Java and MicroProfile on Azure: Building microservices in style

James Falkner
Senior App Dev Technical Evangelist, Red Hat
@schtool

Brian Benz
Senior Cloud Developer Advocate, Microsoft
@bbenz
• What is MicroProfile
• The MicroProfile Community
• MicroProfile Development
• Thorntail (née WildFly Swarm)
• Demo Time!
• More resources
• Q&A
- Eclipse MicroProfile is an open-source community specification for Enterprise Java microservices

- A community of individuals, organizations, and vendors collaborating within an open source (Eclipse) project to bring microservices to the Enterprise Java community
Community - individuals, organizations, vendors
Current MicroProfile implementations
Eclipse MicroProfile 2.2 (Feb 2019)

- Open Tracing 1.3
- Open API 1.1
- Rest Client 1.2
- Config 1.3
- Fault Tolerance 2.0
- Metrics 1.1
- JWT Propagation 1.1
- Health Check 1.0
- CDI 2.0
- JSON-P 1.1
- JAX-RS 2.1
- JSON-B 1.0

Legend:
- New
- Updated
- No change from last release (MicroProfile 2.1)
Eclipse MicroProfile 2.2 Released!

On February 12, 2019, MicroProfile 2.2 was released. Offered in the release:

- Java EE 8 continued alignment
- A richer feature set for Fault Tolerance, Open Tracing, Open API, and Rest Client
- CDI-based and programmatic interfaces
- Test Compatibility Kit (TCK), Javadoc, PDF doc for download

Other news:

- MicroProfile Starter (Beta) now available
- Reactive Streams Operators project plans to release standalone
- MicroProfile GraphQL project has been approved to move to the MicroProfile sandbox
• Leverage Java EE expertise
• Open Standard
• Microservices focus
• Optimized for Containers
• Super Lightweight
• uber-jar and war support
• Implements MicroProfile 1.2*

* 2.0 planning in progress, Q1 CY19

$ java -jar my_microservice.jar
or
$ java -jar custom-runtime.jar myapp.war
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>io.thorntail</groupId>
      <artifactId>bom</artifactId>
      <version>${version.thorntail}</version>
      <scope>import</scope>
      <type>pom</type>
    </dependency>
  </dependencies>
</dependencyManagement>
Externalized Configuration

@Inject
@ConfigProperty(name = "myapp.default-name")
private String defaultName;

Health Probes

@Health
@ApplicationScoped
public class GoodHealthCheck implements HealthCheck {
  public HealthCheckResponse call() {
    return HealthCheckResponse
      .named("MyHealthCheck")
      .withData("someKey", "aValue")
      .up()
      .build();
  }
}

Externalized Configuration

Health Probes
@Counted(name = "hCount",
    absolute = true,
    description = "# calls to /myendpoint",
    monotonic = true)

@GET
@Path("/myendpoint")
public Response myEndpoint {}

@Retry(maxRetries = 2)
@Fallback(fallbackMethod = "greetingFallback")
public Response greeting() {

    return Response.ok()
        .entity(new Greeting("Greetings from a fallback")).build();
}

public Response greetingFallback() {
    return Response.ok();
}
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>jaeger</artifactId>
</dependency>
Demo Time!
Get Involved!

- Google Groups
- MicroProfile Projects
- Bi-Weekly & Quarterly General community Meetings
- YouTube Channel
- Video Hangouts
- MicroProfile - https://microprofile.io/
- Projects: https://microprofile.io/projects/
- Join the Discussion - https://groups.google.com/forum/#!forum/microprofile

- Thorntail (née WildFly Swarm)
  - http://wildfly-swarm.io/documentation/
- Reference Architecture for OpenShift on Azure:
- Java on Azure – http://azure.com/java
- Docs https://docs.com/
Questions?
(Don’t forget the evals!)
(Follow us on Twitter!)

James Falkner
jfalkner@redhat.com
@school

Brian Benz
bbenz@microsoft.com
@bbenz
THANK YOU

linkedin.com/company/Red-Hat
facebook.com/RedHatInc
youtube.com/user/RedHatVideos
twitter.com/RedHat