FOCUS ON YOUR FEATURES

DROPWIZARD TAKES CARE OF THE REST

Felix Braun @ JavaLand 2015

DROPWIZARD'S HIGHLIGHTS

- Develop & deploy a RESTful microservice in 5 minutes
- Application start-up time under 2 seconds
- 15 LOC for a simple service.

LIGHTNING FAST DEVELOPMENT AND DEPLOYMENT OF A PRODUCTION-READY SERVICE.

AGENDA

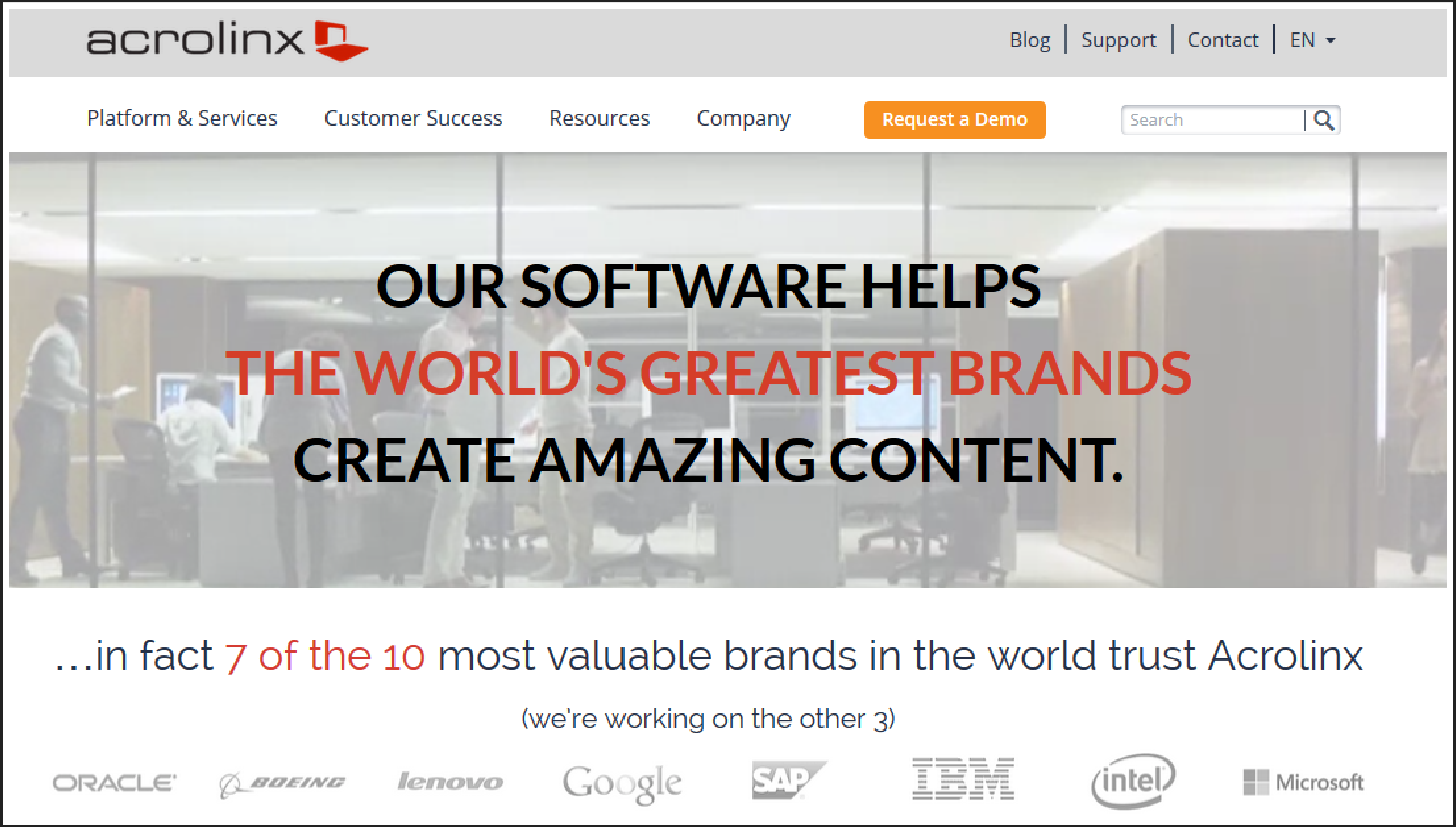
- Motivation for Microservices
- Dropwizard Introduction
- More Dropwizard Bundles
- Dropwizard @ Acrolinx
- Q & A

ABOUT ME

- Team Lead Server Development at Acrolinx
- 12 years experience as Java developer.
- Currently most interested in microservice architecture

Xing: Felix_Braun7 | Mail: an@felixbraun.de

ACROLINX

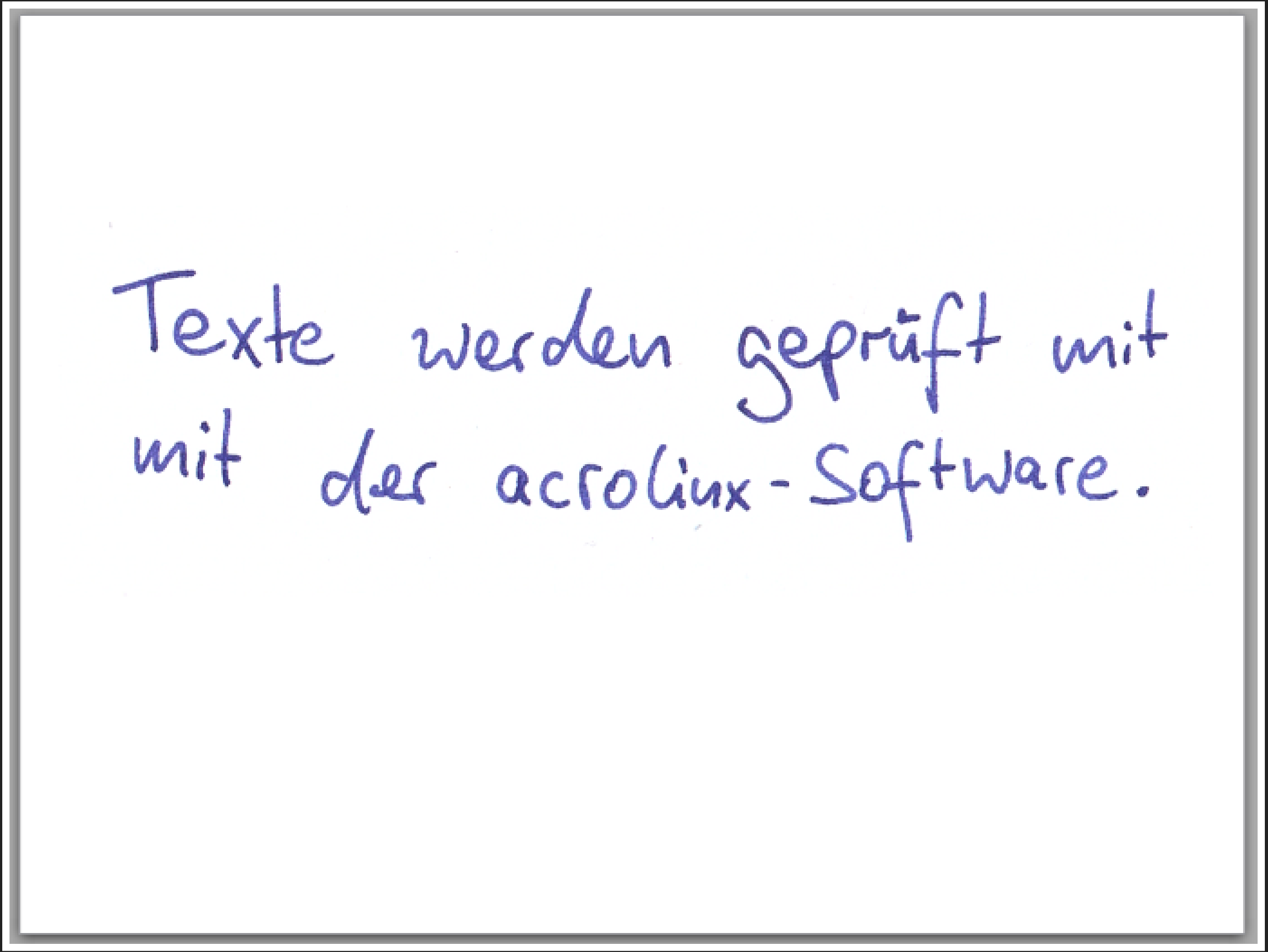


OUR SOFTWARE HELPS
THE WORLD'S GREATEST BRANDS
CREATE AMAZING CONTENT.

...in fact 7 of the 10 most valuable brands in the world trust Acrolinx

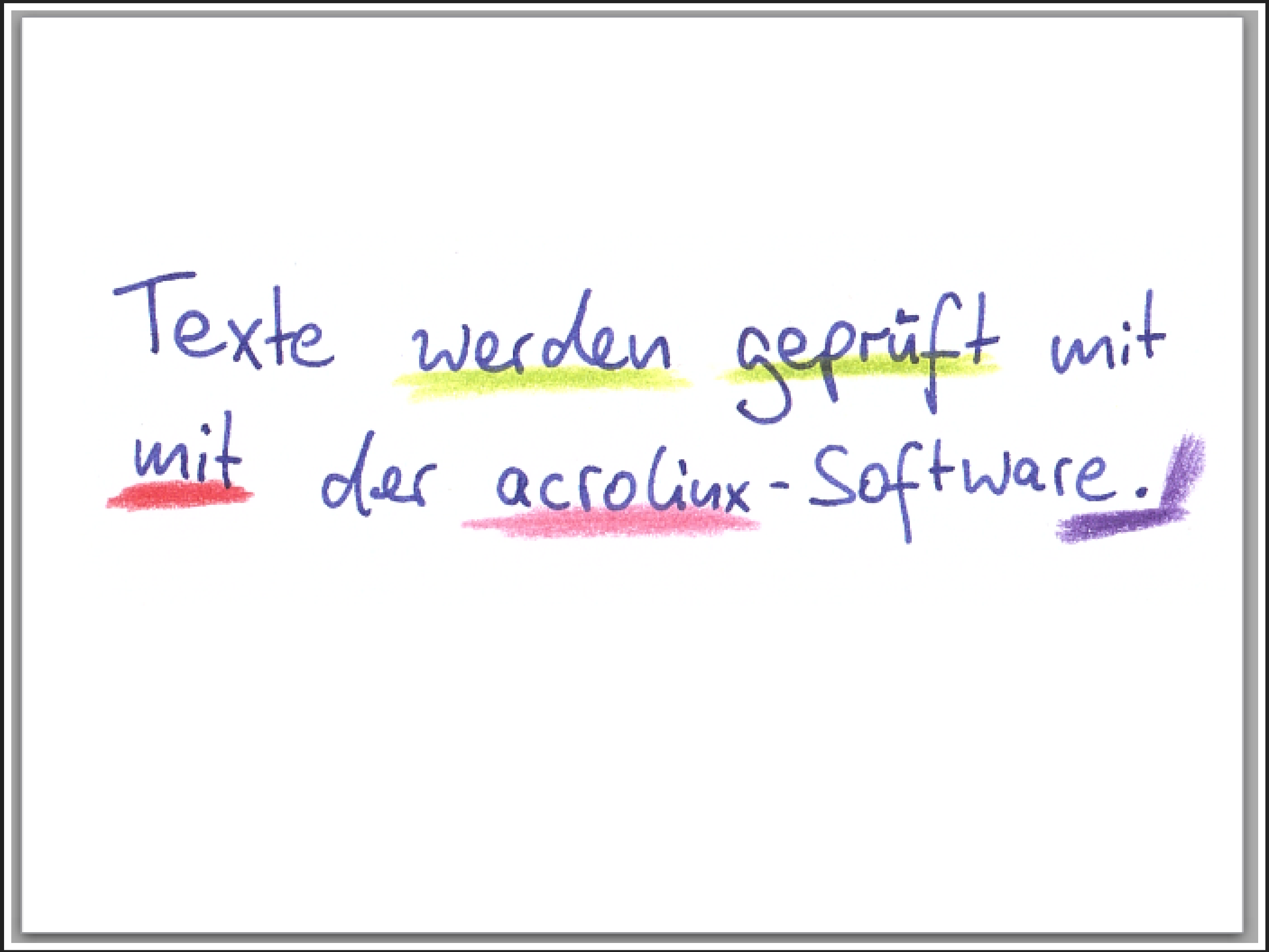
(we're working on the other 3)

ACROLINX



Texte werden geprüft mit
mit der acrolinx-Software.

ACROLINX



Texte werden geprüft mit
mit der acrolinx-Software.

DROPWIZARD MOTIVATION

"DEATH TO THE APPLICATION SERVER"

FIGHTING THE MONOLITH

yammer™

- Launched in 2008
- 2012 nearly 8 million users
- 2012 bought by microsoft

DROPWIZARD

- First release December 2011
- Distilled the patterns from Yammer's services
- Today it's used to develop and deploy a landscape of hundreds of microservices at Yammer

WHAT'S INSIDE?

- Jetty for HTTP
- Jersey for REST
- Jackson for JSON

- Supporting actors:
  - Metrics
  - Guava
  - Mockito
  - Joda-Time
  - JDBI
  - and much more...

THE SIMPLEST DROPWIZARD APPLICATION

1. pom.xml
2. MyApplication.java
3. MyConfiguration.java & config.yml
4. MyResource.java

THE MAVEN POM

```
<project>
    <groupId>com.acrolinx.demo</groupId>
    <artifactId>hipster-o-mat</artifactId>
    <version>0.1-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>io.dropwizard</groupId>
            <artifactId>dropwizard-core</artifactId>
            <version>0.8.0</version>
        </dependency>
    </dependencies>
    ...
</project>
```

THE APPLICATION

```java
public class HipsterApplication
extends Application<HipsterConfiguration> {

    public static void main(final String[] args) {
        new HipsterApplication().run(args);
    }

    @Override
    public void run(final HipsterConfiguration conf,
            final Environment env) throws Exception {

        env.jersey().register(new HipsterResource());
    }
}
```

THE RESOURCE

```
@Path("/hipsters")
public class HipsterResource {

    @GET
    @Path("ping")
    public Pong foobar() {
        return new Pong();
    }
}
```

THE CONFIGURATION

```
public class HipsterConfiguration extends Configuration {
        private String conferenceName;

        public String getConferenceName() {
                return conferenceName;
        }
}
```

```
conferenceName: JavaLand 2015

server:
  type: simple
  applicationContextPath: /
  adminContextPath: /admin
  connector:
    type: http
    port: 12345
```

BUILDING YOUR APPLICATION

```
> mvn package

[INFO] Building hipster-o-mat 0.1-SNAPSHOT
[INFO] Compiling 7 source files to C:\...\hipster-o-mat-jugbb\target\clas

----------------------
T E S T S
Running com.ax.demo.HipsterResourceTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] Building jar: C:\...\target\hipster-o-mat-0.1-SNAPSHOT.jar
[INFO]
[INFO] --- maven-shade-plugin:2.2:shade (default) @ hipster-o-mat ---
[INFO] Including io.dropwizard:dropwizard-core:jar:0.8.0 in the shaded ja
...
[INFO] Replacing original artifact with shaded artifact.
[INFO] --------------------------
BUILD SUCCESS
[INFO] Total time: 10.530s
```

RUNNING THE APPLICATION

```
java -jar hipster-o-mat-0.1-SNAPSHOT.jar server hipster.ym
```

```
Hello JavaLand 2015
INFO [15:52:13,998] io.d.s.ServerFactory: Starting HipsterApplication
INFO [15:52:14,058] org.e.j.SetUIDListener: Opened HipsterApplication@20
INFO [15:52:14,728] io.d.j.DropwizardResourceConfig: The following paths

GET     /hipsters/ping (com.ax.demo.resource.HipsterResource)

INFO [15:52:14,828] o.e.jetty.s.Server: Started @2134ms
```

```
>curl "http://localhost:12345/hipsters/ping"
{"msg":"Pong"}
```

WHAT HAPPENED SO FAR...

With 15 LOC we developed a RESTful Ping-Pong webservice.
Easy to build and easy to deploy.

What Dropwizard adds on-top:
All of your application's **metrics** as JSON.
Healthchecks show if our application is healthy.
We can have a look at the **thread** dump of our application.

HEALTH CHECK

```
public class HipsterServiceHealthCheck extends HealthCheck

    protected Result check() {
    if (store.isRunning()) {
      return Result.healthy("I'm fine. Store is running.")
    } else {
      return Result.unhealthy("Oho, no storage for hipster
    }
  }
}
```

```
public void run(final HipsterConfiguration conf,
  final Environment environment) throws Exception {

  environment.healthChecks().register("hipsterHealth",
  new HipsterServiceHealthCheck(store));
  ...
```

METRICS

```
@Timed
```

```
@GET
@Path("ping")
public Pong foobar() {
  return new Pong();
}
```

MORE FROM THE METRICS LIBRARY:

- Counter
- Gauges
- Meters
- Histograms

Let's look again:
Our Ping-Pong **metric** .
Our **HipsterHealthcheck**.

VIEWS

Fast HTML views using FreeMarker or Mustache.

```
bootstrap.addBundle(new ViewBundle<HipsterConfiguration>(){.
```

```
public class HipsterView extends View {

  public HipsterView(Hipster hipster) {
    super("hipster.mustache");
    this.hipster = hipster;
  }
  public Hipster getHipster() {
    return hipster;
  }}
```

```
@GET
@Path("{name}/view")
@Produces({ MediaType.TEXT_HTML, MediaType.APPLICATION_JSON
public HipsterView getHipsterView(@PathParam("name") String
   return new HipsterView(getHipster(name));
}
```

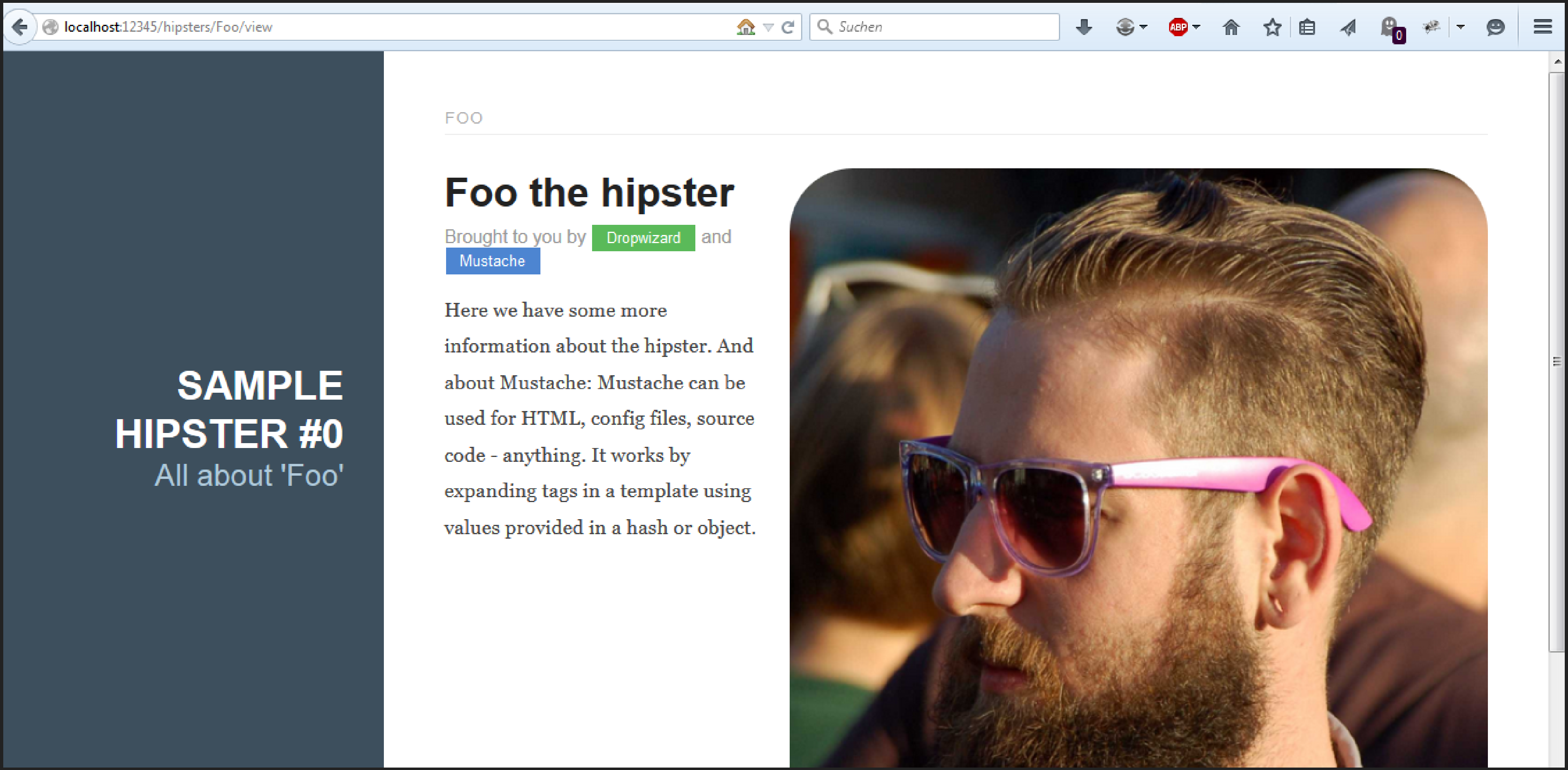
VIEWS

```
<body>
{{#hipster}}
<div id="layout" class="pure-g">
    <div class="sidebar pure-u-1 pure-u-med-1-4">
        <div class="header">
            <hgroup>
                <h1 class="brand-title">Sample Hipster #{{id}}</h1>
                <h2 class="brand-tagline">All about '{{name}}'</h2>
             </hgroup>
        </div>
    </div>
</div>
{{/hipster}}
...
```

VIEWS

One URL - Two Representations

http://localhost:12345/hipsters/Foo/view



SAMPLE HIPSTER #0

All about 'Foo'

FOO

Foo the hipster

Brought to you by Dropwizard and Mustache

Here we have some more information about the hipster. And about Mustache: Mustache can be used for HTML, config files, source code - anything. It works by expanding tags in a template using values provided in a hash or object.

VIEWS

One URL - Two Representations

http://localhost:12345/hipsters/Foo/view

```
curl -X GET -H "Accept: application/json" http://localhost
STATUS 200 OK
{"hipster":
  {"id":0,
  "name":"Foo",
  "jeans":"SKINNY",
  "hornRimmedGlasses":true,
  "imagePath":null}
}
```

TESTS

Support for unit and integration tests:

```
@ClassRule DropwizardAppRule<HipsterConfiguration> RULE
 = new DropwizardAppRule<HipsterConfiguration>(
 HipsterApplication.class, resourceFilePath("hipster.yml"));
```

```
@Test public void testHipsterGetCreateRoundtrip() {

Client client = ClientBuilder.newClient();
Response response = client.target(
  String.format("http://localhost:%d",RULE.getLocalPort())
  .path("hipsters").request(APPLICATION_JSON)
  .post(Entity.json(getHipster("foo")));

assertEquals(201, response.getStatus());

Hipster hipReceived = client.target(String.format(
  "http://localhost:%d/hipsters/foo", RULE.getLocalPort())
  .request(MediaType.APPLICATION_JSON).get(Hipster.class);
assertEquals(getHipster("foo"), hipReceived);
}
```

MANAGED

```
environment.lifecycle().manage(store);
```

```
public class HipsterStore implements Managed {

        @Override
        public void start() throws Exception {...}

        @Override
        public void stop() throws Exception {...}

}
```

VALIDATION

```
public class Hipster {
    @Min(value = 0, message = "Id must be positive")
    private int id;
    ...
}
```

```
@POST
public Response addHipster(@Valid final Hipster hipster){
...
```

```
Status 422 - {"errors":["id Id must be positive (was -2)"]
```

THAT'S ALL

... AT LEAST FOR DROPWIZARD-CORE.

PERFORMANCE

See Oli B. Fischer @ heise Developer
Sample Application one REST-Method with a counter.

Running with warm-up on a MacBook Pro 2.6 GHz i7 with OS X 10.9.4 and Oracle Java 1.7.0.45:

- Dropwizard 0.7.1 -> 55.000 Req/s
- Tomcat 7.0.55 -> 25.000 Req/s
- GlassFish 4.0 -> 19.000 Req/s.

PERFORMANCE

- 5% Metrics-Framework (only in benchmark situations)
- Complete Roundtrip (REST call, JSON De-/Serializing) between two machines in our office ~0.5ms

CONFIGURABLE ASSETS BUNDLE

```
public class SampleConfiguration
extends Configuration implements AssetsBundleConfiguration

  @Valid @NotNull @JsonProperty
  AssetsConfiguration assets = new AssetsConfiguration();

  public AssetsConfiguration getAssetsConfiguration()
   {return assets;}
}
```

```
public void initialize(Bootstrap<SampleConfiguration> bs)
  bs.addBundle(
    new ConfiguredAssetsBundle("/assets/", "/dashboard/"))
}
```

```
assets:
  overrides:
    /dashboard/assets: /some/absolute/path/with/assets/
    /dashboard/images: /some/different/absolute/path/image
  mimeTypes:
    woff: application/font-woff
```

DISCOVERY

`io.dropwizard.modules:dropwizard-discovery`

https://github.com/jplock/dropwizard-discovery

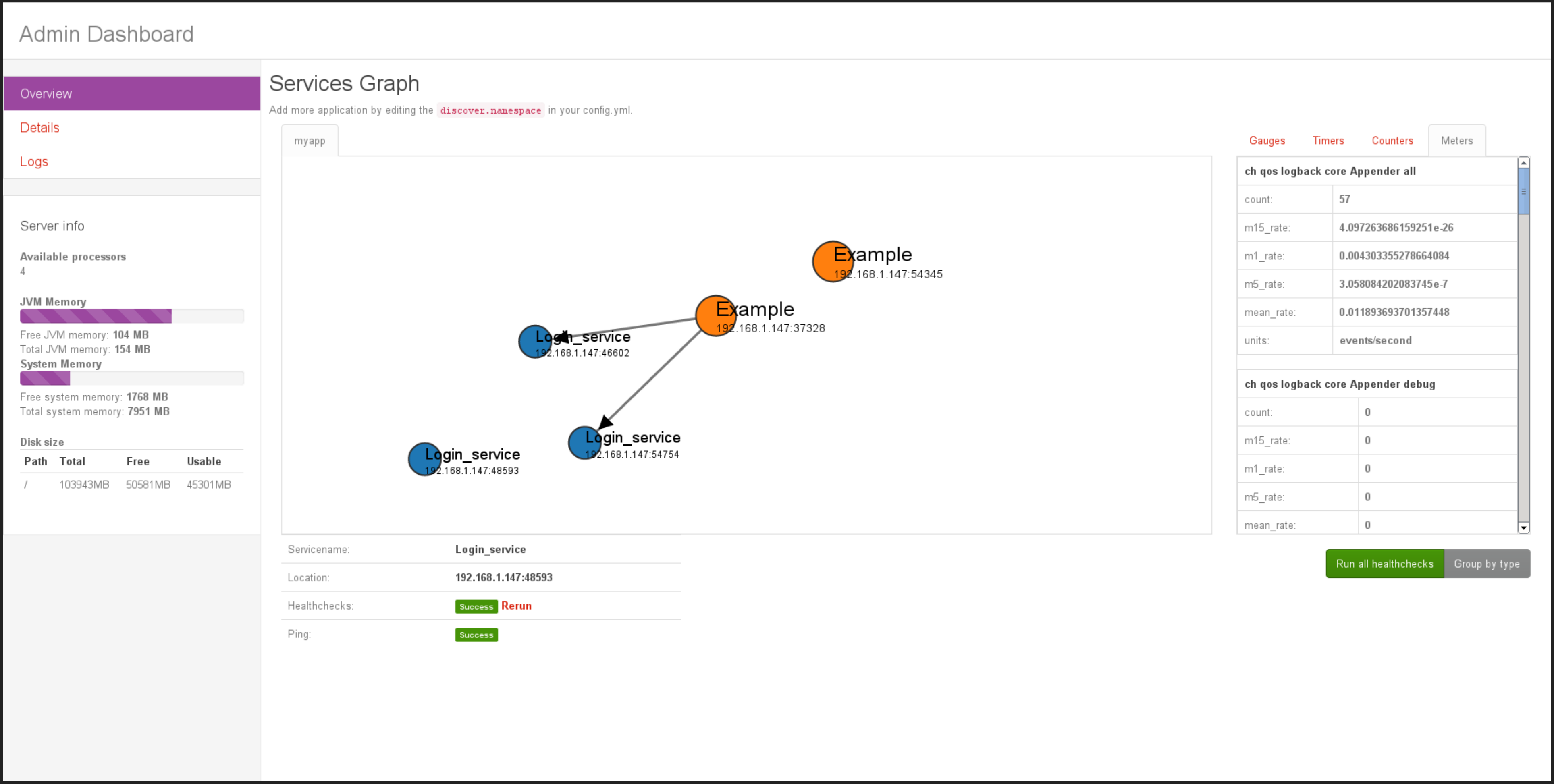
```
discovery:
 serviceName: hello-world
```

```
public void initialize(Bootstrap<HipsterConfiguration> bootstra
bootstrap.addBundle(discoveryBundle);
}
```

```
final DiscoveryClient client =
  discoveryBundle.newDiscoveryClient("other-service");
environment.lifecycle().manage(
  new DiscoveryClientManager(client));
```

ADMIN-DASHBOARD

https://github.com/abduegal/Microservice-Admin-Dashboard



Server info

Available processors
4

JVM Memory
Free JVM memory: **104 MB**
Total JVM memory: **154 MB**

System Memory
Free system memory: **1768 MB**
Total system memory: **7951 MB**

Disk size

Path	Total	Free	Usable
/	103943MB	50581MB	45301MB

Services Graph

Add more application by editing the `discover.namespace` in your config.yml.

Servicename:	**Login_service**
Location:	**192.168.1.147:48593**
Healthchecks:	Success **Rerun**
Ping:	Success

ch qos logback core Appender all	
count:	**57**
m15_rate:	**4.097263686159251e-26**
m1_rate:	**0.004303355278664084**
m5_rate:	**3.058084202083745e-7**
mean_rate:	**0.011893693701357448**
units:	**events/second**

ch qos logback core Appender debug	
count:	**0**
m15_rate:	**0**
m1_rate:	**0**
m5_rate:	**0**
mean_rate:	**0**

RESTFUL API DOCUMENTATION

A maven doclet for your dropwizard application:
https://github.com/teamcarma/swagger-jaxrs-doclet

https://github.com/swagger-api/swagger-ui

Example: Hipster Documentation

LESSONS LEARNED

NEXT STEPS

- Getting Started Guide on www.dropwizard.io
- This Hipster-Application Demo on Github
- Dropwizard als REST-App-Server von Oli B. Fischer auf heise Developer
- Dropwizard User Group