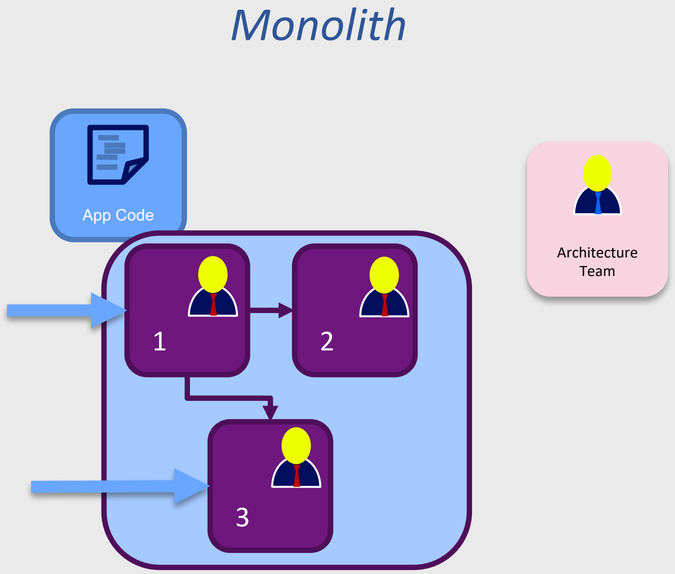
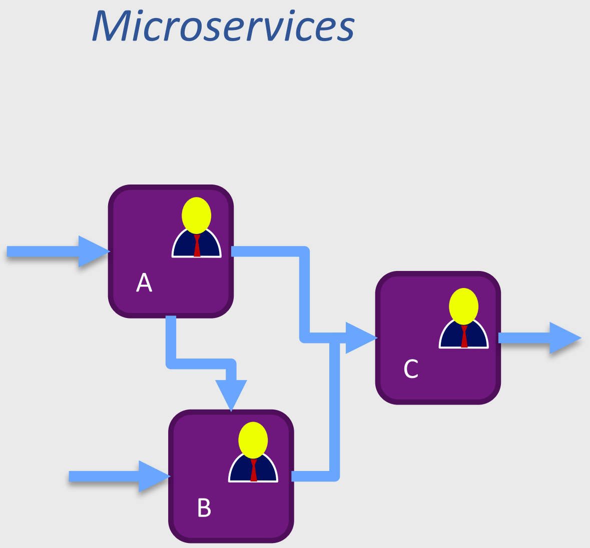
Eclipse MicroProfile

Microservices





Who is Using Microservices?

*About **three-quarters** of developers are at least looking at microservices architecture for some workload. Yet, when asked more specifically about their use in production applications, the numbers drop: **34 percent** in a Lightbend survey and **26 percent** in a DZone survey. Adoption of microservices is closely correlated with use of DevOps, CI/CD and containers. Ditto with serverless.* --Lawrence Hecht*

~ 30%

* https://thenewstack.io/week-numbers-microservices-data-perfect-together/

How can we help?

What can we do to advance microservice development in the Enterprise Java space?

-Java EE Community, early 2016

MICROPROFILE™
OPTIMIZING ENTERPRISE JAVA

Eclipse microProfile Community

IBM

LJC

redhat

Tomitribe

payara

SOUJava
sociedade de usuários java

hazelcast

FUJITSU

kumuluzEE

Hammock
CDI Based MicroServices

ORACLE

Lightbend

Microsoft

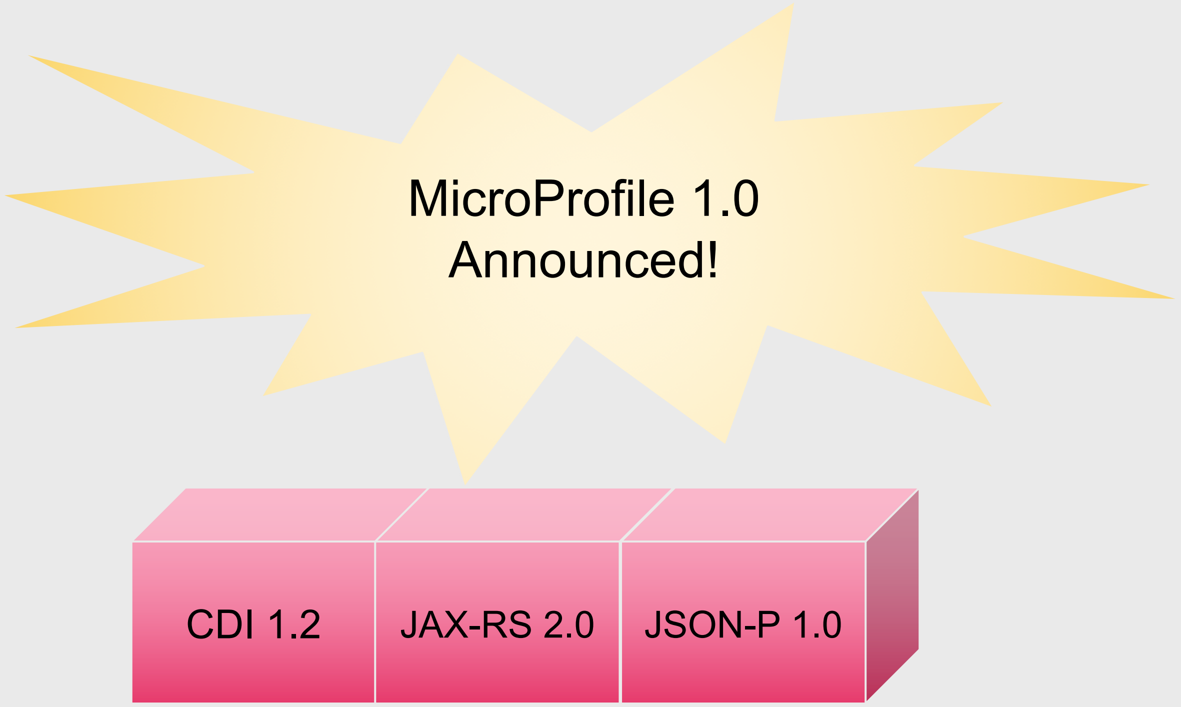
References

- https://microprofile.io
  - https://projects.eclipse.org/projects/technology.microprofile
  - https://microprofile.io/projects/
  - https://wiki.eclipse.org/MicroProfile/Implementation

cloud-native microservice

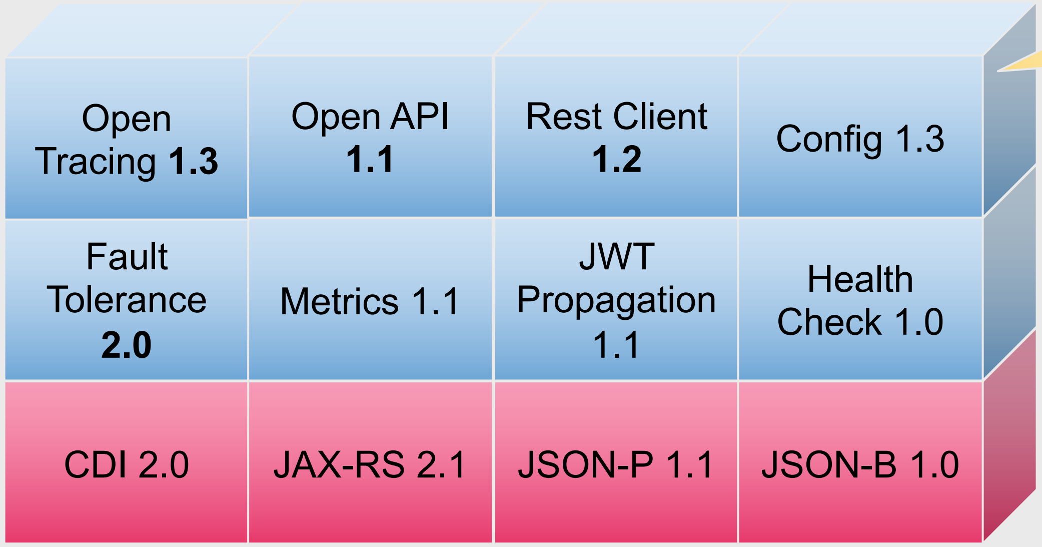
1. RESTful
2. Configurable
3. Fault tolerance
4. Can be discovered
5. Secure
6. Traceable, monitorable
7. Able to communicate with the cloud infrastructure

JavaOne 2016



Basic Building Blocks for Microservices

Fast-forward two years...



MicroProfile **2.2**

MicroProfile 2.2

8 Platform Releases!

21 Component Releases!

MICROPROFILE™

OPTIMIZING ENTERPRISE JAVA

- ✓Open specifications
- ✓Wide vendor support
- ✓REST services
- ✓OpenAPI support
- ✓Security (JWT)
- ✓Fault Tolerance
- ✓Configuration
- ✓Metrics
- ✓Health
- ✓Open Tracing

Open Liberty Project

Hammock CDI Based MicroServices

Apache TomEE

redhat Red Hat OpenShift

kumuluzEE

Launcher

payara®

WebSphere Liberty

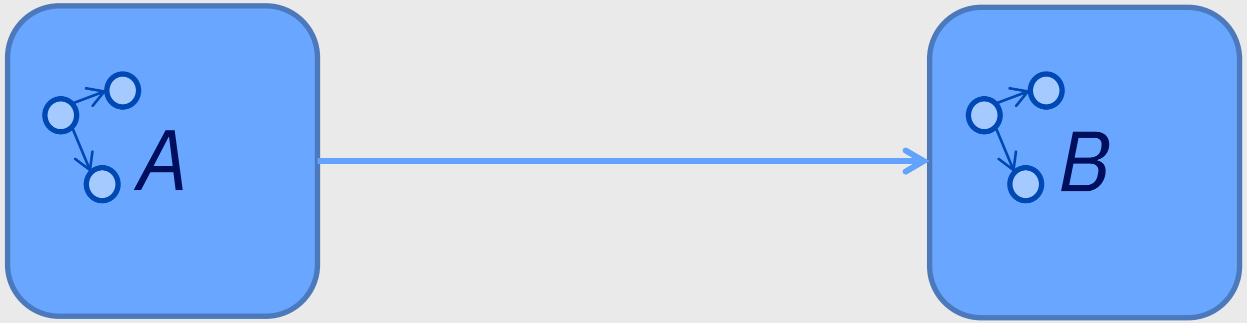
helidon.io

SmallRye

Thorntail

https://wiki.eclipse.org/MicroProfile/Implementation

CDI

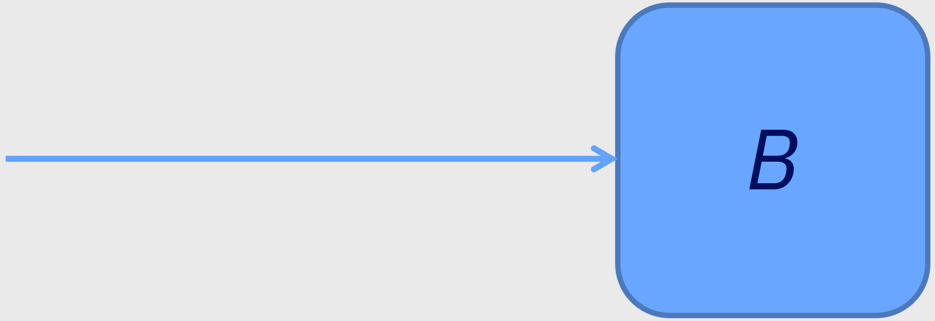


```java
@Path("/orders")
public class OrdersResource {

  @Inject
  CoffeeShop coffeeShop;


  ...
}
```

CDI Notes

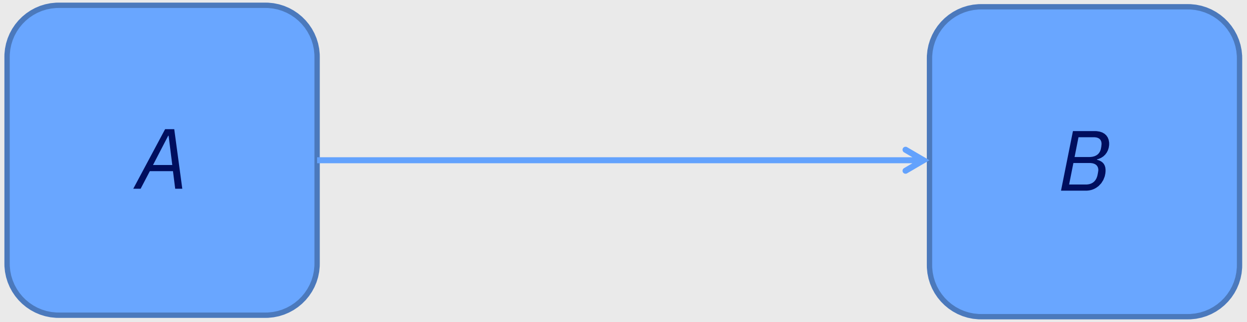
- Create a singleton bean with @ApplicationScoped
- Create a bean per-request with @RequestScoped

JAX-RS



```java
@Path("/brews")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class BrewsResource {
  @POST
  public Response startCoffeeBrew(CoffeeBrew brew) {...}
}
```

MicroProfile REST Client



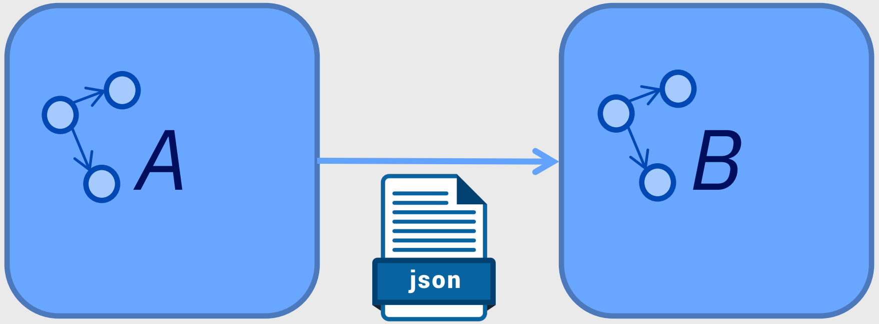
```
@RegisterRestClient
@Path("/resources/brews")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public interface BaristaClient {
  @POST
  public Response startCoffeeBrew(CoffeeBrew brew);
}
```

```
test.BaristaClient/mp-rest/url=http://localhost:9080/system
```

MicroProfile Rest Client Notes

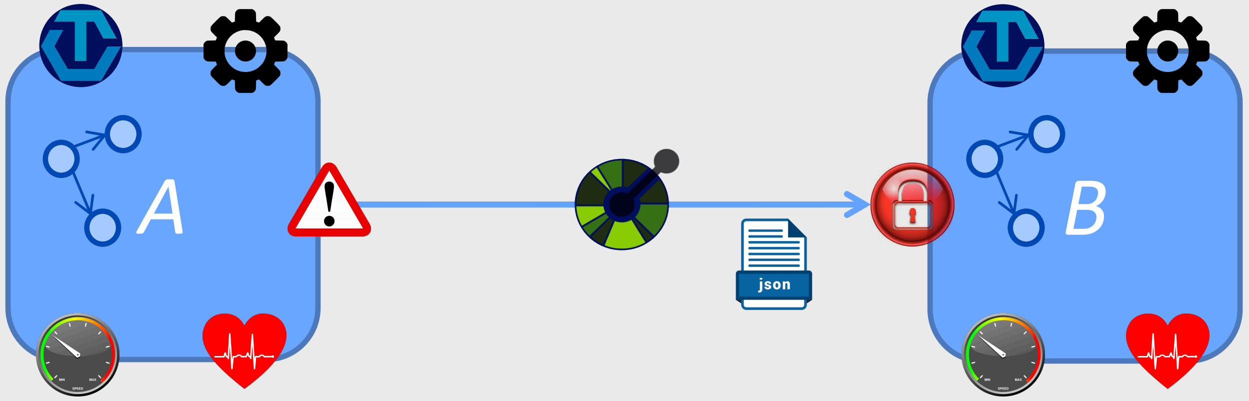
- Use @Inject @RestClient to actually use the client
- Use @RegisterProvider to hook into JAX-RS processing
  - For example, use ResponseExceptionMapper to map an HTTP 404 to an exception
- By default, the rest client is @Dependent, so it picks up the scope that it’s contained in. So if you @Inject it into an @ApplicationScoped bean, then you have just one rest client.

JSON-B & JSON-P



```
public class CoffeeBrew {

  private CoffeeType type;

  public CoffeeType getType() {
    return type;
  }

  public void setType(CoffeeType type) {
    this.type = type;
  }
}

@POST
@Consumes(MediaType.APPLICATION_JSON)
public Response
  startCoffeeBrew(CoffeeBrew brew) {

  CoffeeType coffeeType = brew.getType();
}
```

MicroProfile OpenTracing



JAX-RS methods are automatically traced by default

```
@Traced
public void startBrew(CoffeeType coffeeType) {
   ...
}
```

MicroProfile Config

Static Config

```
@Inject
@ConfigProperty(name=”openweathermap.appid")
private String owmAppid;
```

microprofile-config.properties

openweathermap.appid=66b1f66ea5468ba00130
9f82123571c0
dukes.zipcode=94103

overrides

Dynamic Config

```
@Inject
@ConfigProperty(name=”dukes.zipcode")
private String dukesZipcode;
```

Java Options

-Ddukes_zipcode=55906

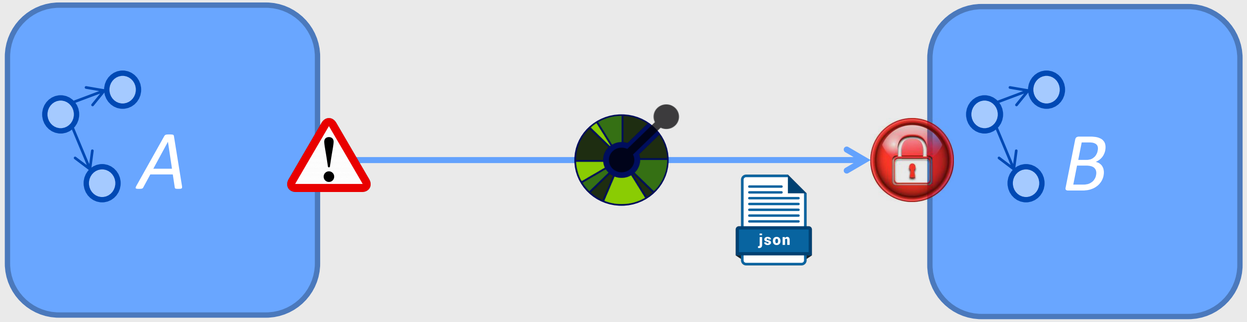
Server.xml

<variable name="dukes.zipcode" value="55906" />

MicroProfile Config Notes

- Lookup order:
  - Annotation defaultValue
  - META-INF/microprofile-config.properties
  - Environment variables
  - -D properties
- Build a custom config loader (e.g. JSON) by implementing ConfigSource and define the class in META-INF/services/org.eclipse.microprofile.config.spi.ConfigSource
- Use javax.inject.Provider to lookup the configuration every time to make it dynamic
- Implement org.eclipse.microprofile.config.spi.Converter<T> for custom types

MicroProfile Fault Tolerance

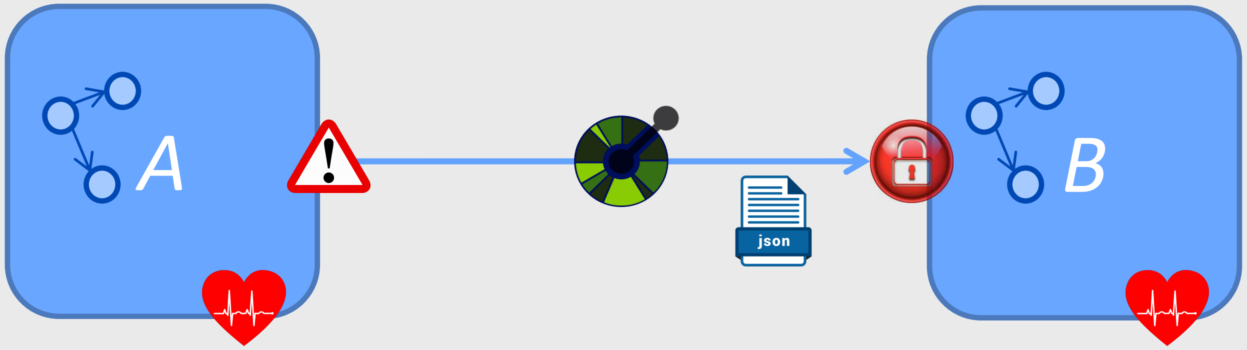


```
@Retry(
  retryOn = TimeoutException.class,
  maxRetries = 4,
  maxDuration = 10,
  durationUnit = ChronoUnit.SECONDS)
public void startCoffeeBrew(CoffeeBrew brew) {
  Response response = baristaClient.startCoffeeBrew(brew);
}
```

MicroProfile Fault Tolerance Notes

- @Retry:
  - Use the delay option to wait between retries. Use the jitter option to add jitter to the delay.
  - Use the abortOn option to fail immediately on certain exceptions.
- @Timeout(value = 2, unit = ChronoUnit.SECONDS)
- @CircuitBreaker to fail immediately for delay milliseconds when failure threshold is met (failureRatio within rollingWindow). After delay, change to half open until successThreshold successive requests succeed.
- @Bulkhead @Asynchronous to limit the number of concurrent requests using a thread pool
- @Fallback service runs when @Bulkhead fills up or exception is thrown on a service or @CircuitBreaker is open

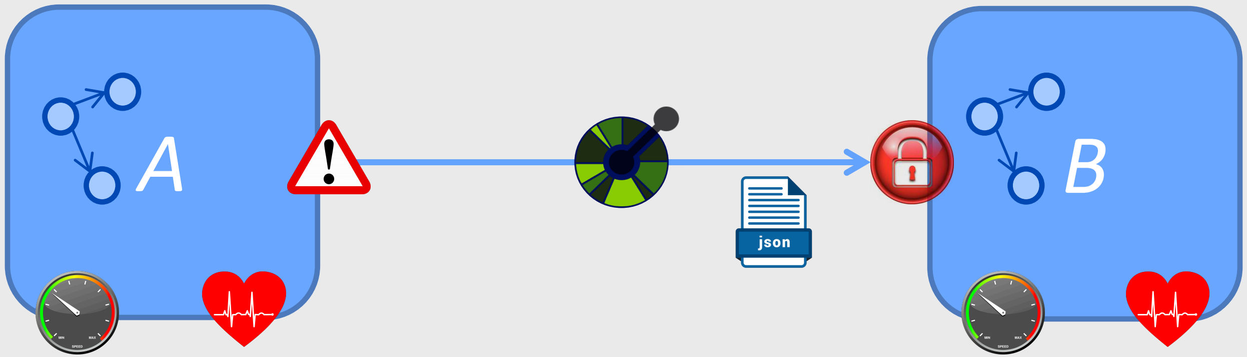
MicroProfile Health



```java
@Health
@ApplicationScoped
public class HealthResource implements HealthCheck {
  public boolean isHealthy() {...}

  @Override
  public HealthCheckResponse call() {
    if (!isHealthy()) {
      return HealthCheckResponse.named(...).down().build();
    }
    return HealthCheckResponse.named(...).up().build();
  }
}
```

```json
{
 "checks": [
   {
     "data": {
        "barista service": "available"
      },
      "name": "CoffeeShopHealth",
      "state": "UP"
    }
  ],
  "outcome": "UP"
}
```

MicroProfile Metrics

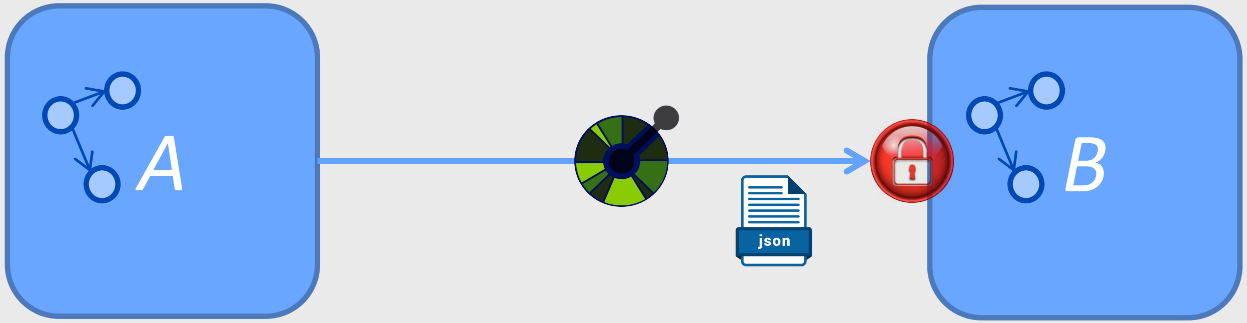


```
@POST
@Metered
public Response orderCoffee(@Valid @NotNull CoffeeOrder order) {
  ...
}
```

MicroProfile Metric Notes

- @Timed to track frequency and duration
- @Counted to track number of calls
- @Gauged to return some application-scoped number of something (e.g inventory)

MicroProfile JWT



```
@POST
@RolesAllowed({ "admin", "coffee-shop" })
public Response startCoffeeBrew(CoffeeBrew brew) {...}
```

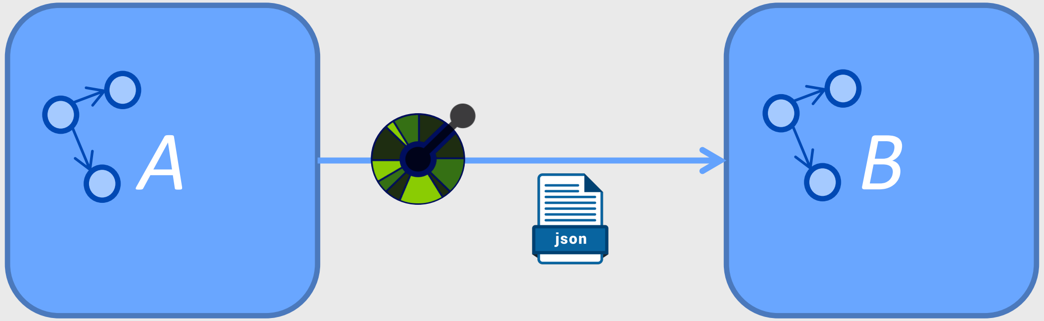
MicroProfile JWT Notes

- @Inject JsonWebToken into the @RequestScoped JAX-RS class to get more details or access custom claims
- @Context SecurityContext securityContext JAX-RS method parameter for details on the authentication

MicroProfile OpenAPI

/openapi/

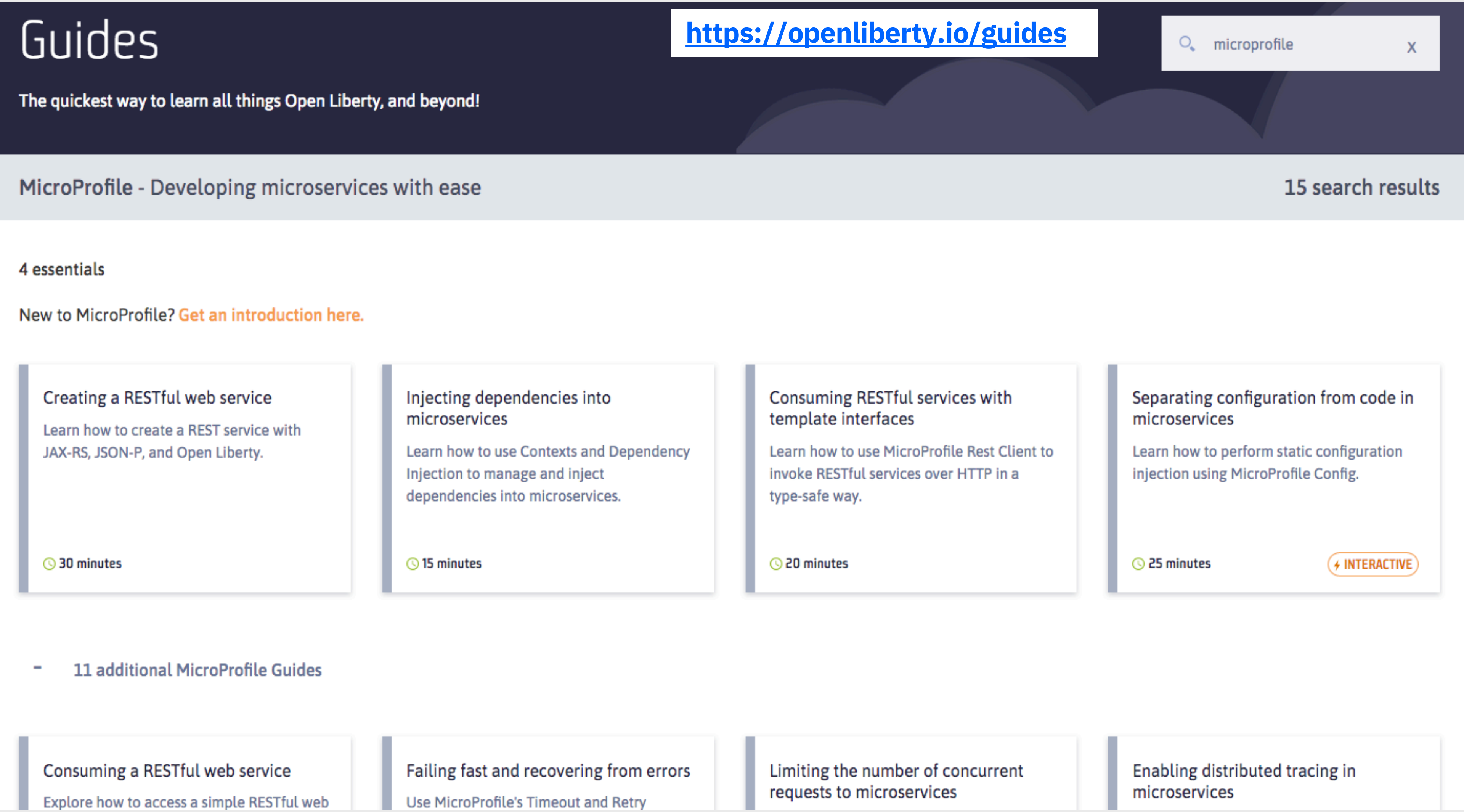
```
openapi: 3.0.0
info:
  title: Deployed APIs
  version: 1.0.0
servers:
- url: http://grahams-mbp-2.lan:9081/barista
paths:
  /resources/brews:
    post:
      operationId: startCoffeeBrew
      requestBody:
        content:
          application/json:
            schema:
              $ref:
'#/components/schemas/CoffeeBrew'
      responses:
        default:
          description: default response
```



```
components:
  schemas:
    CoffeeBrew:
      type: object
      properties:
        type:
          type: string
          enum:
          - ESPRESSO
          - LATTE
          - POUR_OVER
```

MicroProfile OpenAPI Notes

- Based on Swagger
- @Operation to describe the overall API
- @Parameter to describe parameters
- @Schema to describe POJOs
- @APIResponses to describe the possible responses:
  - `@APIResponse(responseCode = "200", description = "JVM system properties of a particular host", content = @Content(mediaType = "application/json")), schema = @Schema(implementation = Properties.class)`
  - `@APIResponse(responseCode = "404", description = "Missing description", content = @Content(mediaType = "text/plain"))`



https://openliberty.io/guides

Guides

microprofile x

The quickest way to learn all things Open Liberty, and beyond!

MicroProfile - Developing microservices with ease

15 search results

4 essentials

New to MicroProfile? Get an introduction here.

Creating a RESTful web service
Learn how to create a REST service with JAX-RS, JSON-P, and Open Liberty.
30 minutes

Injecting dependencies into microservices
Learn how to use Contexts and Dependency Injection to manage and inject dependencies into microservices.
15 minutes

Consuming RESTful services with template interfaces
Learn how to use MicroProfile Rest Client to invoke RESTful services over HTTP in a type-safe way.
20 minutes

Separating configuration from code in microservices
Learn how to perform static configuration injection using MicroProfile Config.
25 minutes
INTERACTIVE

- 11 additional MicroProfile Guides

Consuming a RESTful web service
Explore how to access a simple RESTful web

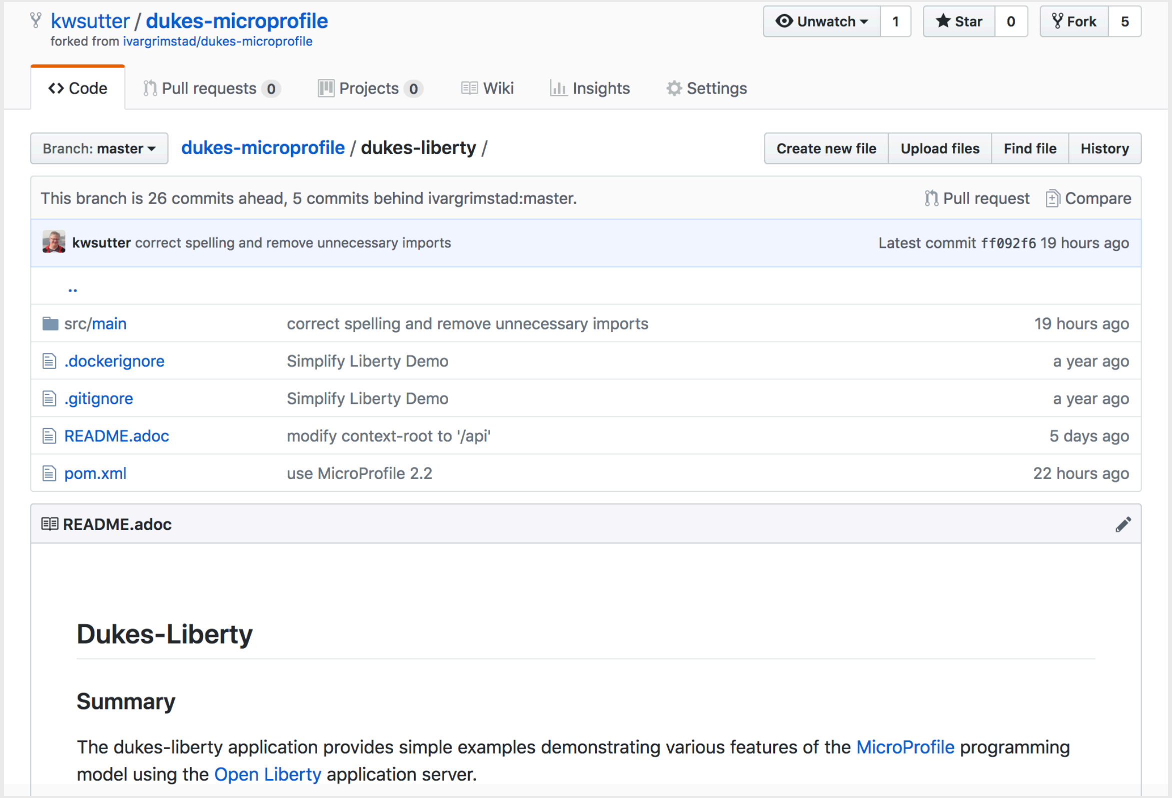
Failing fast and recovering from errors
Use MicroProfile's Timeout and Retry

Limiting the number of concurrent requests to microservices

Enabling distributed tracing in microservices

Sample

https://github.com/kwsutter/dukes-microprofile/tree/master/dukes-liberty



This branch is 26 commits ahead, 5 commits behind ivargrimstad:master.

kwsutter correct spelling and remove unnecessary imports — Latest commit ff092f6 19 hours ago

..		
src/main	correct spelling and remove unnecessary imports	19 hours ago
.dockerignore	Simplify Liberty Demo	a year ago
.gitignore	Simplify Liberty Demo	a year ago
README.adoc	modify context-root to '/api'	5 days ago
pom.xml	use MicroProfile 2.2	22 hours ago

README.adoc

Dukes-Liberty

Summary

The dukes-liberty application provides simple examples demonstrating various features of the MicroProfile programming model using the Open Liberty application server.

Thanks