MicroProfile Fault Tolerance

With MicroProfile Fault Tolerance, we can build services which will work even when something failed. `@Timeout` annotation is used to avoid waiting forever for the response.

```
@Timeout(0)
@GET
public Response getAll() {
    return Response.ok(bookService.getAll()).build();
}
```

`@Fallback` annotation is used when something went wrong with the call then it will still operate without throwing an exception.

```
@Timeout(0)
@Fallback(fallbackMethod = "getAllFallbackMethod")
@GET
public Response getAll() {
    return Response.ok(bookService.getAll()).build();
}

public Response getAllFallbackMethod() {
    return Response.ok(Stream.of("Book One", "Book Two").collect(toList())).build();
}
```

`@Retry` annotation is used to repeat the call when something failed.

```
@Retry(maxRetries = 3, delay = 300)
@GET
public Response getAll() {
    return Response.ok(bookService.getAll()).build();
}
```

With `@Bulkhead` annotation you can limit the number of concurrent request that are made to the method

```
@GET
@Bulkhead(10)
public Response getAll() {
    return Response.ok(bookService.getAll()).build();
}
```

`@CircuitBreaker` annotation is used to immediately interrupt the call if the called services previously failed.

```
@GET
@CircuitBreaker(delay = 2000, requestVolumeThreshold = 2, failureRatio=0.65,
successThreshold = 3)
public Response getAll() {
    return Response.ok(bookService.getAll()).build();
}
```

Summary

In this chapter, we learned how to use MicroProfile Fault Tolerance.