Critter Chronologer - Developer Setup Guide

1. Configure Datasources

Edit the file: src/main/resources/application.properties

Add the following configuration to connect to your MySQL database:

spring.datasource.url=jdbc:mysql://localhost:3306/critter

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.defer-datasource-initialization=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

server.port=8082

Then create the file: src/test/resources/application.properties and add H2 configuration:

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=create-drop

2. Create Entity Classes

File: Customer.java (in user package)

- @Entity, Long id, String name, String phoneNumber, String notes
- OneToMany List<Pet> pets (mappedBy = "owner")

File: Employee.java

- @Entity, Long id, String name
- EnumSet<EmployeeSkill> skills, Set<DayOfWeek> daysAvailable with @ElementCollection

File: Pet.java (in pet package)

- @Entity, Long id, PetType type, String name, LocalDate birthDate, String notes
- ManyToOne Customer owner

File: Schedule.java (in schedule package)

- @Entity, Long id, ManyToMany List<Employee> employees, List<Pet> pets
- LocalDate date, Set<EmployeeSkill> activities

3. Setup DTO Classes

Ensure these DTOs exist in your project:

- CustomerDTO: id, name, phoneNumber, notes, List<Long> petIds
- EmployeeDTO: id, name, Set<EmployeeSkill> skills, Set<DayOfWeek> daysAvailable
- EmployeeRequestDTO: Set<EmployeeSkill> skills, LocalDate date
- PetDTO: id, PetType type, name, birthDate, notes, Long ownerId
- ScheduleDTO: id, List<Long> employeeIds, List<Long> petIds, LocalDate date, Set<EmployeeSkill> activities

4. Create Repository Interfaces

File: CustomerRepository.java extends JpaRepository<Customer, Long>

File: EmployeeRepository.java extends JpaRepository<Employee, Long>

File: PetRepository.java extends JpaRepository<Pet, Long>

- Add: List<Pet> findAllByOwnerId(Long ownerId);

File: ScheduleRepository.java extends JpaRepository<Schedule, Long>

- Add: findAllByPetsContains, findAllByEmployeesContains

5. Implement Service Classes

File: UserService.java

- saveCustomer(Customer), getAllCustomers(), getOwnerByPetId(long)

- addNewEmployee(Employee), getEmployeeById(Long), updateAvailability(...), findMatchingEmployees(...)

File: PetService.java

- savePet(Pet, ownerId), getPetById(Long), getPetsByOwner(Long), getAllPets()

File: ScheduleService.java

- createSchedule(Schedule, employeeIds, petIds), getAllSchedules()
- getScheduleForPet(...), getScheduleForEmployee(...), getScheduleForCustomer(...)

6. Implement Controllers

Manually map DTOs to Entities and vice versa.

Files: UserController.java, PetController.java, ScheduleController.java

- Convert DTO -> Entity before service call
- Convert Entity -> DTO for responses

Avoid using static converter classes to stay aligned with Udacity's official repo.

7. Run Tests

Open: src/test/java/com/udacity/jdnd/course3/critter/CritterFunctionalTest.java

Run all 9 unit tests. All must pass:

- testCreateCustomer
- testCreateEmployee
- testAddPetsToCustomer
- testFindPetsByOwner
- testFindOwnerByPet
- testChangeEmployeeAvailability
- testFindEmployeesByServiceAndTime
- testSchedulePetsForServiceWithEmployee
- testFindScheduleByEntities

Critter Chronologer - Developer Setup Guide

8. Verify with Postman

Import the Postman collection from:

src/main/resources/Udacity.postman_collection.json

Check routes like:

- POST /user/customer

- POST /pet

- POST /schedule

- GET /schedule/pet/{id}, /employee/{id}, /customer/{id}

Ensure all return 200 OK and valid responses