Who am I?

Adrien Baudhuin

mail: adrien+uni[at]baudhuin.fr

Software Engineer at **AODocs**

All resources are available on **GitHub**

What are we going to learn?

- What is Spring
- Create a Spring Boot app
- Create REST API with Spring
- In next class: Connection to a database

Pre-requisites for this class

- Java
- Maven
- HTTP (Documentation)
- REST API (Documentation)

How to Spring

First came JEE (Java Enterprise Edition)

- Set of standards to build enterprise applications.
- A lot of libraries to solve common problems.

Quite old (1999) and not much used anymore.

Then came Spring

- A web framework to build web applications.
- Follows most of the JEE standards.
- Fixes modularity issues of JEE.
- Used everywhere (Netflix, Amazon, etc.)

More recently

New frameworks, specialized in different areas (microservices, serverless, etc.)

Most notable in the Java ecosystem:

- Micronaut
- Quarkus

Spring Web Framework

- A standardized structure.
- Highly configurable and customizable
- A set of libraries to solve common problems
    - Spring Web MVC (Create HTTP APIs)
    - Spring Data (Connect to Databases)
    - Spring Security (Secure and authenticate users)
    - Spring Cloud (Connect to Cloud Providers)

Spring Boot

- Spring is complex and hard to setup because of its flexibility.
- Spring boot is a preconfigured Spring application
- Easy to start a new project

Creating a Spring Boot app

Spring Initializer

Multi-Layer Architecture (Wikipedia)

> A multi-tier architecture is a client–server architecture in which presentation, application processing, and data management functions are physically separated.

Often, we have 3 layers:

- Controllers (Presentation)
- Services (Business logic)
- Repositories (Data)

Inversion of Control (Wikipedia)

> The application is in charge of creating all the objects it needs.

Dependency Injection (Wikipedia)

> The application injects the dependencies each component needs.

Aspect Oriented Programming (Wikipedia)

> Allows to add cross-cutting behaviours to an application without modifying the code.

First HTTP endpoint

- Create a controller
- Create a endpoint method
- Annotations
- Serialization

First service

- Create a service
- Adding logic
- Inject in the controller

Testing

- Unit testing Sercices
- Integration testing Controllers

Production

Lab 1

- Create a TODO app
- Multiple endpoints CRUD
- A service with business logic
- Tests

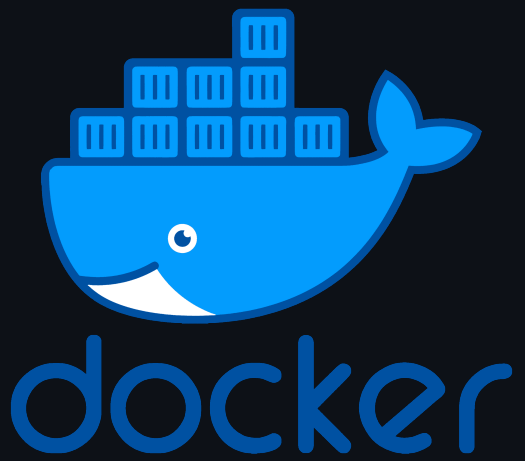
Questions from the previous lab ?

Today's topics

- Dockerize a Spring Boot app
- Spring Data JPA & Database

Docker

- A containerization platform
- Lightweight virtualization
- Containers are isolated from the host
- Backbone of microservices and serverless



Docker image

- A template to create a container
- Contains the application and its dependencies
- Can be run on any machine
- Docker hub is a registry of images (like GitHub for code)

Dockerfile

- A file to build a Docker image
- Contains instructions to build the image

```
# Use the official image as a parent image.
FROM openjdk:8-jdk-alpine

# Copy useful files from your host to your image filesystem.
COPY target/*.jar app.jar

# Command to run the executable when the container starts.
ENTRYPOINT ["java","-jar","/app.jar"]
```

Running a Docker container

- We can run any images: `docker run mysql`
- To build an image from a Dockerfile: `docker build -t my-image .`
- We can run our own images: `docker run my-image`

Spring Data JPA

- An ORM library based on Hibernate
- Connects to a lot of databases (MySQL, PostgreSQL, NoSQL, etc.)
- A set of annotations to map Java objects to database tables
- Simple interface to create queries

In practice: Entity

```
@Entity
public class MyObject {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;

  private String name;
}
```

In practice: CrudRepository

```
@Repository
public interface MyRepository extends CrudRepository<MyObject, Long> {
  @Query("SELECT o FROM MyObject o WHERE o.name = :name")
  List<MyObject> findByName(@Param("name") String name);
}
```

By default, Spring Data JPA will already implements these queries:

- Find all
- Find by id
- Save
- Delete

In practice: JpaRepository

```
@Repository
public interface MyRepository extends JpaRepository<MyObject, Long> {
  List<MyObject> findByName(String name);
}
```

Based on the name of the method, Spring Data JPA will automatically generate the query!

Let's do it

- Add the dependency
- Configure the database
- Create a model
- Create a repository

Advanced feature: Pagination

- We don't want to return all the objects in the database
- We need to do `/my-objects?page=0&size=10`

In Spring Data JPA, the `Pageable` interface handles this for us.

```
Page<MyObject> findAll(Pageable pageable);
```

Testing

- We want to test the service, not the database
- We can use in-memory databases
- We can use mocks to test the service in isolation

Mocking

By default, Spring uses the Mockito library to mock dependencies.

```
class MyServiceTest {
  @MockBean
  private MyRepository repository;

  @Autowired
  private MyService service;

  @Test
  void test() {
    MyObject object = new MyObject();
    when(repository.findByName("name")).thenReturn(List.of(object));

    List<MyObject> result = service.findByName("name");

    assertThat(result).containsExactly(object);
  }
}
```

Lab 2

- Dockerize the TODO app
- Use Mysql with Spring Data JPA for your TODO app
- Connected four with a database !