Ascent
FORMATION

Microservices with Quarkus

Day 2: Data Access & Security

Morning Session 1 (09:00 - 10:00)

Topic: Relational Databases with Panache

What is Panache?

- **Goal**: Simplify persistence with Hibernate ORM.
- **Two Patterns**:
  - **Active Record**: Entity class contains the persistence logic (`extends PanacheEntity`). Great for simpler models.
  - **Repository**: Separates persistence from repository class (`implements PanacheRepository`). Better for complex queries and separation of concerns.
- **Key Features**:
  - No more `EntityManager` boilerplate.
  - Simplified queries: `Person.find("name", "Scott")`
  - Automatic generation of CRUD operations.
- **Guides**:
  - Hibernate ORM with Panache

Morning Session 2 (10:45 - 11:45)

Topic: Implementing Persistence

Activity: Hands-on Lab 3

Lab 3 Objectives

- Persist `TrainStop` data in a relational database.
- Implement a full CRUD REST API for `TrainStop` entities.
- Use a Test-Driven Development (TDD) approach for implementation.
- Leverage Quarkus Dev Services for a zero-config development database.

Morning Session 3 (12:00 - 12:30)

Topic: Exposing and Documenting REST Endpoints

Jakarta RESTful Web Services

- The standard for creating RESTful web services in Java.
- **Core Concepts**:
  - **Annotations**: `@Path`, `@GET`, `@POST`, `@Produces`, `@Consumes`.
  - **Injection**: Use `@Inject` to bring in other components.
  - **Path Parameters**: Use `@PathParam` to capture parts of the URL.
- **Guide**: RESTEasy Reactive

API Documentation with OpenAPI

- **Standard**: OpenAPI is the industry standard for defining REST APIs.
- **Automatic Generation**: Quarkus automatically generates an `openapi.yaml` file for your JAX-RS endpoints.
- **Swagger UI**: A user-friendly interface to explore and test your API.
  - Accessible at `/q/swagger-ui` in dev mode.
- **Guide**: OpenAPI and Swagger UI

Ascent
FORMATION

Lunch Break (12:30 - 13:15)

Afternoon Session 1 (13:15 - 14:15)

Topic: Documenting Persisted Data

Activity: Hands-on Lab 4

Lab 4 Objectives

- Automatically generate API documentation from JAX-RS endpoints.
- Enrich the generated documentation with descriptive annotations.
- Implement an idempotent `create` endpoint for `TrainStopResource`.
- Explore and test the API using the integrated Swagger UI.

Afternoon Session 2 (14:30 - 15:00)

Topic: REST Clients & Security Concepts

MicroProfile REST Client

- **Goal**: Create a type-safe client to consume other RESTful services.
- **How it works**:
  1. Define a Java interface with JAX-RS annotations.
  2. Annotate it with `@RegisterRestClient`.
  3. Inject it with `@RestClient`.
- **Configuration**: Set the base URL in `application.properties`.
- **Guide**: REST Client Reactive

Introduction to Security

- **Core Concepts**:
  - **OAuth2**: An authorization framework for granting access to resources.
  - **OpenID Connect (OIDC)**: An identity layer built on top of OAuth2. It provides authentication and user information.
  - **JWT (JSON Web Token)**: A compact, URL-safe means of representing claims to be transferred between two parties.
- **Identity Provider**: A trusted provider that manages user identity and authentication (e.g., Keycloak, Auth0, Okta).
- **Guides**:
  - OpenID Connect (OIDC) Bearer Token Authentication
  - Using JWT RBAC

Afternoon Session 3 (15:00 - 16:00)

Topic: Securing and Consuming a Protected API

Activity: Hands-on Lab 5

Lab 5 Objectives

- Secure a REST endpoint using role-based access control (RBAC).
- Configure the microservice to authenticate with a Keycloak server.
- Consume a protected, external API using a type-safe REST client.
- Enrich service data with information from an external service.

End of Day 2

- Recap & Q&A
- Preview of Day 3: Reactive Messaging & Monitoring