
FREE eBook
LEARNING
dropwizard
Free unaffiliated eBook created from
Stack Overflow contributors.
#dropwizard

# Table of Contents

# About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: dropwizard

It is an unofficial and free dropwizard ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official dropwizard.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# Chapter 1: Getting started with dropwizard

## Remarks

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.

It uses:

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

## Examples

### Installation or Setup

Detailed instructions on getting dropwizard set up or installed.

Read Getting started with dropwizard online: https://riptutorial.com/dropwizard/topic/3884/getting-started-with-dropwizard

# Chapter 2: Creating Commands

## Examples

### Subcommands

This example shows how to define sub-commands for a given command, and how to easily associate a command handler. This example defines a `thing` command with `get` and `set` subcommands.

```
package things;

import io.dropwizard.cli.Command;
import io.dropwizard.setup.Bootstrap;
import net.sourceforge.argparse4j.inf.FeatureControl;
import net.sourceforge.argparse4j.inf.Namespace;
import net.sourceforge.argparse4j.inf.Subparser;

public class ThingCommand extends Command {

    private static final String THING_COMMAND_NAME = "thing";

    private static final String THING_SUBCOMMAND_GET = "get";
    private static final String THING_SUBCOMMAND_SET = "set";

    public ThingCommand() {
        super(THING_COMMAND_NAME, "Thing management.");
    }

    @Override
    public void configure(Subparser subparser) {
        addSubCommand(subparser.addSubparsers().addParser(THING_SUBCOMMAND_GET)
            .help("Gets a thing."), new GetCommand());

        addSubCommand(subparser.addSubparsers().addParser(THING_SUBCOMMAND_SET)
            .help("Sets a thing."), new SetCommand());
    }

    private void addSubCommand(Subparser parser, Command cmd) {
        // associate the subcommand with a Command object
        parser.addArgument("--subcommand").help(FeatureControl.SUPPRESS).setDefault(cmd);
        parser.description(cmd.getDescription());
        cmd.configure(parser);
    }

    @Override
    public void run(Bootstrap<?> bootstrap, Namespace namespace) throws Exception {
        Command cmd = namespace.get("subcommand");
        assert cmd != null;
        cmd.run(bootstrap, namespace);
    }

    class GetCommand extends Command {

        public GetCommand() {
            super(THING_SUBCOMMAND_GET, "Gets a thing.");
```

```
        }

        @Override
        public void configure(Subparser cmd) {
            super.configure(cmd);
            cmd.addArgument("--name").type(String.class).required(true).help("Name of the
thing");
        }

        @Override
        public void run(Bootstrap<?> bootstrap, Namespace namespace) throws Exception {
            String thingName = namespace.getString("name");
            System.out.println("Getting the thing named: " + thingName);
        }
    }

    class SetCommand extends Command {
        // ...
    }
}
```

Here's an example invocation of the command (assuming an app named `myapp`):

```
$ bin/myapp thing get --name mything
Getting the thing named: mything
```

Read Creating Commands online: https://riptutorial.com/dropwizard/topic/9323/creating-commands

# Chapter 3: Creating Resources

## Examples

### GET requests

```
@Path("/hello")
public class HelloResource {

    /**
     * A request to /hello would get the response "Hello World"
     */
    @GET
    public String exampleGet() {
        return "Hello World";
    }

    /**
     * A request to /hello/bob would get the response "Hello bob"
     */
    @Path("{name}")
    @GET
    public String exampleWithParameter(@PathParam("name") String name) {
        return "Hello " + name;
    }


}
```

### Custom responses

```
import javax.ws.rs.*;
import javax.ws.rs.core.Response;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

@Path("/alphabet/{letter}")
public class AlphabetResource {

    private final Map<String, String> alphabet;

    public AlphabetResource() {
        this.alphabet = new HashMap<>();
        this.alphabet.put("A", "Apple");
    }

    @GET
    public Response get(@PathParam("letter") String letter) {
        if (alphabet.containsKey(letter)) {
            return Response.ok(alphabet.get(letter)).build();
        } else {
            return Response.status(Response.Status.NOT_FOUND).build();
        }
    }
```

```
    @PUT
    public Response put(@PathParam("letter") String letter, String value) {
        if (alphabet.containsKey(letter)) {
            return Response.status(Response.Status.CONFLICT).build();
        } else {
            alphabet.put(letter, value);
            return Response.noContent().build();
        }
    }

    @POST
    public Response post(@PathParam("letter") String letter, String value) {
        if (alphabet.containsKey(letter) && Objects.equals(alphabet.get(letter), value)) {
            return Response.notModified().build();
        } else {
            alphabet.put(letter, value);
            return Response.noContent().build();
        }
    }
}
```

These can be called with the following curl commands

```
curl -v -X GET http://localhost:8080/alphabet/A
curl -v -X PUT http://localhost:8080/alphabet/A -d "Avacado"
curl -v -X PUT http://localhost:8080/alphabet/B -d "Banana"
curl -v -X POST http://localhost:8080/alphabet/A -d "Apple"
curl -v -X POST http://localhost:8080/alphabet/A -d "Avacado"
```

Read Creating Resources online: https://riptutorial.com/dropwizard/topic/7633/creating-resources

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with dropwizard | Community, Tom Reay |
| 2 | Creating Commands | Eron Wright |
| 3 | Creating Resources | Tom Reay |