
# Quarkus World Tour @Consol: OIDC

*Dr. Marco Bungart*

*Senior Software Engineer*

2021-07-01

# $> whoami

# Dr. Marco Bungart

- 2008 – 2013: Studium Bioinformatik/Informatik in Jena
- 2013 – 2018: Promotion in Kassel
- Seit 2018: Software Engineer at ConSol
- Twitter, github, bitbucket, stackoverflow, … : turing85
- Interessenschwerpunkte: Keycloak, GraalVM, Quarkus

# OIDC - Grundkurs

# OIDC

- “Alte Welt”: Session-State wird auf Server-seitig gespeichert
  - Containerisierung → Skaliert nicht naiv
  - Für Skalierung mehr Komponenten notwendig (z.B. redis)
- “Neue Welt”: Session-State auf Client-Seite speichern
  - Skaliert naiv
  - Ggf. Weitere Kommunikation nötig, um Session-State zu validieren

# OIDC

- State wird in JWT gespeichert
  - Enthält Informationen über die Identität
  - Wird von Authorization Server (z.B. keycloak) erstellt
  - Ist signiert → gesichert vor Änderung
- Backend-System erlaubt/verweigert Zugriff auf Ressource auf Grundlage des Tokens

# OIDC

- Nomenklatur:
  - Person: Benutzer, der die Anwendung nutzen möchte
  - Authorization Server: Service, der die Identität der Person validiert (für gewöhnlich über username/password-Abfrage & ggf. 2FA) und JWTs ausstellt, die die Identität bestätigen
  - Client: Technischer Akteur, der die JWTs konsumiert, dabei ggf. Anfragen an den Authorization Server stellt (z.B. zum Validieren von Tokens)

# JWTs

- Token ist Base64-codiertes JSON-Objekt
- Objekt enthält Claims, z.B.
  - “`iss`”: Issuer, meist URL zu Authorization Server
  - “`preferred_username`”: Benutzername
  - “`groups`”: Liste der dem Benutzer zugewiesenen Rollen

# OIDC in Quarkus

# OIDC in Quarkus

- Erweiterung: quarkus-oidc
- Konfiguration:

  quarkus.oidc.enabled
  quarkus.oidc.application-type (web-app, service, hybrid; wir benutzen service da wir ein reines Backend implementieren)
  quarkus.oidc.auth-server-url
  quarkus.oidc.client-id
  quarkus.oidc.introspection-path
  quarkus.oidc.jwks-path
  quarkus.oidc.token.audience
  quarkus.oidc.token.principal-name (welcher Claim ist der Benutzername)
  quarkus.oidc.credentials.secret (client-secret für Kommunikation mit OIDC-Server)

# OIDC in Quarkus

- Absicherung von Endpunkten durch:
  - `@Authenticated`: Endpunkt ist nur mit JWT aufrufubar
    - 401/Unauthorized, falls kein Token vorhanden
  - `@RolesAllowed(...)`: Endpunkt ist nur mit JWT aufrufubar und Benutzer muss eine der aufgelisteten Rollen haben
    - 401/Unauthorized, falls kein Token vorhanden
    - 403/Forbidden, falls Token vorhanden, aber keine passende Rolle
- Direkte Nutzung des Tokens:
  - `@Inject JsonWebToken jwt;` auf Feldebene, Service-Klasse kann trotzdem `@ApplicationScoped` sein

# Code!

# Fragen?

# Vielen Dank!


Dr. Marco Bungart
Senior Software Engineer

Büro Düsseldorf
Kanzlerstraße 8
D-40472 Düsseldorf
Tel.: +49-89-45841-100
Marco.Bungart@consol.de
www.consol.de
Twitter: @turing85