Entity Mapping and Persistence in JPA

Introduction

Java Persistence API (JPA) is a standard for object-relational mapping (ORM) in Java. It allows developers to map Java
objects to database tables and manage relational data in Java applications. This document provides an overview of
entity mapping and persistence in JPA.

Entity Mapping

Entity mapping is the process of associating a Java class with a database table. Each instance of the class corresponds
to a row in the table, and each field in the class corresponds to a column in the table. The @Entity annotation is used
to mark a class as an entity, making it eligible for mapping.

Example:

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

    private String surname;
    private String firstName;
```

```
    private String address;
    private String phoneNumber;
}
```

Persistence

Persistence is the process of storing and retrieving data from a database. In JPA, an entity manager is used to perform
CRUD operations on entities. The entity manager interacts with the database through the persistence context.

Example:

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

    private String departmentCode;
    private String departmentName;
    private String building;

    @OneToOne
    private Doctor director;
}
```

Entity Relationships

JPA allows the mapping of relationships between entities using annotations like @OneToOne, @OneToMany, @ManyToOne, and @ManyToMany.

These annotations define how entities are related in the database.

Example:

```
@OneToMany(mappedBy = "department")
private List<Nurse> nurses;
```

This maps a one-to-many relationship between the Department entity and the Nurse entity.