Docker

By Eng: Abdelrahman Ayman Lotfy

DevOps track

Table of contents

Spring pet clinic

What is Spring PetClinic?

Spring PetClinic is a **sample application** created by the Spring team to demonstrate how to build a real-world web application using the **Spring Framework**.

It simulates a small veterinary clinic management system where you can manage:

- **Owners ➔** people who own pets.
- **Pets ➔** animals with details like type and birth date.
- **Vets ➔** veterinarians who work in the clinic.
- **Visits ➔** medical appointments for pets.

Main Technologies Used

- **Spring Boot ➔** for running the application as a standalone service.
- **Spring MVC ➔** for handling web requests (controllers, routing, etc.).
- **Spring Data JPA ➔** for database access and repository management.
- **Thymeleaf ➔** as the template engine for rendering web pages.
- **H2 Database** (by default) **➔** an in-memory database, but you can switch to MySQL/PostgreSQL.

Project Structure

- model/ ➔ domain classes (Owner, Pet, Vet, Visit).
- repository/ ➔ interfaces that handle database operations.

- service/ ➔ contains business logic (often thin, thanks to Spring Data JPA).
- controller/ ➔ Spring MVC controllers that handle web requests.
- resources/templates/ ➔ Thymeleaf HTML pages.
- application.properties ➔ configuration file.

Key Features

- Search for owners and view their pets.
- Add new owners and pets.
- View and manage veterinarians.
- Record visits for pets.

How to Run It

```
git clone https://github.com/spring-projects/spring-petclinic.git
cd spring-petclinic
./mvnw package
java -jar target/*.jar
```

You can write localhost:8080 to see the application

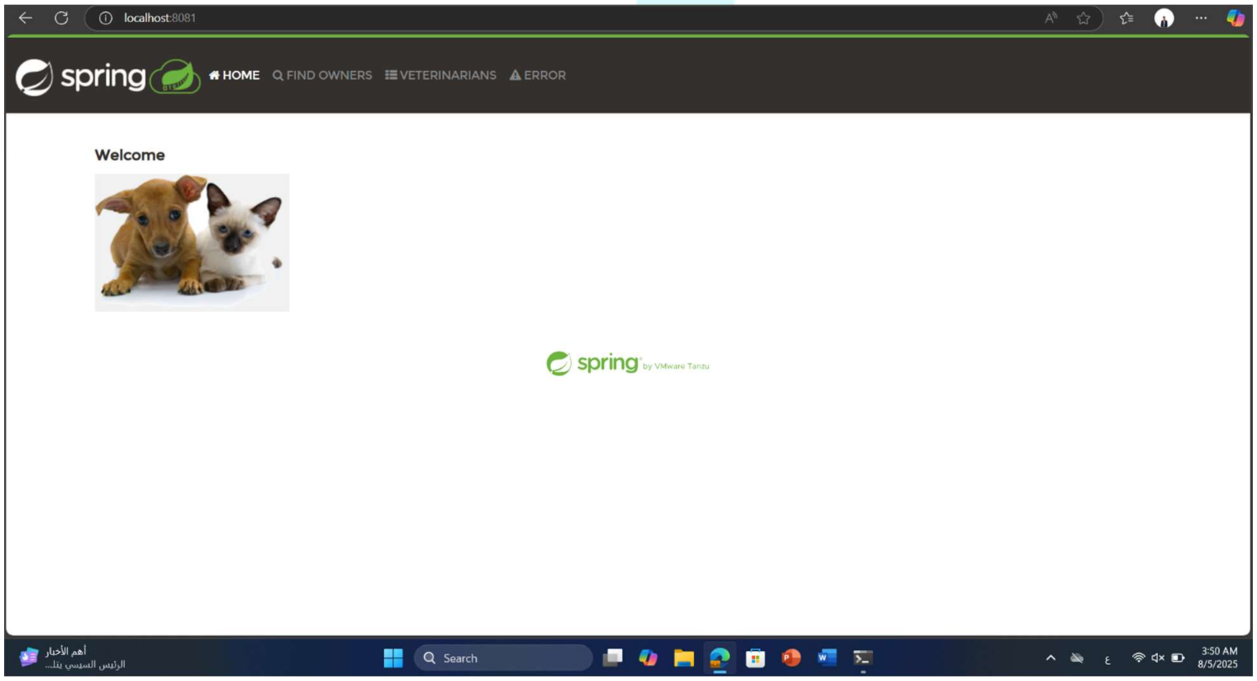
Method two pull an existing image

docker pull {image name and its tag} get it from dockerhub

docker run -d -P {image} ➔ will run it on a random port

docker run -d -p 8081:8080 {image} ➔will run it on 8081 port

You will see:



Now we need to make a dockerfile for it

1. **Goal** ➔ Production-ready, small, secure, and efficient.
2. **Base Image** ➔ Use a JDK only for building, then a JRE (or distroless) for running.
3. **Layers** ➔ Multi-stage build (dependencies → build → runtime).
4. **Security** ➔ Run as non-root, avoid unnecessary files.

5. **Entrypoint** ➔ Run the jar with java -jar.

And my final Dockerfile

```
# Build stage

FROM maven:3.9.9-eclipse-temurin-17 AS build

WORKDIR /app

#to improve caching

COPY pom.xml .

# Cache dependencies

RUN mvn dependency:go-offline

COPY src ./src

RUN mvn clean package -DskipTests


# Run stage with jre not jdk

FROM eclipse-temurin:17-jre-alpine

WORKDIR /app

COPY --from=build /app/target/*.jar app.jar


EXPOSE 9966


ENTRYPOINT ["java", "-jar", "app.jar"]
```

Breakdown of Best Practices Used

- **Multi-stage build ➔** Maven image for build, slim JRE for runtime ➔ smaller, cleaner final image.
- **Dependency caching ➔** COPY pom.xml + mvn dependency:go-offline before copying src ➔ faster rebuilds.
- **Security ➔** non-root user.
- **ENTRYPOINT ➔** Entrypoint is fixed (java -jar app.jar) .

Note

- I modified the default port of the application from 8080 to 9966 by editing the application.yml file and add server.port=9966
- Also to limit the size of the image I used dockerignore inside it I put

```
.git
.gitignore
target
*.md
.idea
*.iml
src/test
```

Building & Runnig

```
Docker build -t petclinic:v1 .
Docker run-d -p 9999:9966 petclinic:v1
```

Conclusion

Spring PetClinic is a simple yet practical sample application that demonstrates how to build a real-world web app with Spring Boot, Spring MVC, and Spring Data JPA. It's widely used for learning, training, and experimenting with modern software development practices such as CI/CD, containerization, and cloud deployment.