MicroProfile Health

This cheat sheet covers the basics of MicroProfile Health specification uses.

DEFINING A HEALTH CHECK

1. Implement `org.eclipse.microprofile.health.HealthCheck` interface.
2. Mark the implementation as a CDI bean (e.g., `@ApplicationScoped`).
3. Mark the implementation with one or more of the defined health qualifiers (`@Readiness`, `@Liveness`, `@Health`).
4. Implement the `call()` method.

DIFFERENT KINDS OF HEALTH CHECKS

Readiness health check

Allows third party services to know if the application is ready to process requests or not.

`Qualifier`

```
@Readiness
```

`Endpoint`

```
/health/ready
```

Liveness health check

Allows third party services to determine if the application is running. Failure means that application can be discarded.

`Qualifier`

```
@Liveness
```

`Endpoint`

```
/health/live
```

Backward compatible health check

Warning: This kind of health check is currently deprecated and will be removed in the future version of the specification.

`Qualifier`

```
@Health
```

`Endpoint`

```
/health
```

Note: The `/health` endpoint aggregates all defined procedures (defined with `@Health`, `@Liveness`, `@Readiness`).

Note: Multiple kinds of health check can be combined on a single health check procedure.

CONSTRUCTING A `HealthCheckResponse`

Use one of the static methods of `HealthCheckResponse` to either:

`contruct the HealthCheckResponse object directly:`

```
HealthCheckResponse.up("successful-check")
```

or

```
HealthCheckResponse.up(“failed-check")
```

`contruct the HealthCheckResponseBuilder using following methods for a more detailed response specification:`

```
HealthCheckResponse#named(String)
```

or

```
HealthCheckResponse#builder()
```

Note: Every health check is required to have a name.

Adding custom data values to the health check response

With the `HealthCheckResponseBuilder` you can use one of the overloadings of the `withData` method to add a custom key-value pairs to your health check response:

```
public class CheckDiskspace implements HealthCheck {
    @Override
    public HealthCheckResponse call() {
        return HealthCheckResponse.named("diskspace")
            .withData("free", "780mb")
            .up()
            .build();
    }
}
```

HTTP STATUS CODES OVERVIEW

`200`

health check with a positive status (`UP`)

`503`

health check with a negative status (`DOWN`)

`500`

error in the procedure execution

Author Martin Stefanko
Senior Software Engineer
Middleware Runtimes Sustaining Engineering Team