

# DROPWIZARD

Pragmatische Web Services mit Dropwizard





## Dropwizard is a Java framework for developing ops-friendly, high-performance, RESTful web services.

Dropwizard pulls together **stable, mature** libraries from the Java ecosystem into a **simple, light-weight** package that lets you focus on *getting things done*.

Dropwizard has *out-of-the-box* support for sophisticated **configuration, application metrics, logging, operational tools**, and much more, allowing you and your team to ship a *production-quality* web service in the shortest time possible.

[Getting Started »](#)[User Manual »](#)[Javadoc »](#)[About Dropwizard »](#)[Other Versions »](#)

---

*Dropwizard is a Java framework for  
developing ops-friendly, high-performance,  
RESTful web services.*

---

---

*Dropwizard pulls together **stable, mature** libraries from the Java ecosystem into a **simple, light-weight** package that lets you focus on getting things done.*

---

---

*Dropwizard has out-of-the-box support for sophisticated **configuration, application metrics, logging, operational tools**, and much more, allowing you and your team to ship a **production-quality** web service in the shortest time possible.*

---



## Metrics is a Java library which gives you unparalleled insight into what your code does in production.

**Metrics** provides a powerful toolkit of ways to measure the behavior of critical components **in your production environment**.

With modules for common libraries like **Jetty**, **Logback**, **Log4j**, **Apache HttpClient**, **Ehcache**, **JDBI**, **Jersey** and reporting backends like **Ganglia** and **Graphite**, Metrics provides you with full-stack visibility.

[Getting Started »](#)

[User Manual »](#)

[About Metrics »](#)

YourKit is kindly supporting the Metrics project with its full-featured Java Profiler. YourKit, LLC is the creator of innovative and intelligent tools for profiling Java and .NET applications. Take a look at YourKit's leading software products: [YourKit Java Profiler](#) and [YourKit .NET Profiler](#).

# EINE KURZE GESCHICHTE DES FRAMEWORKS

- Entwickelt von [Coda Hale](#) bei Yammer
- Häufig auftretende Muster der Backend-Anwendungen bei Yammer in ein Framework extrahiert
- Erstes Release: 2011-12-22 ([Dropwizard 0.1.0](#))
- Eigenständiges Projekt seit 2014 ([Dropwizard 0.7.0](#))
- Aktuelles Release: [Dropwizard 1.0.5](#) (Stand: 2017-01-11)

# WARUM DROPWIZARD?

- Dropwizard bringt viele Standardkomponenten mit
  - Bekannte und bewährte Bibliotheken
  - Alle Teile bestens integriert
- Gute Dokumentation
- Wiederverwendung von Java EE Wissen
- Operations friendly
- Kurze **Time-to-Market**



# LANGWEILIGE TECHNOLOGIEN (1)

- Jetty als Web-Server (HTTP, HTTPS, HTTP/2)
- Jersey (JAX-RS) als Web-Framework
- Jackson für (De-) Serialisierung
- Logback und SLF4J für Logging
- Freemarker oder Mustache für Templates
- JDBC oder Hibernate für Persistenz

# LANGWEILIGE TECHNOLOGIEN (2)

- Guava als Ergänzung zur Java Standardbibliothek
- Hibernate Validator zur Validierung von Daten
- Liquibase für Datenmigration
- Guter Support für Tests (vorzugsweise JUnit)
- ...und vieles mehr, siehe [Dropwizard User Manual](#)

# DROPWIZARD APPLICATION (1)

- Normale Java-Klasse mit `main` Methode
- Keine Code-Generierung
- Keine (bzw. wenig) "Annotations Magic"

# DROPWIZARD APPLICATION (2)

```
public class DemoApplication extends Application<DemoConfiguration> {  
    public static void main(String[] args) throws Exception {  
        new DemoApplication().run(args);  
    }  
  
    @Override  
    public void initialize(Bootstrap<DemoConfiguration> bootstrap) {  
    }  
  
    @Override  
    public void run(DemoConfiguration configuration,  
                   Environment environment) throws Exception {  
    }  
}
```

# DROPWIZARD APPLICATION (3)

```
public void initialize(Bootstrap<DemoConfiguration> bootstrap) {  
    bootstrap.addBundle(...);  
    bootstrap.addCommand(...);  
}
```

# DROPWIZARD APPLICATION (4)

```
public void run(DemoConfiguration configuration,  
                Environment environment) throws Exception {  
    environment.jersey().register(EchoResource.class);  
    environment.lifecycle().manage(...);  
    environment.healthChecks().register("demo-health", ...);  
    environment.admin().addTask(...);  
}
```

# DROPWIZARD APPLICATION (5)

```
public class DemoConfiguration extends Configuration {  
    @NotBlank  
    private String customSetting;  
  
    public String getCustomSetting() {  
        return customSetting;  
    }  
}
```

# DROPWIZARD APPLICATION (6)

```
@Path("/echo")
@Produces(MediaType.TEXT_PLAIN)
public class EchoResource {
    @GET
    @Timed
    public String echo(@QueryParam("message")
                      @NotBlank
                      String message) {
        return message;
    }
}
```



# TESTS

- JUnit rules für
  - (JAX-RS) Resource Tests
  - Integration Tests
  - (Jersey) Client Tests

# TESTS - RESOURCE TESTS

```
@ClassRule
public static final ResourceTestRule resources =
    ResourceTestRule.builder()
        .addResource(new PingResource())
        .build();

@Test
public void testPingPong() throws Exception {
    final Pong expectedPong = ImmutablePong.builder()
        .message("pong")
        .build();
    final Pong actualPong = resources.client().target("/ping")
        .request()
        .get(Pong.class);

    assertEquals(expectedPong, actualPong);
}
```

# TESTS - INTEGRATION TESTS

```
@ClassRule
public static final DropwizardAppRule<DemoConfiguration> RULE =
    new DropwizardAppRule<>(
        DemoApplication.class,
        ResourceHelpers.resourceFilePath("demo_test.yml"));

@Test
public void testListKittens() throws IOException {
    final Client client = ClientBuilder.newClient();
    final Collection<Kitten> kittens = client
        .target(String.format("http://localhost:%d/kittens/", RULE.getPort()))
        .request(MediaType.APPLICATION_JSON)
        .get(new GenericType<Collection<Kitten>>() {});

    assertEquals(1, kittens.size());
    assertEquals("Findus", Iterables.getOnlyElement(kittens).getName());
}
```

# TESTS - HAPPY BUILDS

## Test Summary

10

tests

0

failures

0

ignored

0.332s

duration

100%

successful

Packages

Classes

Package	Tests	Failures	Ignored	Duration	Success rate
<a href="#">com.example.demo</a>	1	0	0	0.015s	100%
<a href="#">com.example.demo.health</a>	3	0	0	0.001s	100%
<a href="#">com.example.demo.resource</a>	5	0	0	0.313s	100%
<a href="#">com.example.demo.tasks</a>	1	0	0	0.003s	100%

# OPERATIONS FRIENDLY

- Healthchecks
- Tasks (ad-hoc ausführbare Aktionen via HTTP)
- Commands (ad-hoc ausführbare Aktionen via CLI)
- Metrics (via [Dropwizard Metrics](#))
- Gut konfigurierbares Logging (Logback, SLF4J)
- Einfaches Deployment als JAR-Datei

# OPERATIONS FRIENDLY - HEALTHCHECKS (1)

```
protected Result check() throws Exception {  
    if (store.isRunning()) {  
        return Result.healthy("Store is OK.");  
    } else {  
        return Result.unhealthy("Store is offline.");  
    }  
}
```

# OPERATIONS FRIENDLY - HEALTHCHECK (2)

```
{  
  "deadlocks" : {  
    "healthy" : true  
  },  
  "demo-health" : {  
    "healthy" : true,  
    "message" : "Everything is fine. :)"  
  },  
  "store-health" : {  
    "healthy" : true,  
    "message" : "Store is OK."  
  }  
}
```

# OPERATIONS FRIENDLY - TASKS (1)

```
public void execute(ImmutableMultimap<String, String> parameters,  
                    PrintWriter output) throws Exception {  
    output.println(parameters);  
}
```



# OPERATIONS FRIENDLY - TASKS (2)

```
$ curl -X POST \  
  'http://localhost:8080/admin/tasks/echo?param=value1&param=value2'  
{param=[value1, value2]}
```

# OPERATIONS FRIENDLY - COMMANDS (1)

```
protected void run(Bootstrap<DemoConfiguration> bootstrap,  
                  Namespace namespace,  
                  DemoConfiguration configuration) throws Exception {  
    System.out.format(  
        Locale.ENGLISH,  
        "Hi! This demo is presented by %s in %d.\n",  
        configuration.getSpeakerName(),  
        configuration.getYear());  
}
```

# OPERATIONS FRIENDLY - COMMANDS (2)

```
$ java -jar build/libs/demo-1.0.0-SNAPSHOT-all.jar greet config.yml  
Hi! This demo is presented by Jochen Schalanda in 2017.
```

# OPERATIONS FRIENDLY - COMMANDS (3)

```
$ java -jar build/libs/demo-1.0.0-SNAPSHOT-all.jar check config.yml  
INFO [2017-01-11 15:02:26,269] io.dropwizard.cli.CheckCommand:  
Configuration is OK
```

```
$ java -jar build/libs/demo-1.0.0-SNAPSHOT-all.jar check config.yml  
config.yml has an error:  
  * Unrecognized field at: spiekerName  
    Did you mean?:  
      - speakerName
```

# OPERATIONS FRIENDLY - METRICS

```
{  
  "com.example.demo.resource.PingResource.pingPong" : {  
    "count" : 6,  
    "m15_rate" : 1.0,  
    "m1_rate" : 1.0,  
    "m5_rate" : 1.0,  
    "mean_rate" : 0.7386288754923656,  
    "units" : "events/second"  
  }  
}
```

Metrics Reporter für Graphite, InfluxDB, Prometheus uvm.

# DEMO

## Demo application on GitHub

- Microservice
- NoSQL
- DevOps-friendly
- **BINGO!**

# ERWEITERUNGEN

- <http://modules.dropwizard.io/>
- 5 offiziell, 74 von der Community (Stand: 2017-01-11)

# WER BENUTZT EIGENTLICH DROPWIZARD?

- Uber (Gurafu,  $\mu$ ETA)
- Yammer
- Palantir
- Airbnb (Airpal)
- Sky
- Gini (Fotoüberweisung)
- Längere Liste im Dropwizard Wiki



# DIE LIEBE VERWANDTSCHAFT

- Bootique
- Spring Boot

# KONTAKT



**FRAGEN?**



# WEITERFÜHRENDE QUELLEN

- Dropwizard
- RESTful Microservices mit Dropwizard (codecentric)
- Micro-Services in Java realisieren – Teil 1:  
Leichtgewichtige Web-Apps mit Dropwizard (InnoQ)
- Microservices - Are your Frameworks ready? (InnoQ)
- Dropwizard als REST-App-Server (Heise)
- REST-Services mit Dropwizard ruck-zuck erstellt,  
dokumentiert und getestet (BED-Con)

# WARUM EIGENTLICH DER KOMISCHE NAME?

