Java EE Microservices with WildFly Swarm

Heiko Braun <hbraun@redhat.com>

Oct 2016

About me

- Heiko Braun
- Principal Software Engineer at Red Hat
- Focus on Java Middleware
- Java middleware components (WildFly/EAP, J2EE)
- Tools and frameworks for enterprise systems integration (Web Services, BPEL, SOA, BPM)

This evening

- The Context: Microservices and Java EE
- WildFly Swarm: Concepts, Ideas & workflow
- Code and Demo
- Outlook, Discussions

What are Microservices anyway?

Like SOA, but different …

- Microservices are different primarily due to innovations like:
  - Linux containers,
  - automated, elastic infrastructure, you know, the cloud
  - plus wide adoption of CI, continuous integration
  - and the growing adoption of DevOps principles & practices

"In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies."

– **Martin Fowler**, ThoughtWorks

What is Java EE anyway?

Perspectives on Java EE

- It’s different things to different people:
  - A collection of (useful) API’s
  - Technical capabilities of a system
  - A love/hate relationship (of the past)
  - (Existing) knowledge and expertise

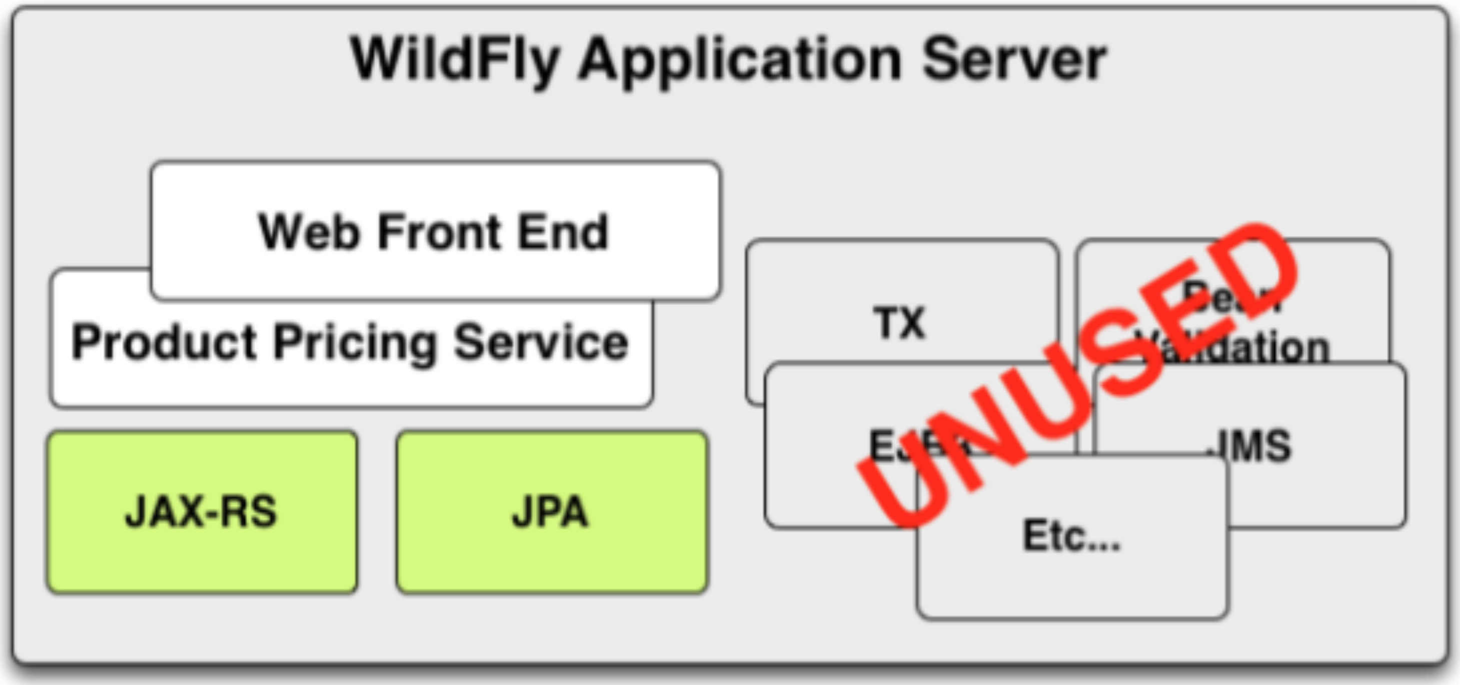
Hello WildFly Swarm

WildFly Swarm

- OSS Project sponsored by Red Hat
- Sidekick of Wildfly Application Server
- Small, but ambitious and friendly community
- Part of a bigger system of interrelated projects under the JBoss / Red Hat umbrella

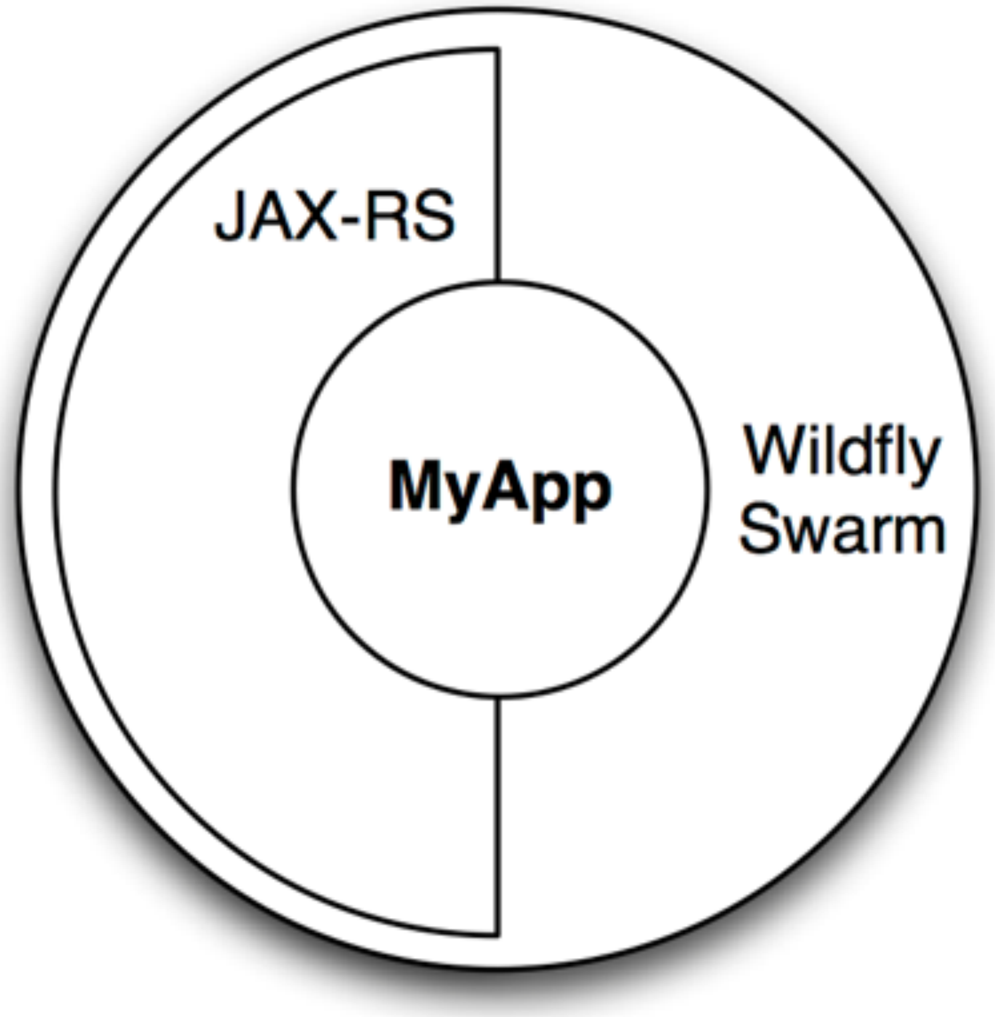
Rightsize your runtime

- Use the API's you want
- Include the capabilities you need
- Wrap it up for deployment

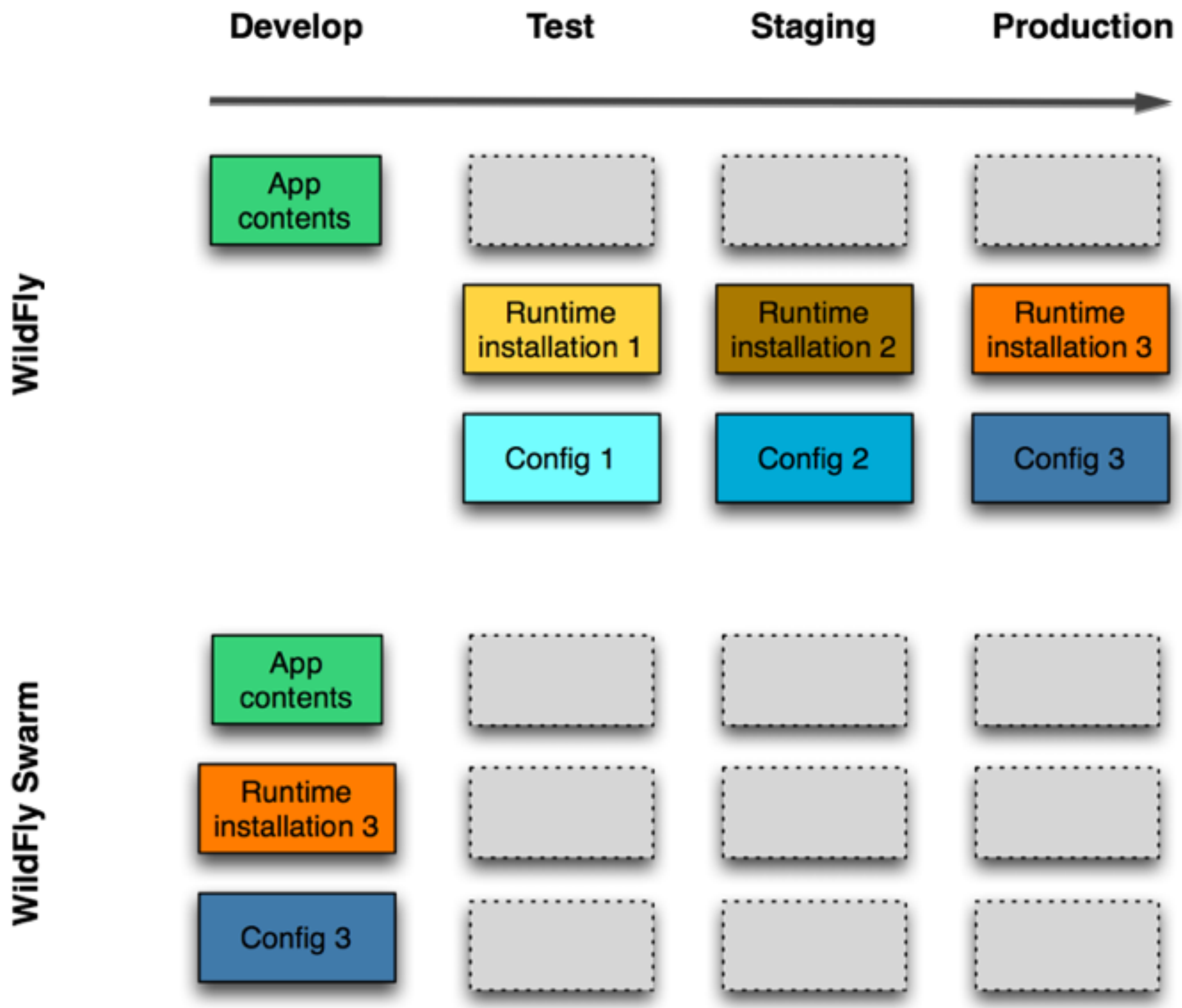


Self-contained JAR

- A single .jar file containing your application,
- the portions of WildFly required to support it,
- an internal Maven repository of dependencies,
- plus a shim to bootstrap it all



Self-contained executables



- Dev / Ops separation
- App / Runtime separation
- Configuration drift

- DevOps unification
- Self-containment
- Reproducibility

Fractions

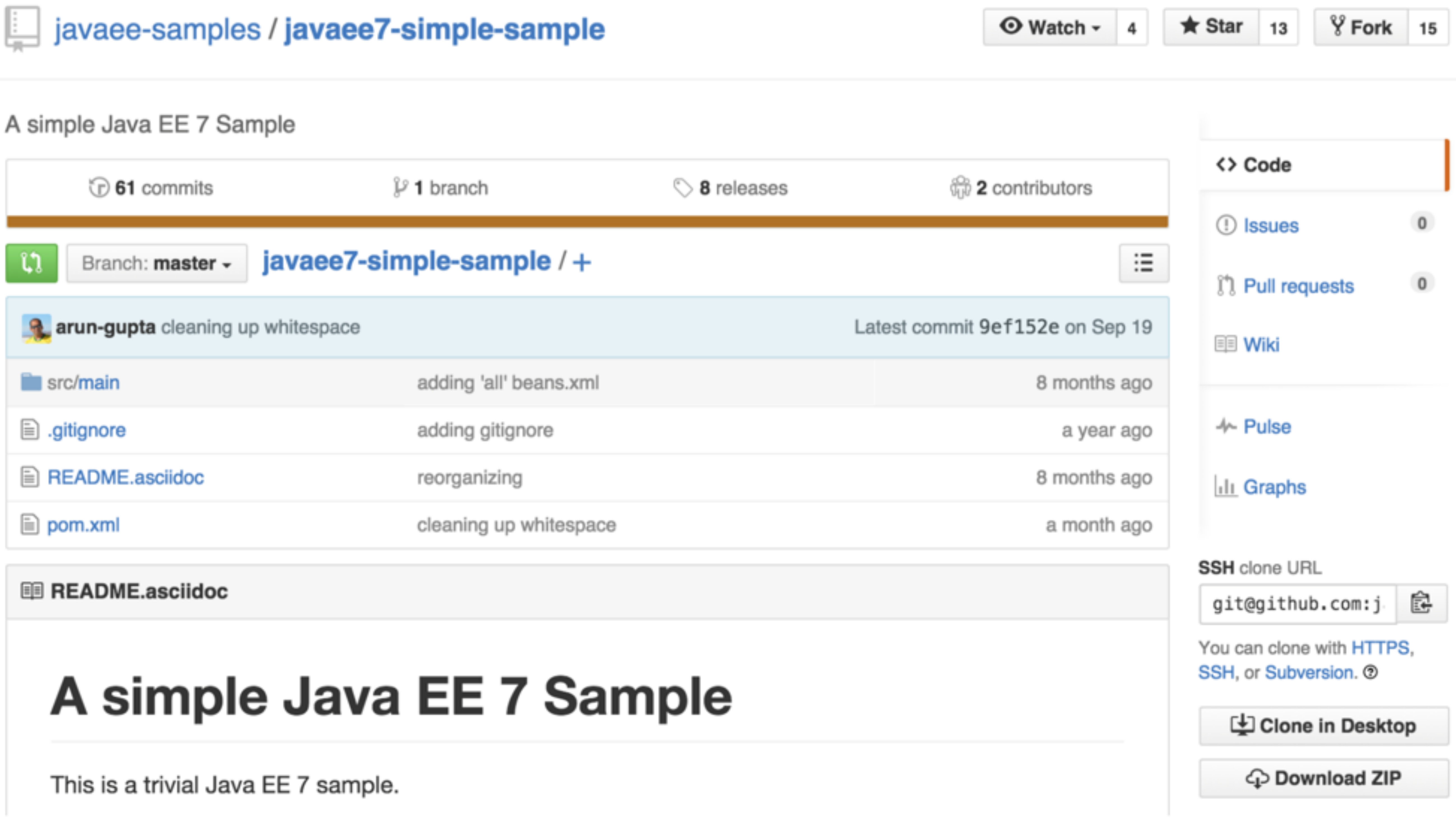
- A tangible unit, expressed as maven GAV
    - Focus on end users
    - To support the compositional aspect in Swarm
- Belongs to a dependency tree
- Ties to together multiple contents:
    - Modules, Subsystems, MSC services, Deployments

What Fractions can do

- Enable WildFly subsystems (i.e Logging, Datasources)
- Configure runtime components
- Integrate additional system capabilities (i.e Topology)
- Add API dependencies (i.e. JAX-RS)
- Provide deployments (i.e. Swagger)
- Alter deployments (i.e. SSO)

Converting a Java EE App to use WildFly Swarm

Code Example



javaee-samples / **javaee7-simple-sample**

Watch 4 | Star 13 | Fork 15

A simple Java EE 7 Sample

61 commits | 1 branch | 8 releases | 2 contributors

Branch: **master** javaee7-simple-sample / +

arun-gupta cleaning up whitespace — Latest commit 9ef152e on Sep 19

src/main	adding 'all' beans.xml	8 months ago
.gitignore	adding gitignore	a year ago
README.asciidoc	reorganizing	8 months ago
pom.xml	cleaning up whitespace	a month ago

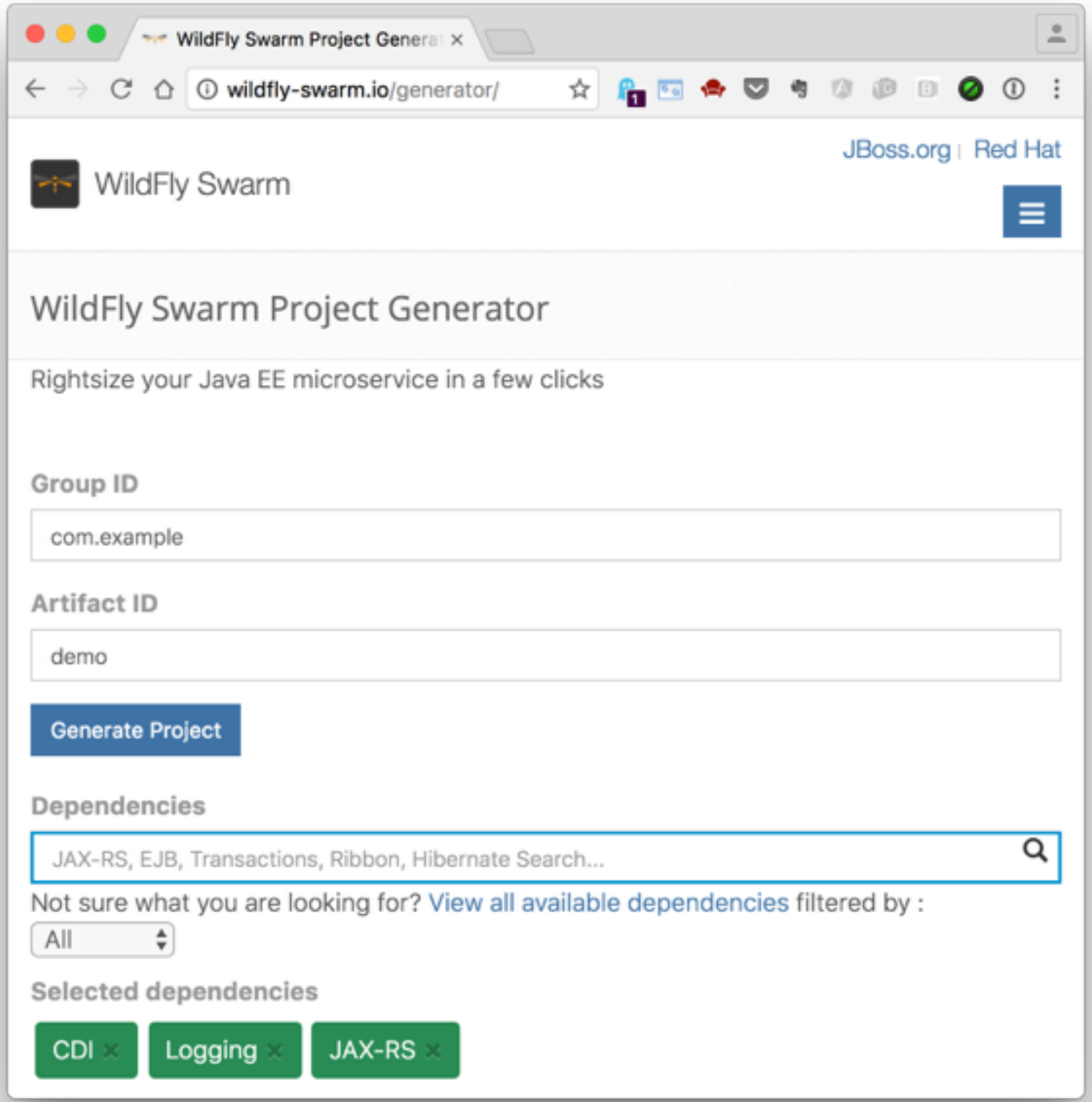
README.asciidoc

A simple Java EE 7 Sample

This is a trivial Java EE 7 sample.

Using custom fractions to build an application

Code example


WildFly Swarm Project Genera ×

wildfly-swarm.io/generator/

WildFly Swarm

JBoss.org | Red Hat

WildFly Swarm Project Generator

Rightsize your Java EE microservice in a few clicks

Group ID

com.example

Artifact ID

demo

Generate Project

Dependencies

JAX-RS, EJB, Transactions, Ribbon, Hibernate Search...

Not sure what you are looking for? View all available dependencies filtered by :

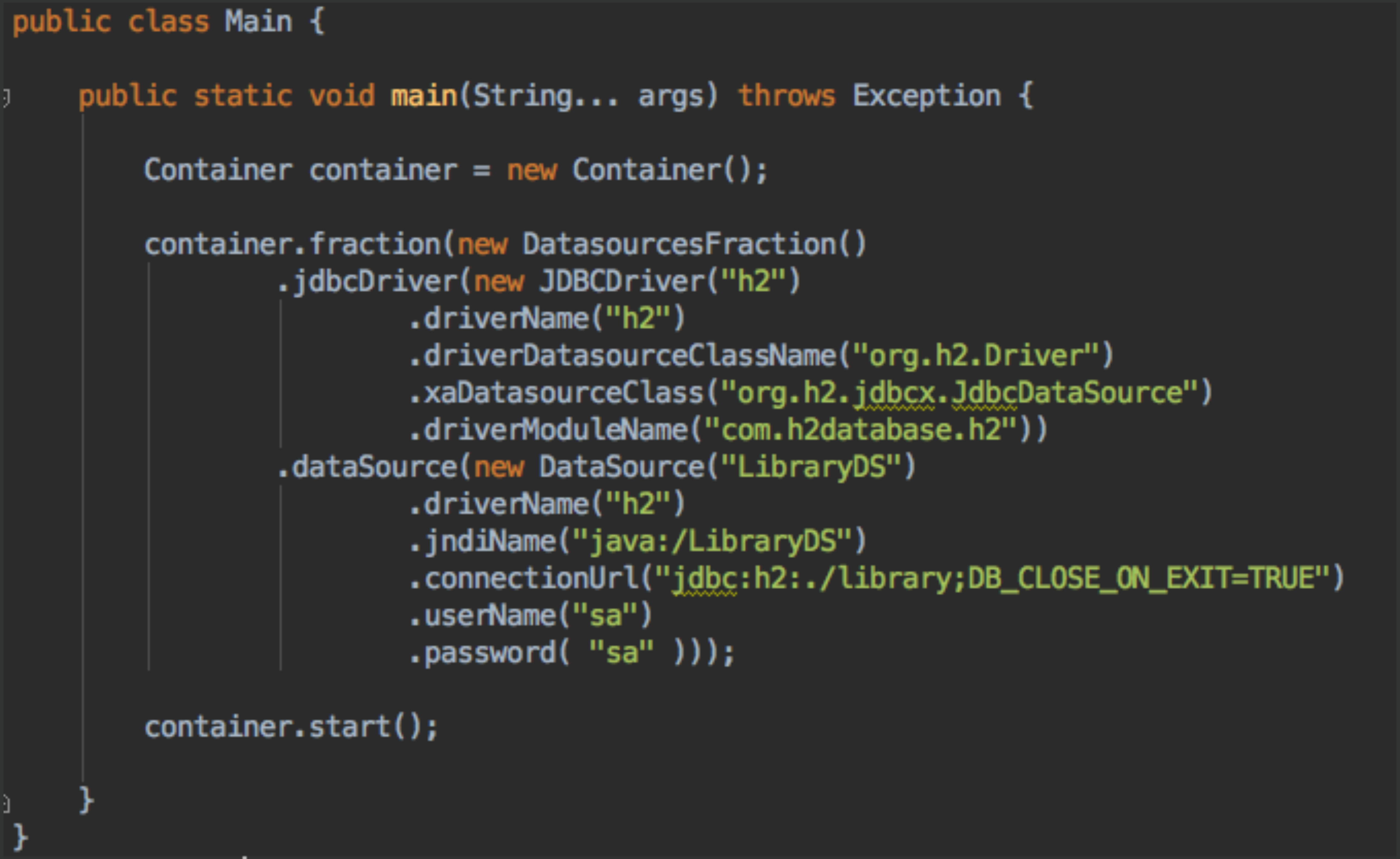
All

Selected dependencies

CDI × Logging × JAX-RS ×

Going beyond simple (and Java EE)

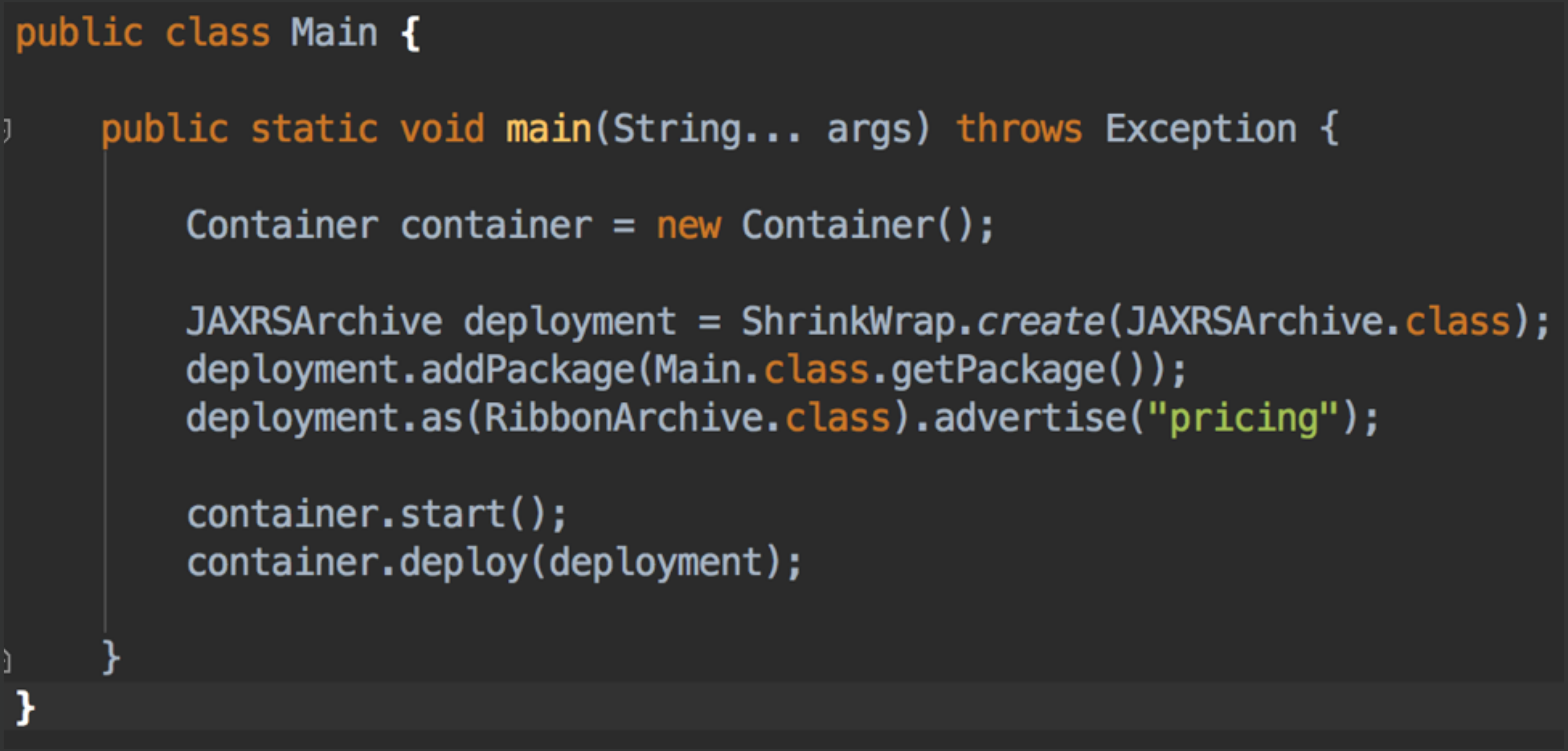
Custom Configuration



```java
public class Main {

    public static void main(String... args) throws Exception {

        Container container = new Container();

        container.fraction(new DatasourcesFraction()
                .jdbcDriver(new JDBCDriver("h2")
                        .driverName("h2")
                        .driverDatasourceClassName("org.h2.Driver")
                        .xaDatasourceClass("org.h2.jdbcx.JdbcDataSource")
                        .driverModuleName("com.h2database.h2"))
                .dataSource(new DataSource("LibraryDS")
                        .driverName("h2")
                        .jndiName("java:/LibraryDS")
                        .connectionUrl("jdbc:h2:./library;DB_CLOSE_ON_EXIT=TRUE")
                        .userName("sa")
                        .password( "sa" )));

        container.start();

    }
}
```

(alternatively use standalone.xml)

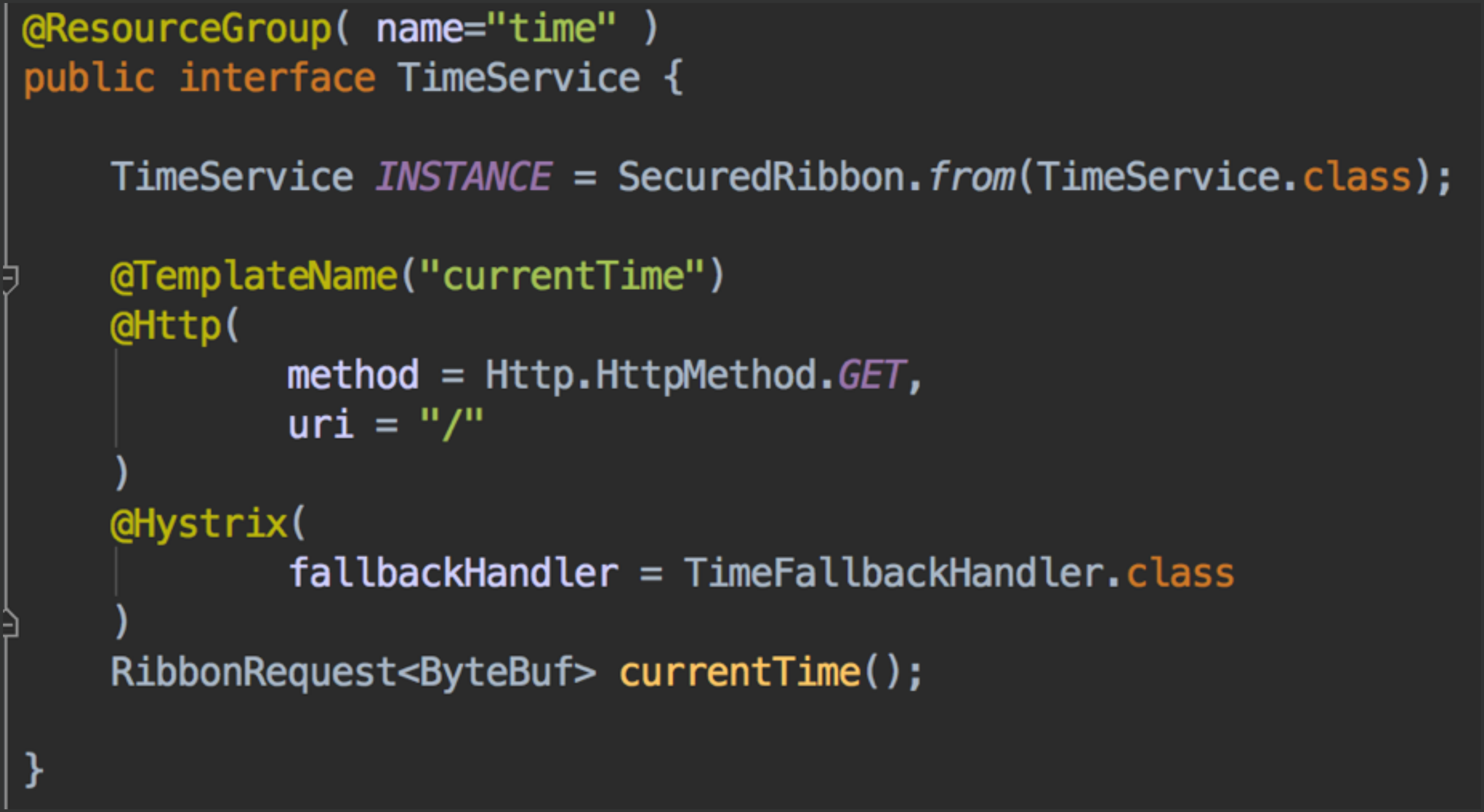
Advertising Services



```java
public class Main {

    public static void main(String... args) throws Exception {

        Container container = new Container();

        JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
        deployment.addPackage(Main.class.getPackage());
        deployment.as(RibbonArchive.class).advertise("pricing");

        container.start();
        container.deploy(deployment);

    }
}
```

(supports different service registries)

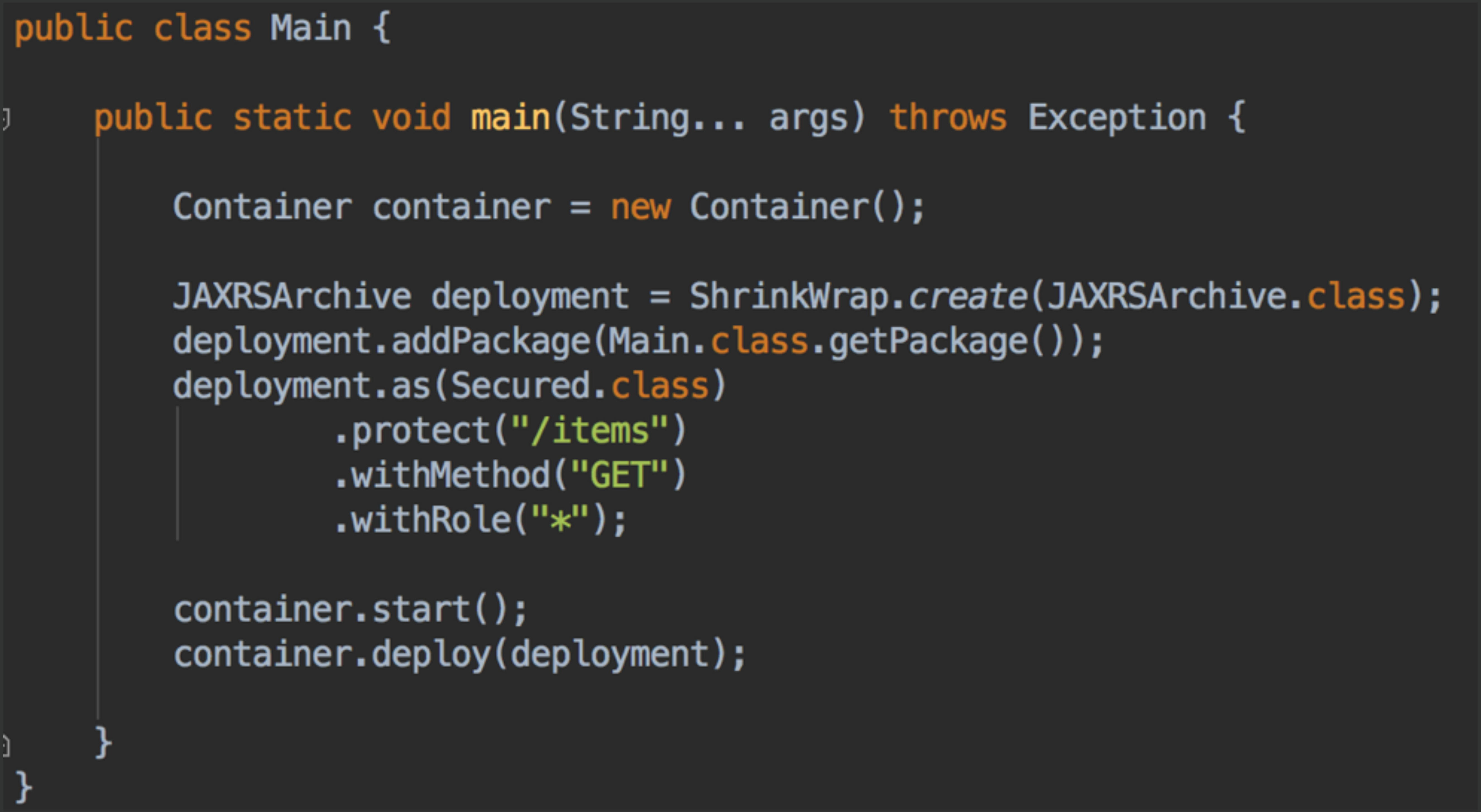
Load Balancing & Circuit Breaking



```java
@ResourceGroup( name="time" )
public interface TimeService {

    TimeService INSTANCE = SecuredRibbon.from(TimeService.class);

    @TemplateName("currentTime")
    @Http(
            method = Http.HttpMethod.GET,
            uri = "/"
    )
    @Hystrix(
            fallbackHandler = TimeFallbackHandler.class
    )
    RibbonRequest<ByteBuf> currentTime();

}
```

(Integration of Ribbon with Topology. Supports Hystrix)

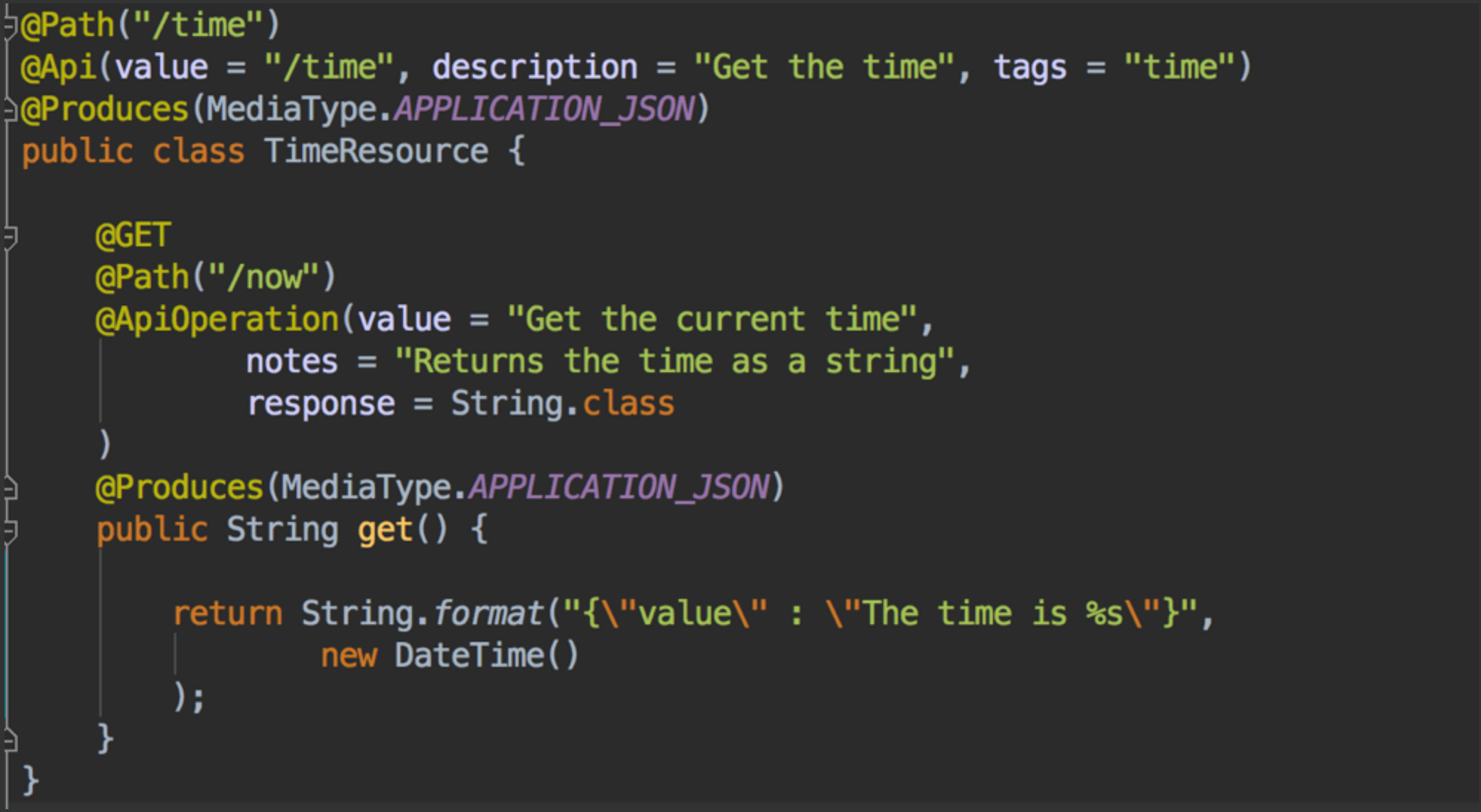
Securing Access to Services



```
public class Main {

    public static void main(String... args) throws Exception {

        Container container = new Container();

        JAXRSArchive deployment = ShrinkWrap.create(JAXRSArchive.class);
        deployment.addPackage(Main.class.getPackage());
        deployment.as(Secured.class)
                .protect("/items")
                .withMethod("GET")
                .withRole("*");

        container.start();
        container.deploy(deployment);

    }
}
```

(provided by Keycloak: OpenID, SAML, Social Login, OAuth, LDAP, Active Directory)

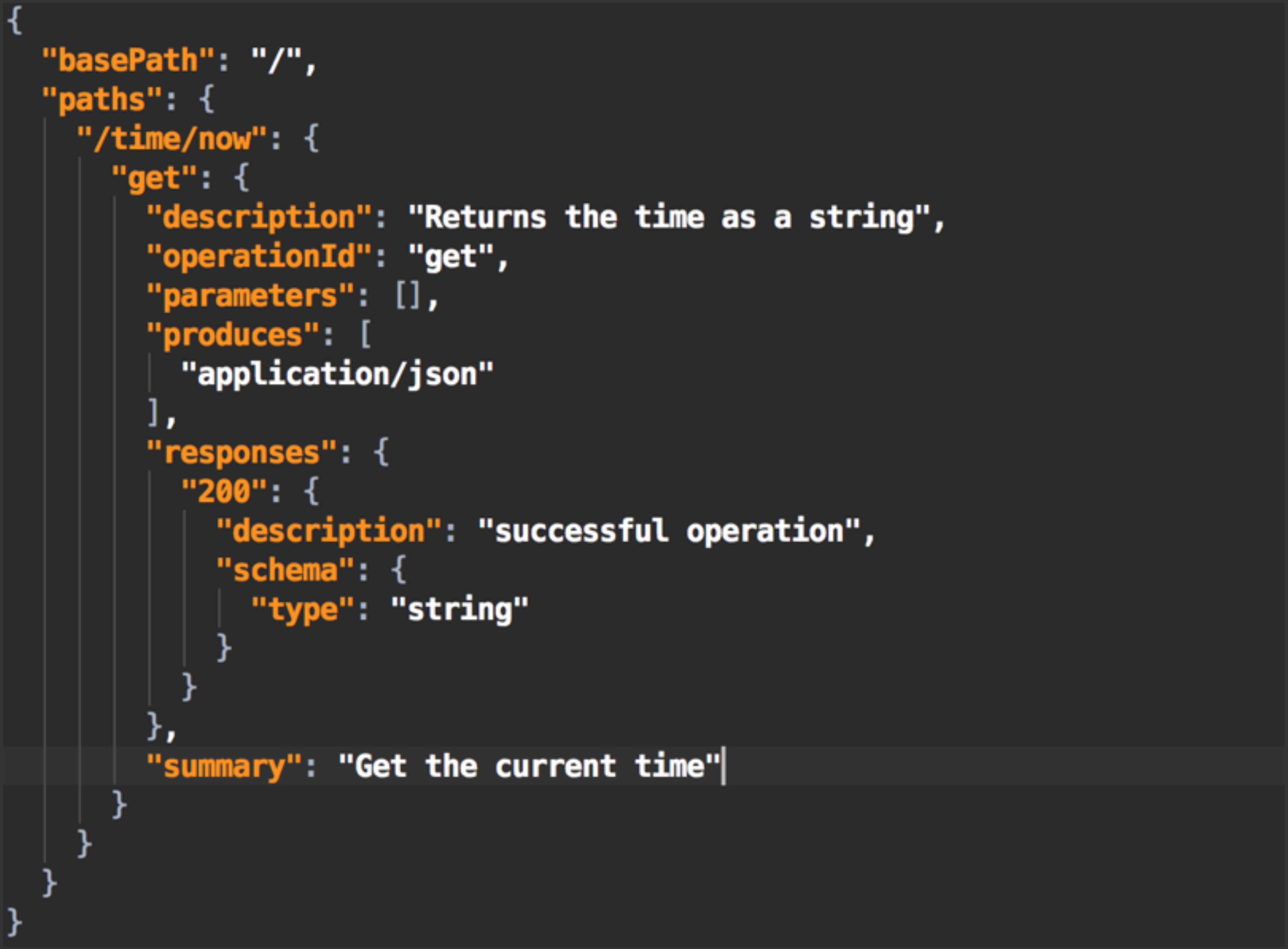
Publishing Service Interface Descriptions



```java
@Path("/time")
@Api(value = "/time", description = "Get the time", tags = "time")
@Produces(MediaType.APPLICATION_JSON)
public class TimeResource {

    @GET
    @Path("/now")
    @ApiOperation(value = "Get the current time",
            notes = "Returns the time as a string",
            response = String.class
    )
    @Produces(MediaType.APPLICATION_JSON)
    public String get() {

        return String.format("{\"value\" : \"The time is %s\"}",
                new DateTime()
        );
    }
}
```

(provided by Swagger)

```
$ curl http://localhost:8080/swagger.json
```



```
{
  "basePath": "/",
  "paths": {
    "/time/now": {
      "get": {
        "description": "Returns the time as a string",
        "operationId": "get",
        "parameters": [],
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "successful operation",
            "schema": {
              "type": "string"
            }
          }
        },
        "summary": "Get the current time"
      }
    }
  }
}
```

Other Noteworthy Features

- Testing:
  - Arquillian (in container, web driver)
  - Consumer-Driven Contracts
    (expressing and asserting expectations of a provider contract)
- Logging & Monitoring
  - Simple REST interface on each node
  - Centralised Logging with Logstash
  - Push Runtime Data to Hawkular,Influx,etc
- Remote Management
  - CLI (full access to the server config and runtime state)

Get Involved

- Project Home: http://wildfly-swarm.io
- GitHub: https://github.com/wildfly-swarm
- Twitter: @wildflyswarm
- Freenode: @wildfly-swarm
- Issues: https://issues.jboss.org/projects/SWARM (see ‘getting-started’ labels)