


MicroProfile

microservices made easy



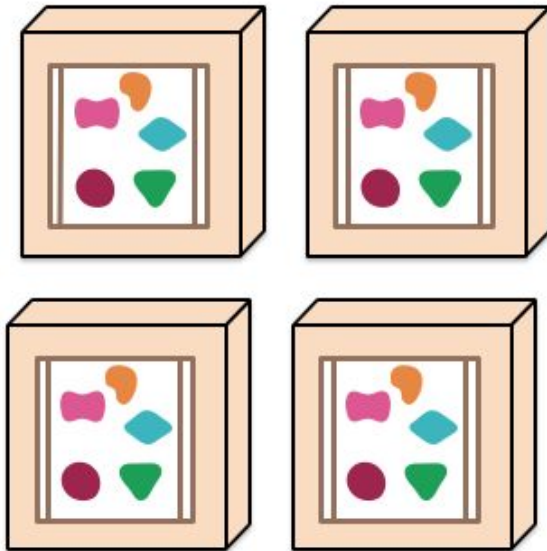
whoami

- Martin Štefanko
- Software engineer 3+ years, Red Hat
- MicroProfile contributor
-  @xstefank

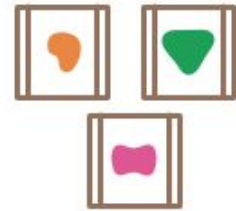
A monolithic application puts all its functionality into a single process...



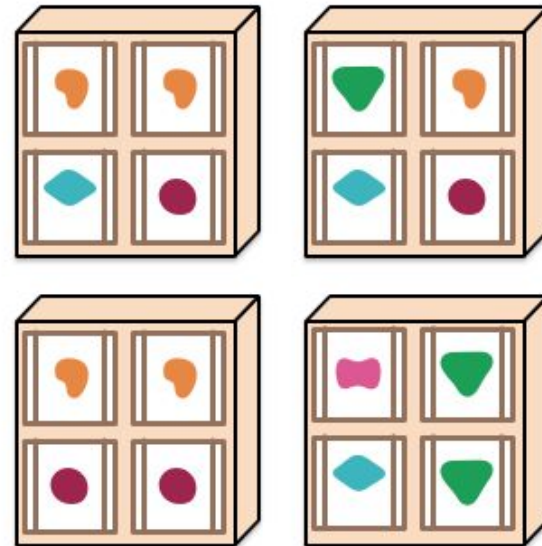
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



<https://martinfowler.com/articles/microservices.html>

Enterprise Java in past 20 years

- Java EE (currently Jakarta EE)
 - Java EE 5 – May 11, 2006
 - Java EE 6 – December 10, 2009
 - Java EE 7 – June 12, 2013
 - Java EE 8 – August 31, 2017

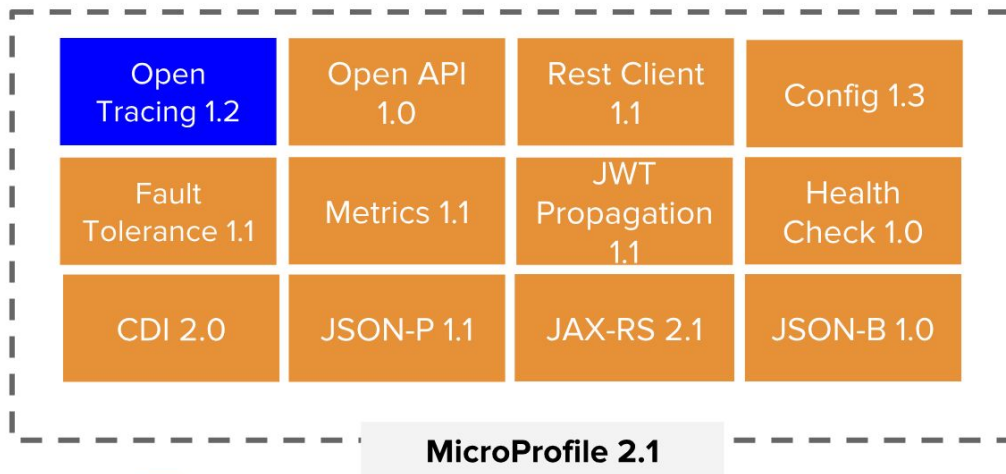



MicroProfile

- 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



Eclipse MicroProfile 2.1 (Oct, 2018)



 = New

 = Updated

 = No change from last release (MicroProfile 2.0)

+Under discussion

- Long Running Actions (LRA)
- Reactive Streams Operators
- Reactive messaging
- Service mesh
- Concurrency
- GraphQL
- ...

Community - individuals, organizations, vendors



Current MicroProfile implementations



Differences from Java EE









- open source and open community
- code first approach
- 3 releases per year (Feb, Jun, Oct)
 - MP 1.0 – Sep 2016
 - MP 1.1 – Aug 2017
 - MP 1.2 – Sep 2017
 - MP 1.3 – Jan 2018
 - MP 1.4 / MP 2.0 – Jun 2018
 - MP 2.1 – Oct 2018

MicroProfile 2.2

[New issue](#)

 Due by February 06, 2019 0% complete

The MicroProfile 2.2 release is targeted for Feb 2019 (with additional releases in June and October of 2019). We will use this Milestone to help track the content for this platform release. The expected Release Announce date will be Tuesday, Feb 12.

| | | |
|---|---|-----|
| 📄 8 Open ✓ 0 Closed | | |
| ☰ ⓘ Update MicroProfile spec for 2.2 release |  | |
| #77 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include Reactive Operators 1.0 |  | 💬 1 |
| #75 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include OpenTracing 1.3 |  | 💬 2 |
| #72 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include Metrics 2.0 |  | 💬 2 |
| #71 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include Rest Client 1.2 |  | 💬 2 |
| #70 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include Health Check 2.0 |  | |
| #69 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include OpenAPI 1.1 |  | 💬 4 |
| #68 opened on Nov 27, 2018 by kwsutter | | |
| ⓘ Include Fault Tolerance 2.0 |  | 💬 2 |
| #67 opened on Nov 27, 2018 by kwsutter | | |

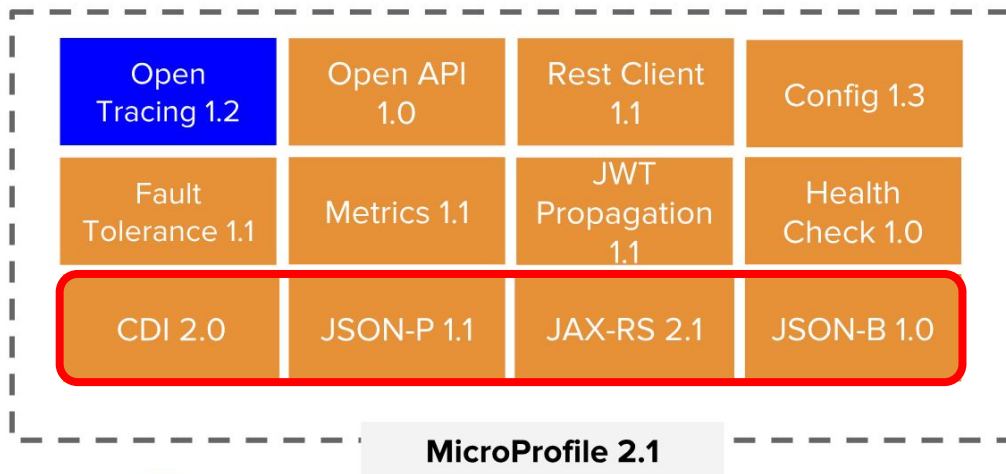
Eclipse MicroProfile

Optimizing Enterprise Java
for a Microservices Architecture



microprofile.io

Eclipse MicroProfile 2.1 (Oct, 2018)



■ = New

■ = Updated

■ = No change from last release (MicroProfile 2.0)

JAX-RS

```
@ApplicationPath("/api")  
public class ApplicationConfig extends Application {  
  
}
```

host:port/api/path1/path2/...

```

@Path("/")
@Consumes(MediaType.APPLICATION_JSON) @Produces(MediaType.APPLICATION_JSON)
public class ProfileService {

    @POST
    public Response logEvent(...) {
        return Response.accepted(event).build();
    }

    @GET
    @Path("user/{userId}")
    public Response getUserEvents(@PathParam("userId") int userId) {

        try {
            validateMembership(userId);
        } catch (NotFoundException nfe){
            return Response.status(Status.PRECONDITION_FAILED).header(REASON, "Membership [" + userId + "] does not exist").build();
        }
        return eventSearcher.search(UserEventConverter.USER_ID,userId,size);
    }

    ...
}

```

JSON-B

```
public class Membership implements Serializable {  
    private int membershipId;  
    private Person owner;  
    private Type type;  
}  
  
public class Person implements Serializable {  
    private int id;  
    private List<String> names;  
    private String surname;  
    private String email;  
}
```



```
{  
    "membershipId": 4,  
    "owner": {  
        "email": "minki@gmail.com",  
        "id": 4,  
        "names": [  
            "Minki"  
        ],  
        "surname": "van der Westhuizen"  
    },  
    "type": "FREE"  
}
```


JSON-B

```
Jsonb jsonb = JsonbBuilder.create();  
String json = jsonb.toJson(Avenger.name("Iron Man")  
    .realName("Tony Stark")  
    .alive(true)  
    .build());  
  
Avenger ironMan = jsonb.fromJson(json, Avenger.class);
```