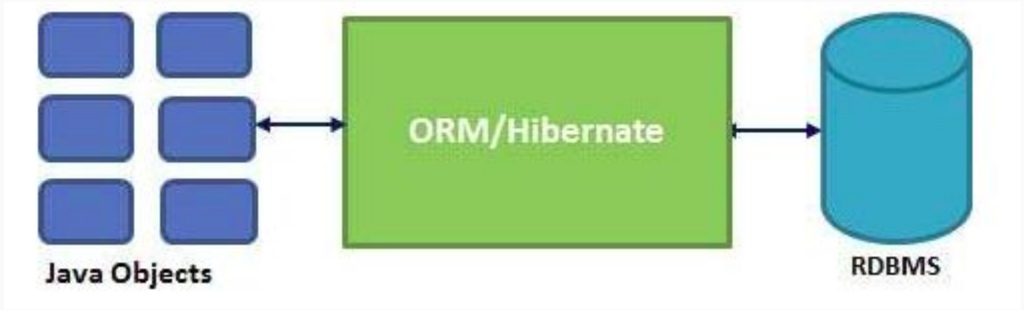
Hibernate

Hibernate is an Object-Relational Mapping (ORM) solution for JAVA

What is ORM?

Object-Relational Mapping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages.



Why Object Relational Mapping (ORM)?

- Mismatch between the object model and the relational database.
- Databases are designed with Tables/Relations
- Java objects are designed using OOPS.
- We would want to store the data from objects into tables and vice-versa.
- Earlier approaches involved writing SQL Queries : JDBC
- How about mapping the objects directly to tables/relationships?

Object-Relational Mapping

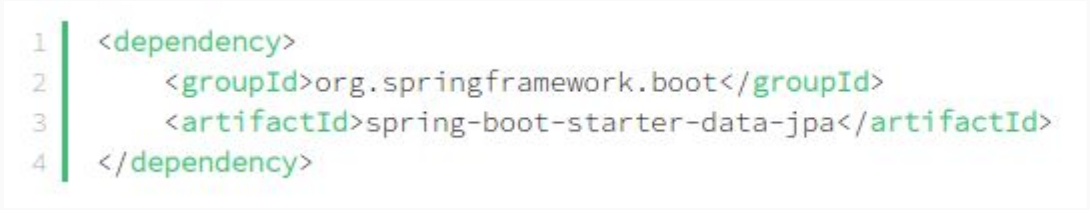
- Let’s business code access objects rather than DB tables.
- Hides details of SQL queries from OO logic.
- Entities based on business concepts rather than database structure.

Hibernate

- Hibernate is one of the popular implementations of JPA.
- Hibernate understands the mappings that we add between objects and tables
- It ensures that data is stored/retrieved from the database based on the mapping.

Spring Boot with Hibernate

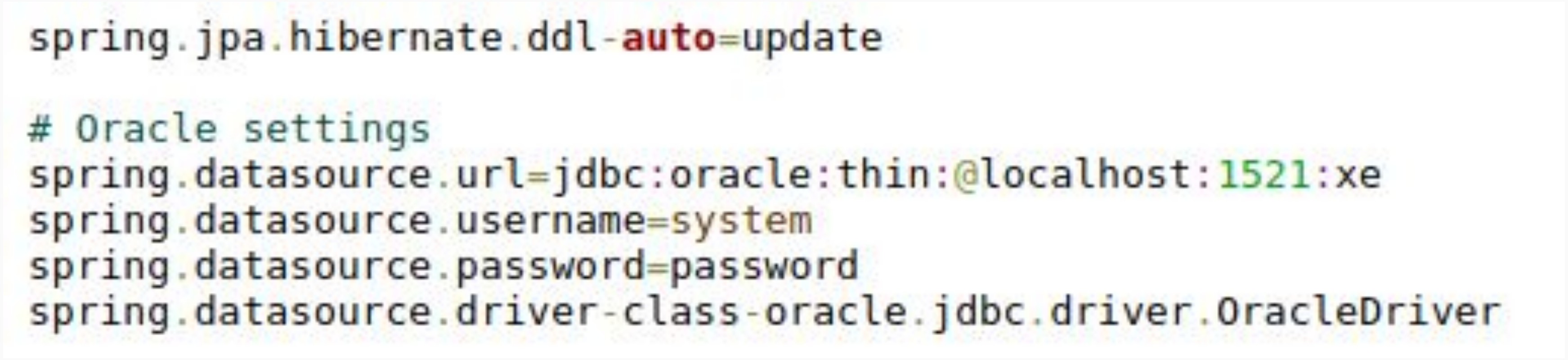
1. **Add *spring-boot-starter-data-jpa* as maven dependencies in pom.xml**



```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

- This dependency includes JPA API, JPA Implementation, JDBC and other needed libraries.
- Default JPA implementation is Hibernate

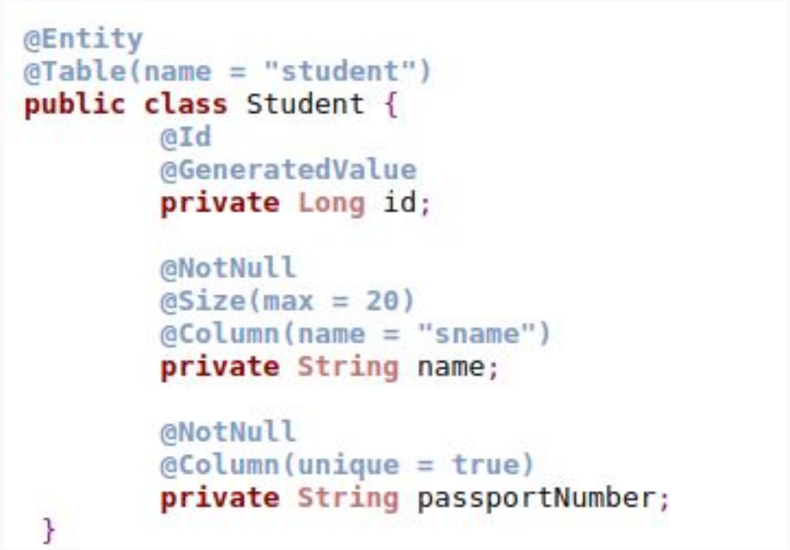
2. Add db connection details in application.properties



```
spring.jpa.hibernate.ddl-auto=update

# Oracle settings
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=system
spring.datasource.password=password
spring.datasource.driver-class-oracle.jdbc.driver.OracleDriver
```

3. Creating the Entity

- Entities are POJOs representing data that can be persisted to the database.
- An entity represents a table stored in a database.
- Every instance of an entity represents a row in the table.

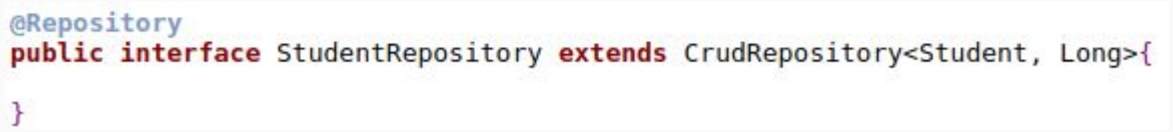


```
@Entity
@Table(name = "student")
public class Student {
        @Id
        @GeneratedValue
        private Long id;

        @NotNull
        @Size(max = 20)
        @Column(name = "sname")
        private String name;

        @NotNull
        @Column(unique = true)
        private String passportNumber;
 }
```

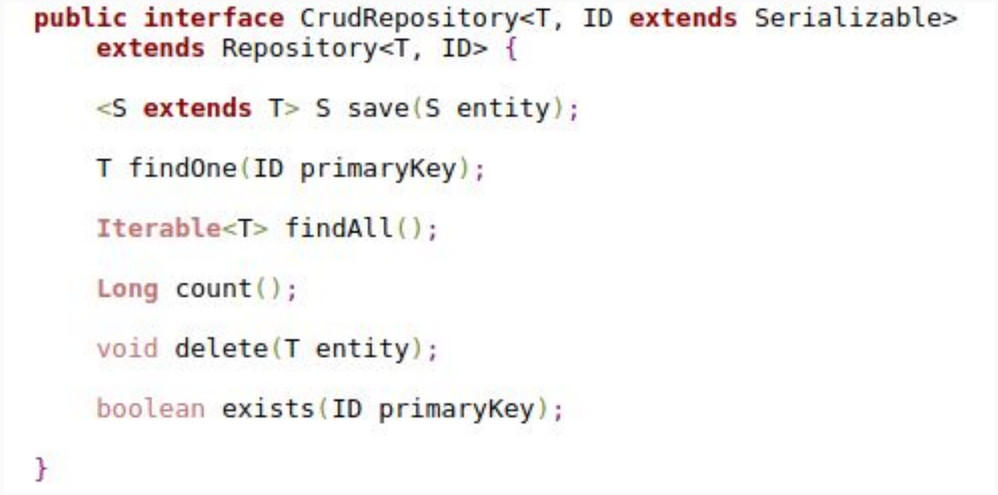
- @Entity - specifies to declare the class as entity or a table.
- @Table - specifies to declare table name
- @Id - specifies to use for identity (primary key of a table) of the class.
- @GeneratedValue - specifies, how the identity attribute can be initialized such as Automatic, manual, or value taken from sequence table.
- @Column - specify column or attribute for persistence property.
- @Transient - specifies the property which in not persistent

4. Create Repository class to Read Student information

- @Repository annotation is used to indicate that the class provides the mechanism for storage, retrieval, search, update and delete operation on objects.

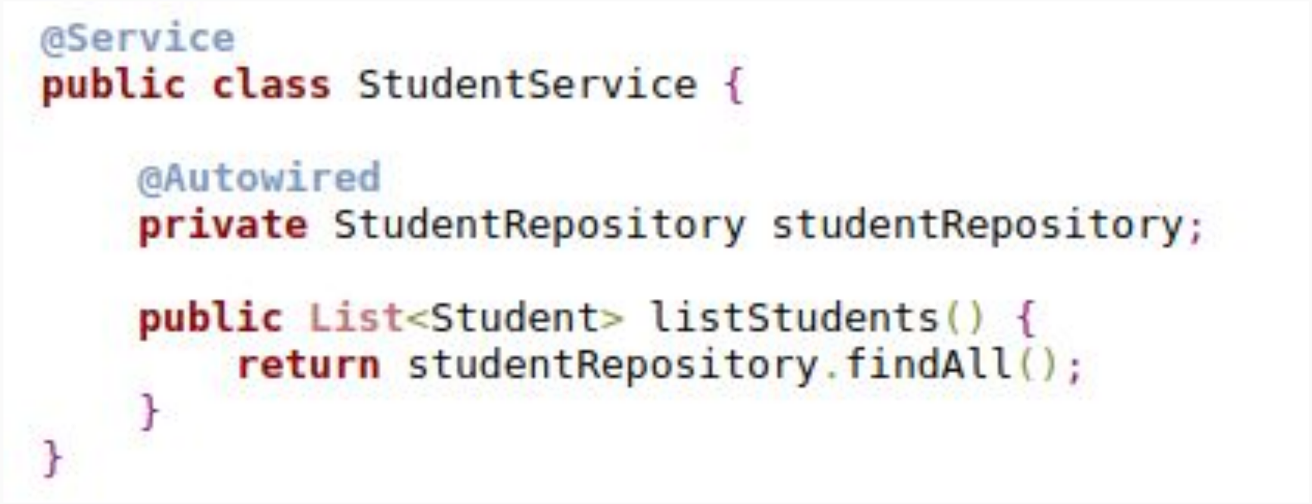


```
@Repository
public interface StudentRepository extends CrudRepository<Student, Long>{

}
```



```
public interface CrudRepository<T, ID extends Serializable>
    extends Repository<T, ID> {

    <S extends T> S save(S entity);

    T findOne(ID primaryKey);

    Iterable<T> findAll();

    Long count();

    void delete(T entity);

    boolean exists(ID primaryKey);

}
```

5. Service to call repository methods

- 

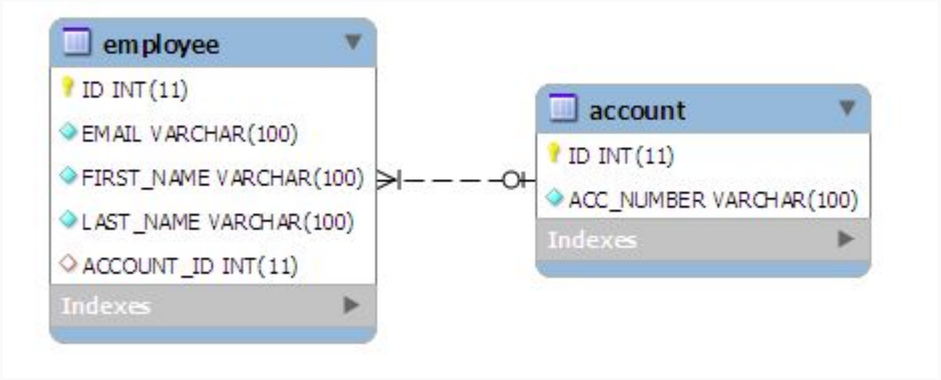
```java
@Service
public class StudentService {

    @Autowired
    private StudentRepository studentRepository;

    public List<Student> listStudents() {
        return studentRepository.findAll();
    }
}
```

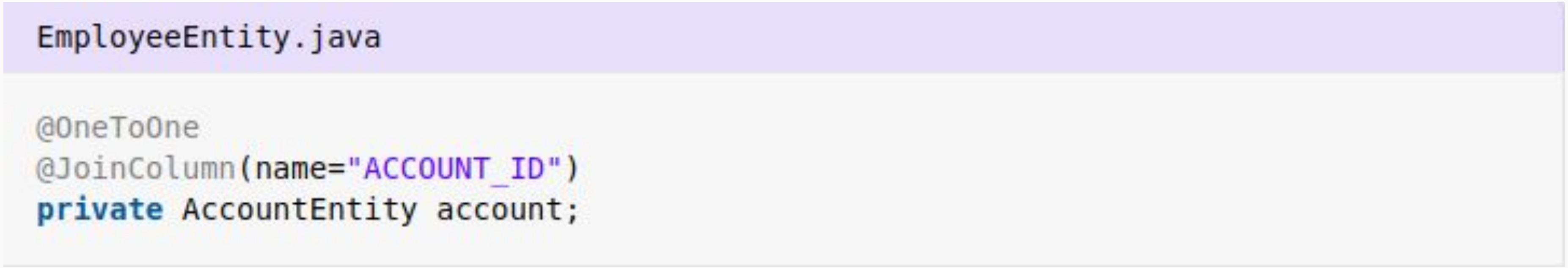
JPA - Entity Relationships

- @OneToOne Relation
- @OneToMany Relation
- @ManyToOne Relation
- @ManyToMany Relation

OneToOne Relation

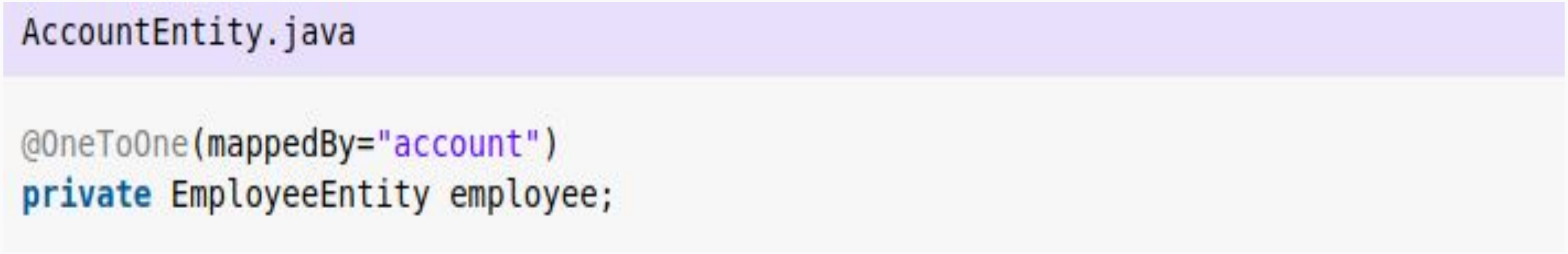
- It means each row of one entity is referred to one and only one row of another entity.





EmployeeEntity.java

```java
@OneToOne
@JoinColumn(name="ACCOUNT_ID")
private AccountEntity account;
```



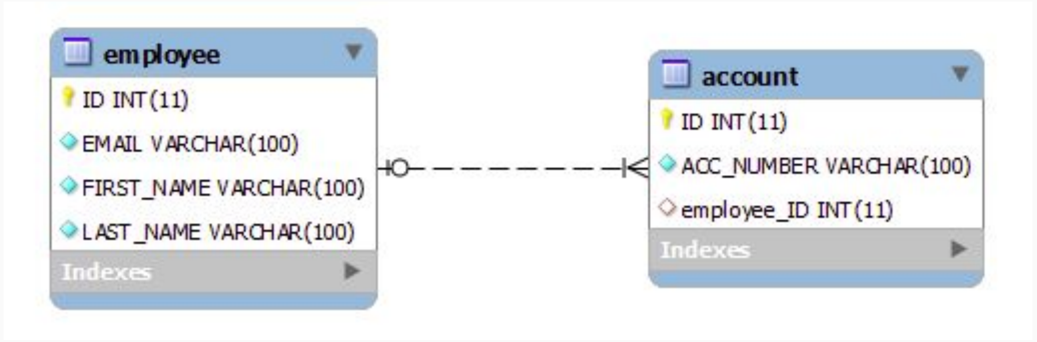
AccountEntity.java

```java
@OneToOne(mappedBy="account")
private EmployeeEntity employee;
```

- “mappedBy” attribute declares that it is dependent on owner entity for mapping

OneToMany Relation

- Hibernate one to many mapping is made between two entities where first entity can have relation with multiple second entity instances but second can be associated with only one instance of first entity.



Employee Entity

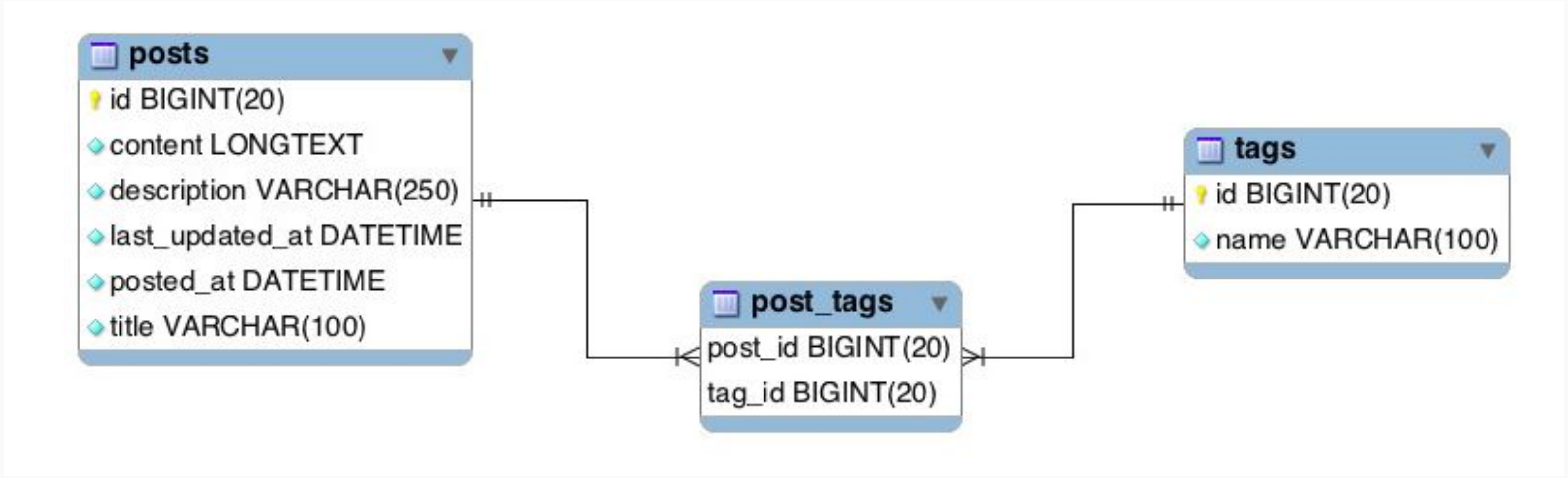
```
@OneToMany(cascade=CascadeType.ALL)
@JoinColumn(name="EMPLOYEE_ID")
private Set<AccountEntity> accounts;
```

Account Entity

```
@ManyToOne
private EmployeeEntity employee;
```

ManyToMany Relation

Many-To-Many relationship is where one or more rows from one entity are associated with more than one row in other entity.



Post Entity

```
@ManyToMany(fetch = FetchType.LAZY,
        cascade = {
            CascadeType.PERSIST,
            CascadeType.MERGE
        })
@JoinTable(name = "post_tags",
        joinColumns = { @JoinColumn(name = "post_id") },
        inverseJoinColumns = { @JoinColumn(name = "tag_id") })
private Set<Tag> tags = new HashSet<>();
```

Tag Entity

-

```
@ManyToMany(fetch = FetchType.LAZY,
        cascade = {
            CascadeType.PERSIST,
            CascadeType.MERGE
        },
        mappedBy = "tags")
private Set<Post> posts = new HashSet<>();
```