : 87.287836008,
  "p98" : 87.287836008,
  "p99" : 87.287836008,
  "p999" : 87.287836008,
  "stddev" : 43.07780008957747,
  "m15_rate" : 0.002965381740714462,
  "m1_rate" : 0.009920491383704705,
  "m5_rate" : 0.007067423113367419,
  "mean_rate" : 0.005264468900328868,
  "duration_units" : "seconds",
  "rate_units" : "calls/second"
```

HealthChecks: http://localhost:8081/healthcheck

```
$ curl http://localhost:8081/healthcheck
{
“ApplicationCheck”:{“healthy":true},
“ConfigurationCheck”:{"healthy":true},
“DatabaseCheck”:{"healthy":true},
“deadlocks":{"healthy":true}
}
```

TextAnalyzer Code tree (minus tests)

```
.
├── pom.xml
├── src
│   ├── main
│   │   ├── java
│   │   │   └── cwd
│   │   │       └── ta
│   │   │           └── app
│   │   │               ├── TextAnalyzerApp.java
│   │   │               ├── TextAnalyzerConfig.java
│   │   │               ├── analyzer
│   │   │               │   ├── Analysis.java
│   │   │               │   ├── DollarWordAnalyzer.java
│   │   │               │   ├── IAnalyzer.java
│   │   │               │   ├── IdentityAnalyzer.java
│   │   │               │   └── SummaryAnalyzer.java
│   │   │               ├── health
│   │   │               │   └── TextAnalyzerHealthCheck.java
│   │   │               ├── resource
│   │   │               │   └── TextAnalyzerResource.java
│   │   │               └── view
│   │   │                   └── AnalysisView.java
│   │   └── resources
│   │       ├── app
│   │       │   └── index.html
│   │       ├── banner.txt
│   │       └── cwd
│   │           └── ta
│   │               └── app
│   │                   └── view
│   │                       └── analysis.mustache
├── startServer.sh
└── text-analyzer.yml
```

App and Config Classes

Response Class

Analyzers

Resource Class

Input Form

Config file

Questions / Comments?

http://dropwizard.io/
https://github.com/cdavidson825/TextAnalyzer

Thanks