WEB SERVICES WITH GROOVY AND DROPWIZARD

Kyle Boon

KYLE BOON

Lead Developer @ Bloomhealth
kyle.f.boon@gmail.com
@kyleboon
http://www.about.me/kyleboon

WHAT IS DROPWIZARD

Dropwizard is a heavily opinionated framework for building web services on the JVM. It is mostly glue around mature java libraries like Jetty, Jersey, Jackson and Guava.

"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 HTTP+JSON web service in the shortest time possible. "

WHO CREATED IT?

@coda

As I've said before, the only reason Dropwizard exists at all is to provide opinions on what a service should be.

It uses fat JARs because I think they work better.

It embeds Jetty because I think that works better.

It uses Jackson because I think that works better.

It uses Jersey because I think that works better.

It has a single YAML configuration file because I think that works better.

It wraps Logback because I think that works better.

WHO USES IT

- Yammer
- Simple
- Bloomhealth

THE STACK AT BLOOMHEALTH

- Groovy for programming
- Grails for web applications
- Dropwizard for JSON web services
- Gradle for builds
- Swagger for Service Discovery
- Spock for testing
- Gatling for Performace/Load Testing
- Redis for Caching
- RabbitMQ for messaging

HOW DOES IT WORK?

Deployed as a fat jar and starts jetty. No need for a container.

```
public static void main(String[] args) throws Exception {
    new ContactsService().run(args)
}
```

Start the server from the command line by running:

```
java -jar contact_dropwizard/build/libs/contact_dropwizard-shadow-0.1.0
-SNAPSHOT.jar server start dev_config.yml
```

THE SERVICE

Services are a collection of bundles, commands, healthchecks, tasks and resources. The service class defines all of the abilities of your application.

```
@Override
public void initialize(Bootstrap bootstrap) {
    bootstrap.name = 'configuration_service'

    bootstrap.addBundle migrationsBundle
    bootstrap.addBundle hibernateBundle
    bootstrap.addBundle(new AssetsBundle('/swagger-ui-1.1.0/', '/swagger'))
    bootstrap.addCommand(new migrationsCommand())
}

@Override
    public void run(ContactsConfiguration configuration,
    Environment environment) throws ClassNotFoundException {

    ContactDAO contactDAO = new ContactDAO(hibernateBundle.sessionFactory)
    environment.addResource(new ContactResource(contactDAO))
}
```

THE RESOURCE

Resources model what is exposed via your RESTful API. Dropwizard uses Jersey for this so these classes are mostly jersey annotations.

```
@Path('/contacts')
@Produces(MediaType.APPLICATION_JSON)
class ContactResource {
    private final ContactDAO contactDAO

    public ContactResource(ContactDAO contactDAO) {
        this.contactDAO = contactDAO
    }


    @Timed(name = 'createContact')
    @POST
    @UnitOfWork
    public Contact createContact(@Valid Contact contact) {
        return contactDAO.saveOrUpdate(contact)
    }
}
```

THE REPRESENTATION

Your POGOs will be turned into JSON via Jackson. Hibernate Validator lets you specify validation rules.

```
@ToString
@EqualsAndHashCode
class Contact {
    @JsonProperty
    Long id

    @NotNull
    @NotEmpty
    @JsonProperty
    String firstName

    @NotNull
    @NotEmpty
    @JsonProperty
    String lastName

    @Transient
    @JsonIgnore
```

METRICS

Yammer Metrics is built in and provides metrics over an administration port (8081). Resources can be annotated with `@Timed` or `@Metered`, and `@ExceptionMetered`. Dropwizard augments Jersey to automatically record runtime information about your resource methods.

HEALTH CHECKS

Health Checks are a method to make sure the infrastructure your service depends on are all running. They are accessible on the administration port.

```
public class MySQLHealthCheck extends HealthCheck {
    private final SessionFactory sessionFactory

    public MySQLHealthCheck(SessionFactory sessionFactory) {
        super("MySQL")
        this.sessionFactory = sessionFactory
    }

    @Override
    protected com.yammer.metrics.core.HealthCheck.Result check() {
        if (!sessionFactory.closed) {
            return healthy()
        } else {
            return unhealthy('Session Factory is Closed!')
        }
    }
}
```

BUNDLES

Bundles are reusable blocks of behaviour designed to be reused across services. Assets, Hibernate and Liquibase are all implemented as Dropwizard Bundles.

COMMANDS

Commands add options to the command line interface of your service. For example the server starts based on the 'server' command. Migrations run based on the 'db migrate' command. You might add your own command for running functional tests or seeding the database.

TASKS

Tasks are run time actions available over the administration port. Dropwizard ships with a garbage collection task. You might want to right a task to clean a cache by key.

OTHER STUFF

- Configuration
- Logging
- Hibernate/JDBI
- Clients
- Authentication
- Views

LETS LOOK AT ’REAL’ CODE

REFERENCES

- Dropwizard User Guide
- Dropwizard User Group
- https://github.com/codahale/dropwizard
- Presentation about Dropwizard @ Yammer
- Presentation about Dropwizard @ Simple
- Coda Hale and Metrics
- Coda Hale and the Programming Ape

CODE WE'VE LOOKED AT

- Example Contact Application
- Swagger
- Enhanced Groovy Doc
- swagger-jaxrs-doclet
- dropwizard dashboard

THANKS

A special thanks to the people who actually figured all this stuff out. Including but not limited to:

- Chad Small
- Sairam Rekapalli
- Ryley Gahagan
- Charlie Knudsen
- John Engelman