Dropwizard in 10 minutes

Kevin Scaldeferri
April 5, 2012

Why?

A Modest Proposal

- Maven/Nexus
- Scala
- Gerrit
- Releases RSS
- DropWizard
- Coverage
- Project Sites
- .NET

Dropwizard is:

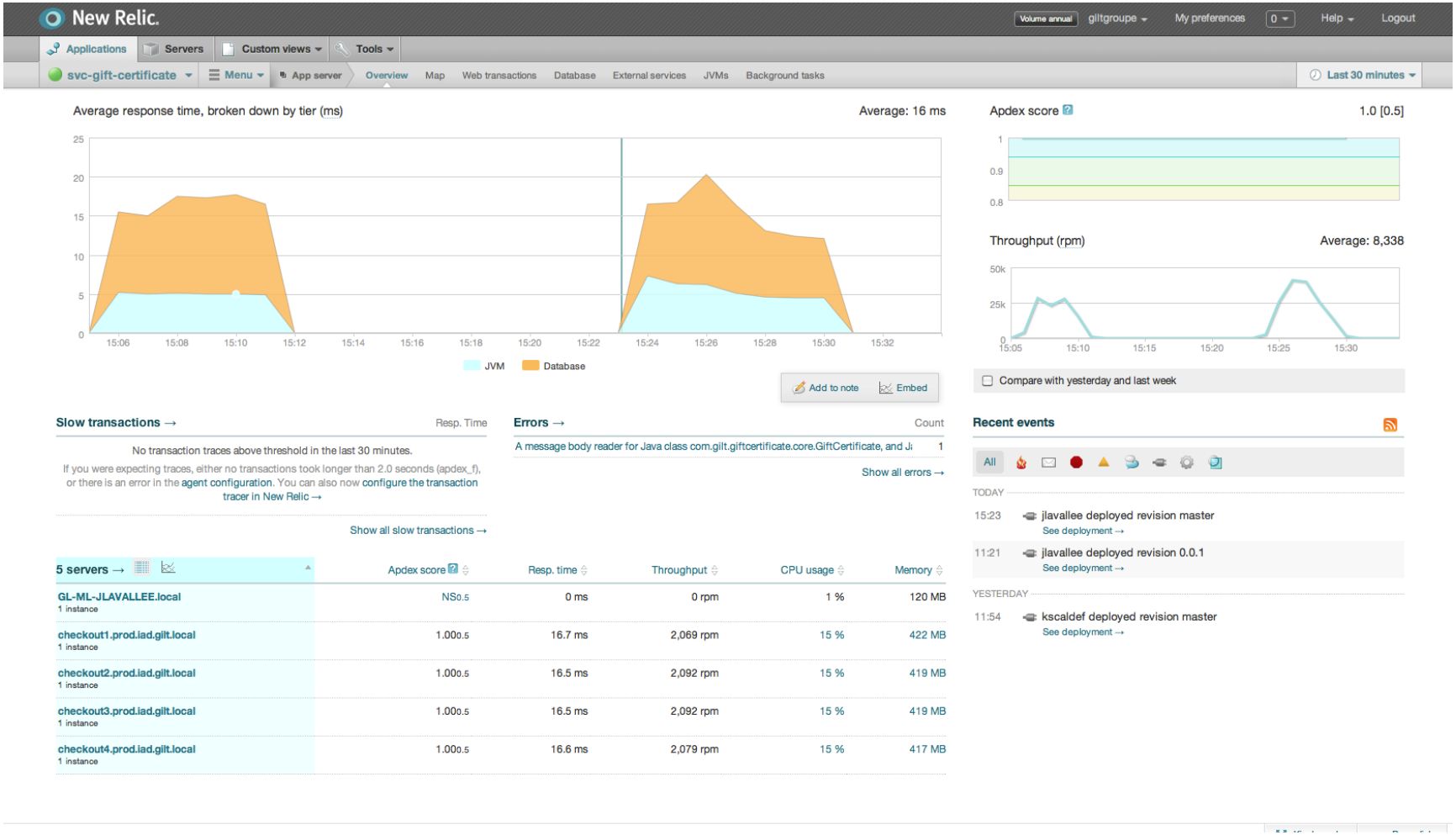
"a Java [and Scala] framework for developing ops-friendly, high-performance, RESTful web services."

http://dropwizard.codahale.com/

Dropwizard is: Java

- Jetty
- Jersey
- Jackson
- Metrics
- JDBI
- Log4j, Guava, Hibernate Validator

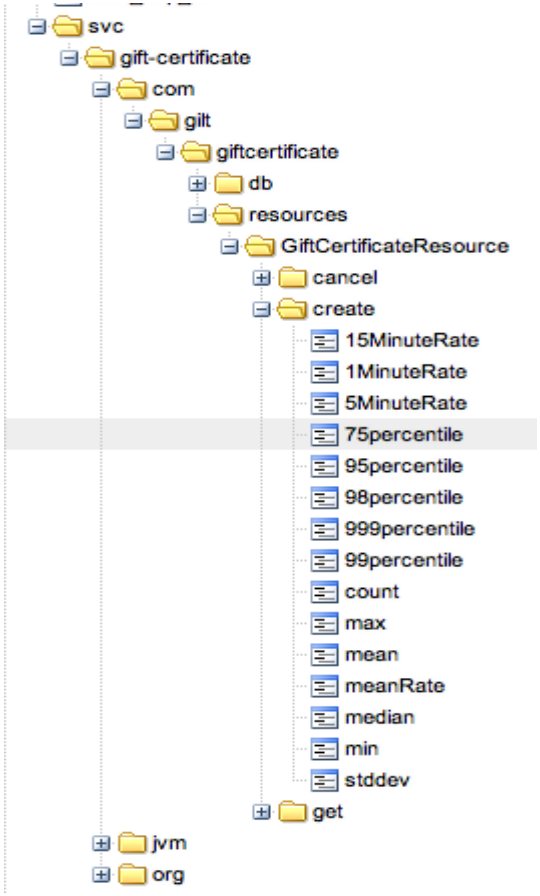
Dropwizard is: High-Perfomance



Dropwizard is: Ops-friendly

Admin port

Graphite Metrics



RESTful Web Services

... all code from here out ...

Representations

```
@JsonSnakeCase
case class GiftCertificate(
  guid: String,
  amount: JInteger,
  senderGuid: String,
  senderName: String,
  creditCardGuid: String,
  recipientName: String,
  recipientEmail: String,
  message: Option[String],
  channelId: JLong,
  storeId: JLong
)
```

(Type-Safe) Configuration

```
class GiftCertificateConfiguration extends Configuration {
  @Valid
  @JsonProperty
  val database = new DatabaseConfiguration()

  @Valid
  @JsonProperty
  val commons = new CommonsConfiguration()

  @Valid
  @JsonProperty
  val metrics: Option[MetricsConfiguration] = None

...
```

(Type-Safe) Configuration

```
class MetricsConfiguration {
  @NotEmpty
  @JsonProperty
  val graphiteHost: String = ""

  @Range(min=1024,max=65535)
  @JsonProperty
  val graphitePort: Int = 0
}
```

YAML Config

```
http:
  port: 7883
  adminPort: 7884

commons:
  user_service_uri: "http://localhost:6501/users/1.0"
  auth_service_uri: "http://localhost:6101/auth_service"
  principal_guid: "9472ae70-30c2-012c-8f71-0015177442e6"

metrics:
  graphiteHost: 184.73.182.206
  graphitePort: 2003

database:
  ...
```

Resources

```
@Path("/svc-gift-certificate/gift_certificate/{guid}")
@Produces(Array(MediaType.APPLICATION_JSON))
@Consumes(Array(MediaType.APPLICATION_JSON))
class GiftCertificateResource(...) extends Logging {
  @GET
  @Timed
  def get(@PathParam("guid") guid: String) = {
    Option(gcDao.findByGuid(guid)) match {
      case Some(gc) => gc
      case None => throw new WebApplicationException
(Status.NOT_FOUND)
    }
  }
  ...
```

JDBI SQLObjects

```
trait GiftCertificateDAO {

  @SqlQuery("""
select gc.guid, gc.amount, gc.sender_id,
       sender_name, cc_guid, recipient_name, recipient_email_address,
       message, channel_id, store_id
  from gift_certificates gc
  join gift_certificate_assignments gca on gca.gift_certificate_id = gc.id
  left join payments on payments.gift_certificate_id = gc.id
 where gc.guid = :guid
""")
  def findByGuid(@Bind("guid") guid: String): GiftCertificate

...
```

JDBI

```
  @SqlUpdate("""
insert into gift_certificates
  (sender_id, amount, status, guid,
   esch_sender_name, esch_sender_street_line1,
   esch_sender_street_line2, esch_sender_city,
   esch_sender_state, esch_sender_postal_code
  )
values
  (:sender_id, :gc.amount, 's', :gc.guid,
   :cc.firstName || ' ' || :cc.lastName, :cc.streetLine1,
   :cc.streetLine2, :cc.city,
   :cc.state, :cc.postalCode
  )
""")
  def insert(@Bind("sender_id") senderId: JLong, @BindBean("gc")
giftCertificate: GiftCertificate, @BindBean("cc") creditCard: CreditCard)
```

Tying it together

```
object GiftCertificateService
    extends ScalaService[GCConfiguration]("svc-gift-certificate") {

  def initialize(configuration: GCConfiguration, environment: Environment)
{
    val factory = new DatabaseFactory(environment)
    val db = factory.build(configuration.database, "postgresql")
    val users = UsersFactory.instance(configuration.commons)

    environment.addResource(new GiftCertificateResource(db, users))

    configuration.metrics match {
      case Some(metrics) => GraphiteReporter.enable(1, TimeUnit.MINUTES,
metrics.graphiteHost, metrics.graphitePort, "svc.gift-certificate")
      case None => ()
    }
  }
```

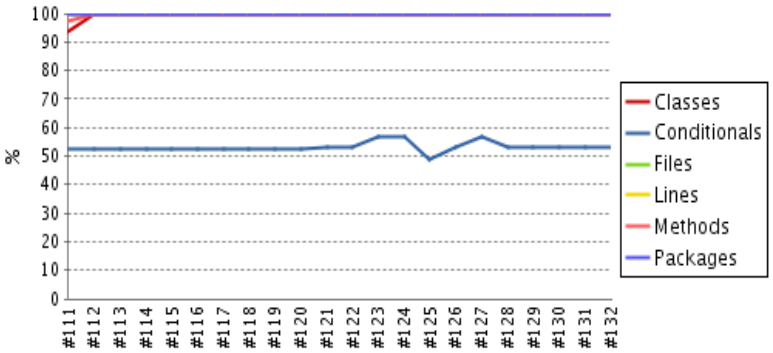
What to Like

Very little boilerplate
All the application, all the time
Fast & lightweight
Well-documented
Most things you want out of the box
Zero to service in production in < 2 weeks
Highly testable

Code Coverage

Cobertura Coverage Report

Trend



Project Coverage summary

Name	Classes	Conditionals	Files	Lines	Methods	Packages
Cobertura Coverage Report	100% 20/20	53% 65/123	100% 11/11	100% 153/153	100% 45/45	100% 5/5

Coverage Breakdown by Package

Name	Classes	Conditionals	Files	Lines	Methods
com.gilt.giftcertificate	100% 7/7	83% 5/6	100% 7/7	100% 45/45	100% 13/13
com.gilt.giftcertificate.config	100% 4/4	60% 6/10	100% 1/1	100% 11/11	100% 9/9
com.gilt.giftcertificate.core	100% 3/3	36% 27/75	100% 1/1	100% 14/14	100% 5/5
com.gilt.giftcertificate.db	100% 1/1	100% 2/2	100% 1/1	100% 10/10	100% 2/2
com.gilt.giftcertificate.resources	100% 5/5	83% 25/30	100% 1/1	100% 73/73	100% 16/16

Warts

Slightly annoying to integrate with NewRelic

Many libraries (*cough* Commons) don't expect strongly-typed configuration objects

Hibernate Validator doesn't play well with Scala

Would like:

```
case class GiftCertificate(
  @NonEmpty guid: String,
  @Range(min=1,max=2000) amount: JInteger,
  @NonEmpty senderGuid: String,
  @NonEmpty senderName: String,
  @NonEmpty creditCardGuid: String,
...
)

def create(@Valid giftCertificate: GiftCertificate) = {
  ...
}
```

Instead:

```
def isValid: Boolean =
  guid.nonEmpty &&
  amount > 0 &&
  amount <= 2000 &&
  senderGuid.nonEmpty &&
  senderName.nonEmpty &&
  creditCardGuid.nonEmpty &&
  ...
```

And manually validate on creation

```
def create(giftCertificate: GiftCertificate): Response = {
  if (!giftCertificate.isValid) return Response.status(422).build()
```

Thank You

Questions?