Ciências ULisboa Faculdade de Ciências da Universidade de Lisboa

2019/2020

Construção de Sistemas de Software

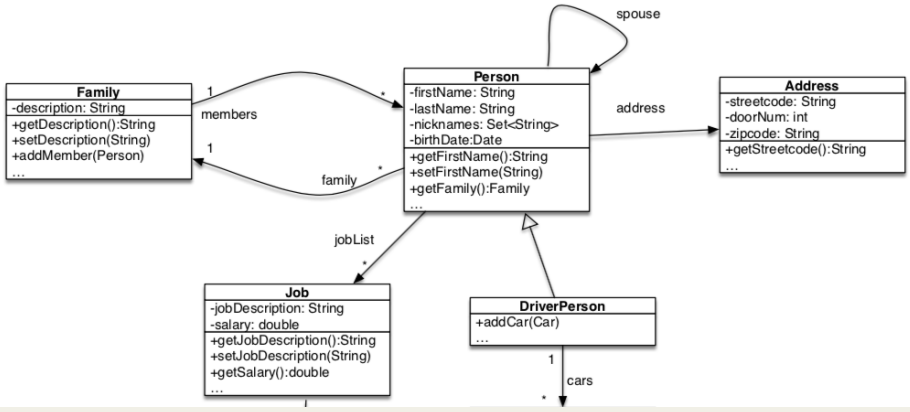
JPA — The Java Persistence API
ORM Structural Patterns

Ciências ULisboa | Informática

CSS (2019/2020)
Caderno de Exercícios
Folha 5

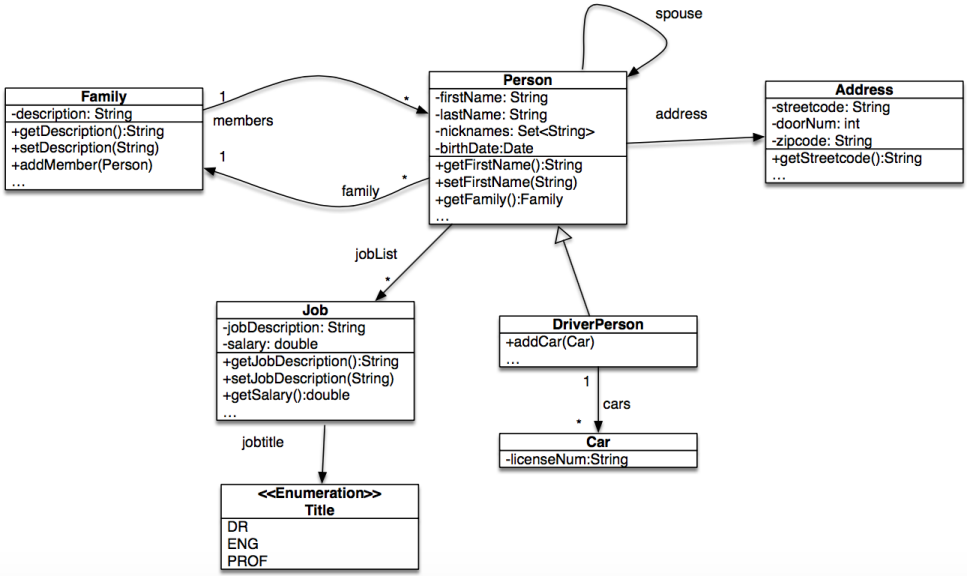
JPA Mapper (ORM Structural Patterns)

1. Considere o diagrama de classes abaixo, que faz parte da camada de negócio de uma aplicação implementada recorrendo ao padrão *Domain Model*. Recorrendo ao JPA, defina um mapeamento deste Modelo de Objetos num Modelo Relacional e indique em que Modelo Relacional é feito o mapeamento.

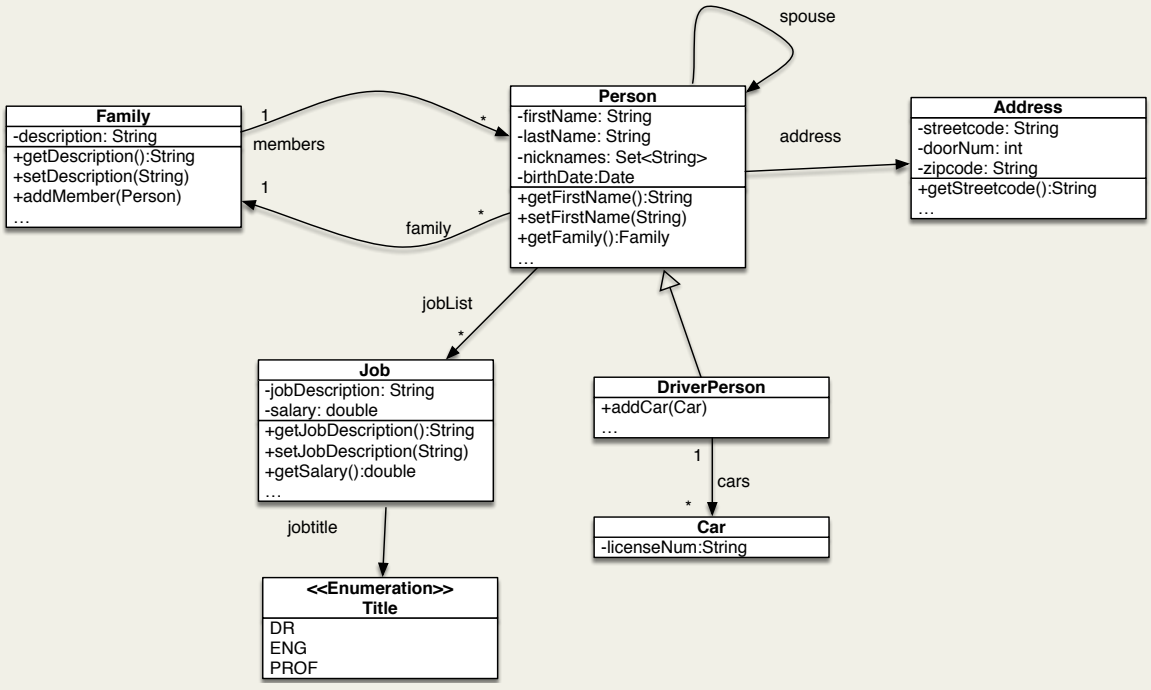


Exercício

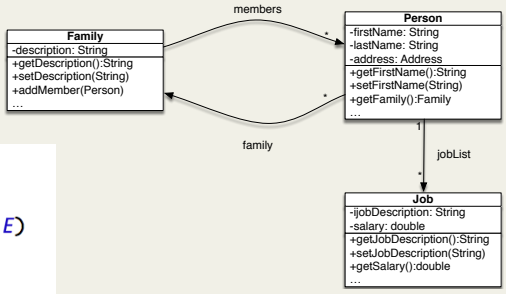
Considere o diagrama de classes abaixo, que faz parte da camada de negócio de uma aplicação implementada recorrendo ao padrão *Domain Model*. Recorrendo ao JPA, defina um mapeamento deste Modelo de Objetos num Modelo Relacional e indique em que Modelo Relacional é feito o mapeamento.



ORM ?



Entidades, Chaves e Propriedades de Tipo Básico




```
@Entity
public class Family {
    @Id @GeneratedValue(strategy = GenerationType.TABLE)
    private int id;
    private String description;
```

```
@Entity
public class Person {

    @Id @GeneratedValue
    private int id;
    private String firstName;
    private String lastName;
```

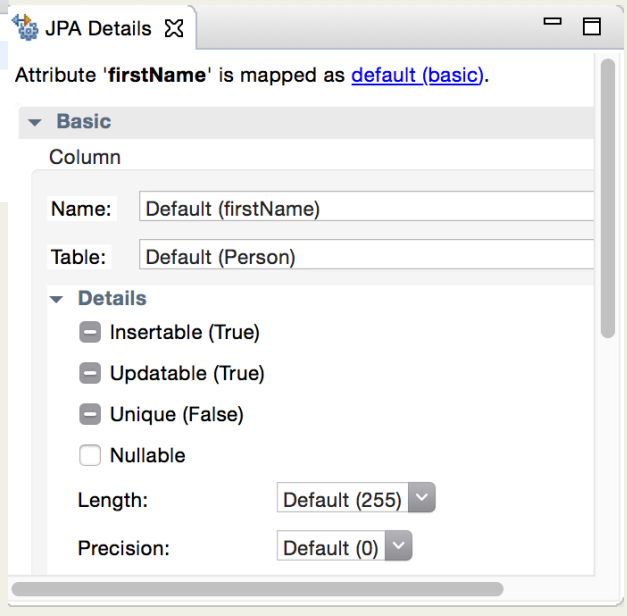
```
@Entity
public class Job {

    @Id @GeneratedValue
    private int idJob;
    private double salary;
    private String jobDescr;
```

Ciências ULisboa | Informática

Entidades, Chaves e Propriedades de Tipo Básico

Person.java

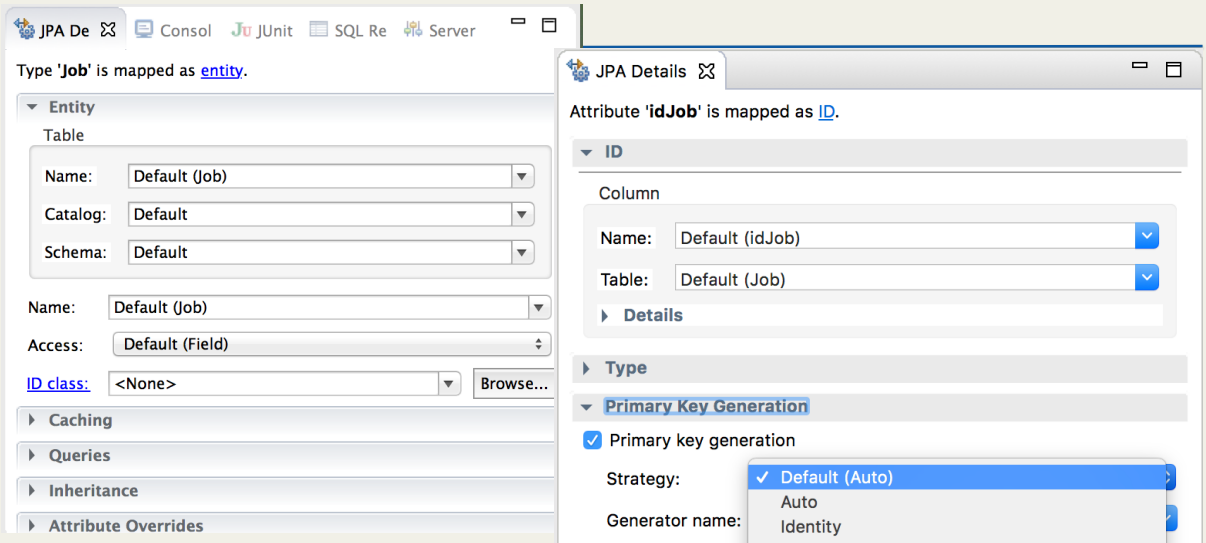
```
65
66      private String firstName;
67      private String lastName;
68
69      @Embedded
70      private Address address;
71
```



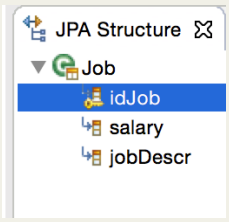

```
@Column(nullable = false)
private String firstName;
private String lastName;
```

JPA Provider: Eclipse Link

Ciências ULisboa | Informática




```java
@Entity
public class Job {

    @Id @GeneratedValue
    private int idJob;
    private double salary;
    private String jobDescr;
```

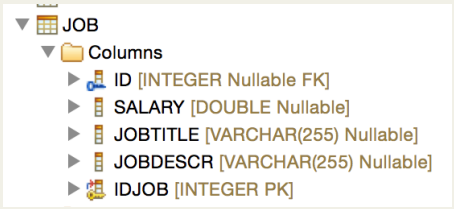


Ciências ULisboa | Informática

Propriedades de Tipo Data e Enumerado

```java
@Entity
public class Job {
    @Id
    @GeneratedValue
    private int idJob;

    private double salary;

    private String jobDescr;

    @Enumerated(EnumType.STRING)
    private Title jobTitle;
```

```java
@Entity
@Inheritance(strategy=SINGLE_TABLE)
public class Person {
    @Id @GeneratedValue(strategy = GenerationType.TABLE)
    private int id;
    private String firstName;
    private String lastName;

    @Temporal(TemporalType.DATE)
    private Date birthDate;
```



Ciências ULisboa | Informática

Mais sobre mapeamento de datas

- Mapeamento de datas do *Date and Time API do Java 8*
  - *o JPA 2.1 saiu antes do Java 8, pelo que o @Temporal não pode ser usado*

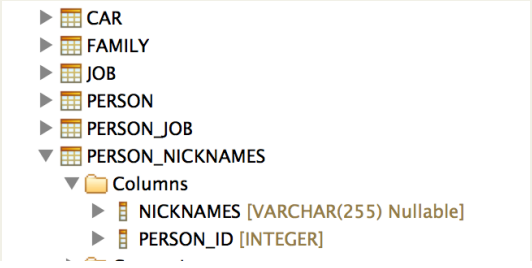
```
@Temporal(TemporalType.DATE)
private Date birthDate;

@Column
@Convert(converter = LocalDateAttributeConverter.class) //not required
private LocalDate alternativeBirthDate;
```

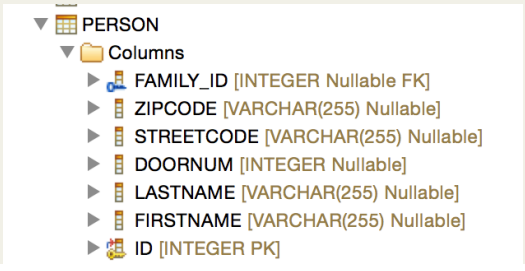
```
import java.sql.Date;
import java.time.LocalDate;

import javax.persistence.AttributeConverter;
import javax.persistence.Converter;

@Converter(autoApply = true)
public class LocalDateAttributeConverter implements AttributeConverter<LocalDate, Date> {

    @Override
    public Date convertToDatabaseColumn(LocalDate locDate) {
        return locDate == null ? null : Date.valueOf(locDate);
    }

    @Override
    public LocalDate convertToEntityAttribute(Date sqlDate) {
        return sqlDate == null ? null : sqlDate.toLocalDate();
    }
}
```

Propriedades de tipo *Collection* e *User-Defined*

```
@Entity
public class Person {

    @Id @GeneratedValue
    private int id;

    private String firstName;
    private String lastName;

    @Embedded
    private  Address address;
    @ElementCollection
    private Set<String> nicknames = new HashSet<String>();

    @ManyToOne
    private Family family;
```

```
@Embeddable
public class Address {

    private String streetCode;
    private int doorNum;
    private String zipcode;
```



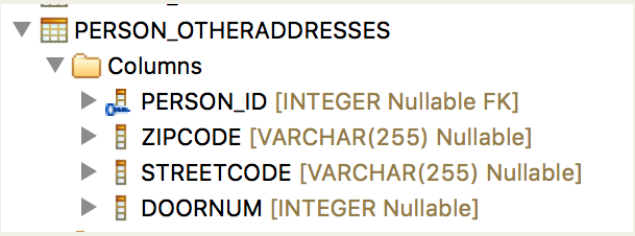


Propriedades de tipo *Collection* e *User-Defined*

```java
@Entity
public class Person {

    @Id @GeneratedValue
    private int id;

    private String firstName;
    private String lastName;

    @Embedded
    private  Address address;
```

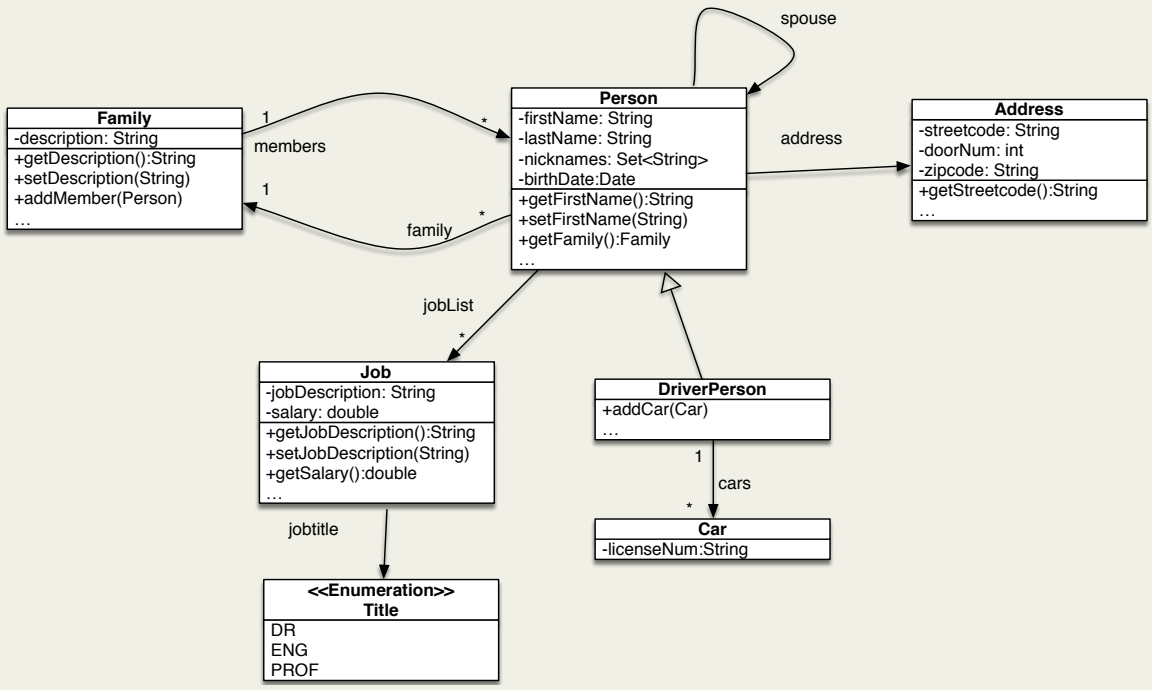
```java
@ElementCollection
private Set<Address> otherAddresses;
```

```java
@Embeddable
public class Address {

    private String streetCode;
    private int doorNum;
    private String zipcode;
```



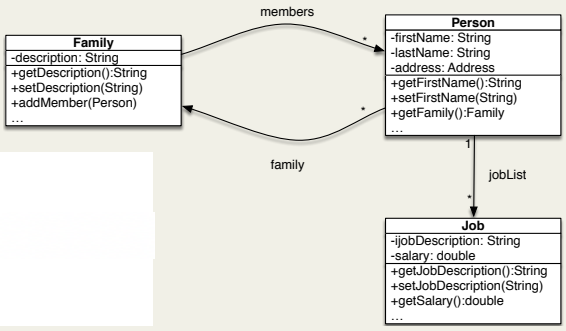
Ciências ULisboa | Informática

Mapeamento das associações



Ciências ULisboa | Informática

Associação Unidirecional entre *Person* e *Job* (v1)



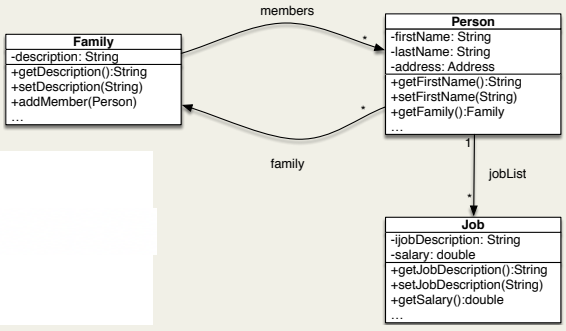

```java
@Entity
public class Person {
    @Id @GeneratedValue
    private int id;
    private String firstName;
    private String lastName;
```

```java
    @OneToMany
    private List<Job> jobList = new ArrayList<Job>();
```

```java
@Entity
public class Job {

    @Id @GeneratedValue
    private int idJob;
    private double salary;
    private String jobDescr;
```

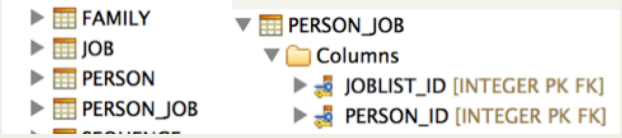
Ciências ULisboa | Informática

Associação Unidirecional entre *Person* e *Job* (v1)




```java
@Entity
public class Person {
    @Id @GeneratedValue
    private int id;
    private String firstName;
    private String lastName;
```

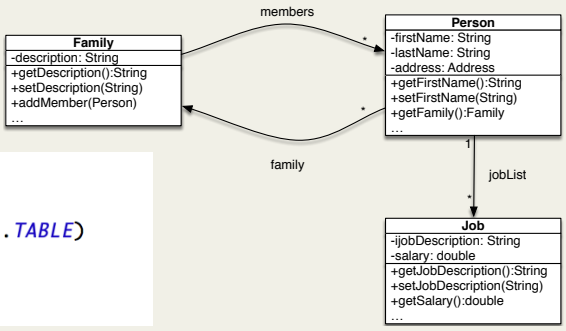
```java
    @OneToMany
    private List<Job> jobList = new ArrayList<Job>();
```




```java
@Entity
public class Job {

    @Id @GeneratedValue
    private int idJob;
    private double salary;
    private String jobDescr;
```

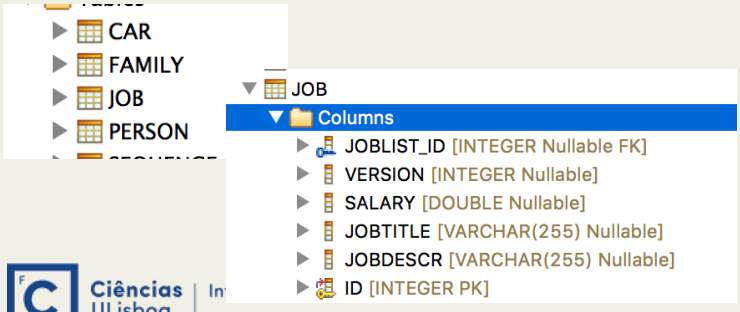
Ciências ULisboa | Informática

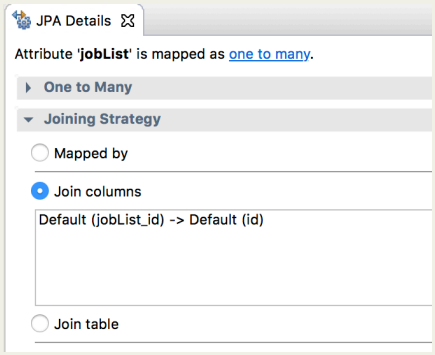
Associação Unidirecional entre *Person* e *Job* (v2)




```java
@Entity
public class Person {
    @Id @GeneratedValue(strategy = GenerationType.TABLE)
    private int id;
    private String firstName;
    private String lastName;
```

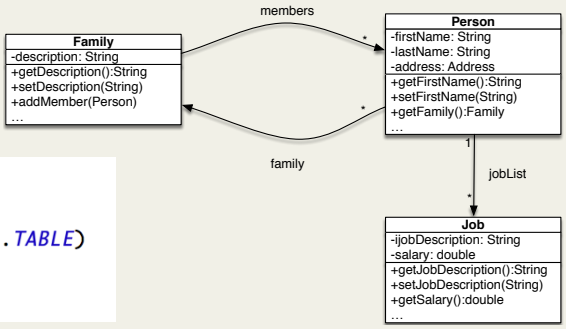
```java
@OneToMany @JoinColumn
private List<Job> jobList = new ArrayList<>();
```





Ciências ULisboa | In

Associação Bidirecional entre *Person* e *Family*




```java
@Entity
public class Family {
    @Id @GeneratedValue(strategy = GenerationType.TABLE)
    private int id;
    private String description;
```

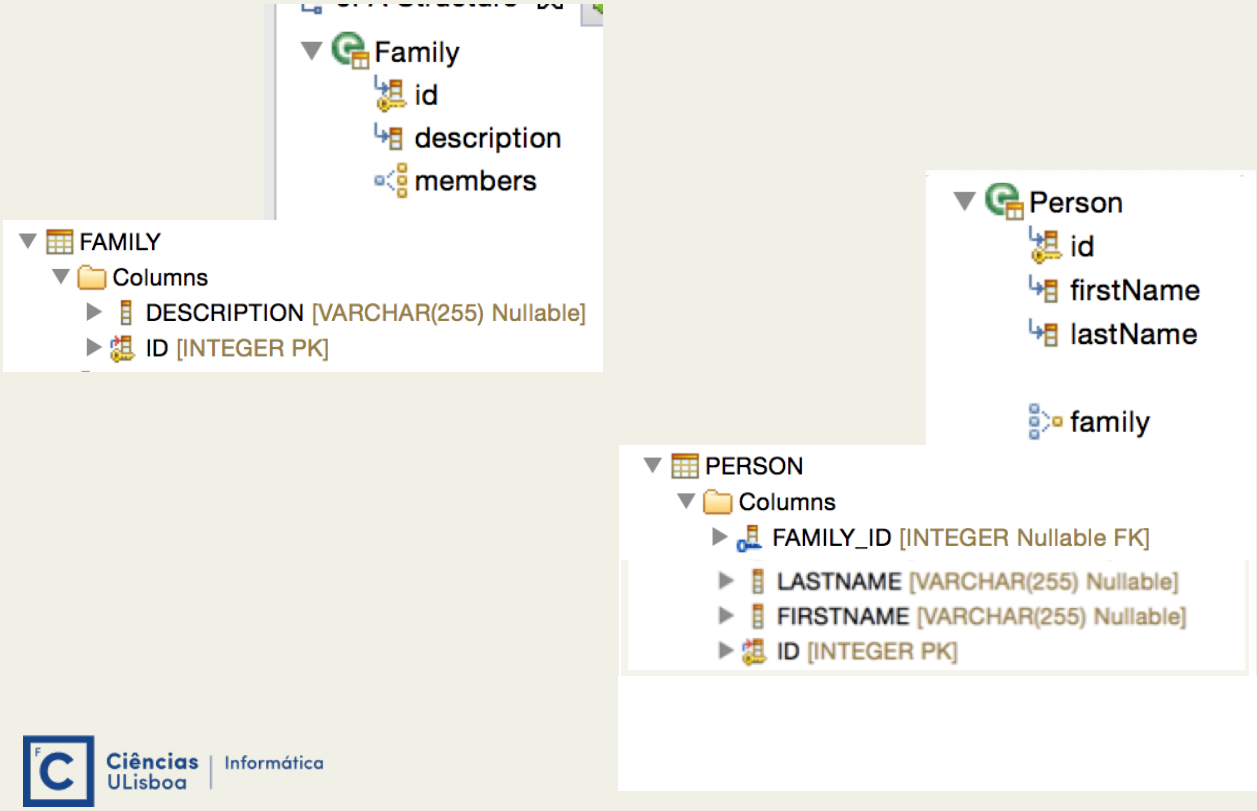
```java
    @OneToMany(mappedBy = "family")
    private List<Person> members = new ArrayList<Person>();
```

```java
@Entity
public class Person {
    @Id @GeneratedValue(strategy = GenerationType.TABLE)
    private int id;
    private String firstName;
    private String lastName;
```

```java
    @ManyToOne
    private Family family;
```

Ciências ULisboa | Informática

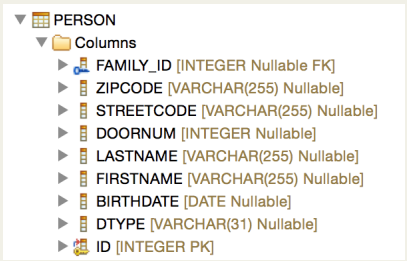
Associação Bidirecional entre *Person* e *Family*



Herança de *Person*

```
@Entity
@Inheritance(strategy=SINGLE_TABLE)
public class Person {
    @Id @GeneratedValue
    private int id;
    private String firstName;
    private String lastName;
    @Embedded
    private  Address adress;

    @ManyToOne
    private Family family;

    @ElementCollection
    private Set<String> nicknames;
```

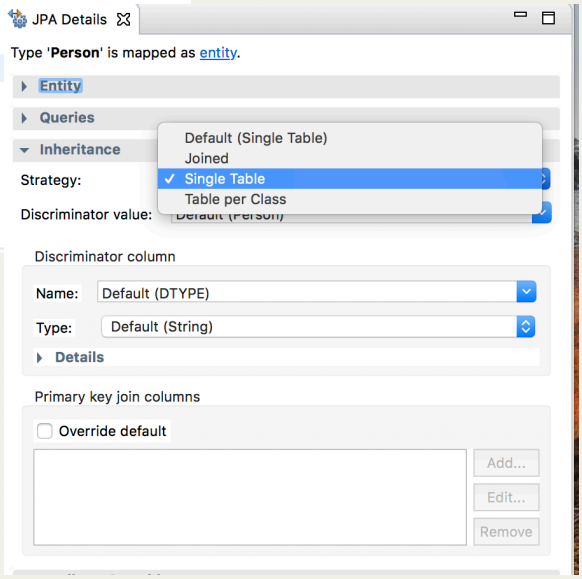
```
@Entity
public class DriverPerson extends Person {

    @OneToMany
    private Set<Car> cars;

    public DriverPerson(){
```



Herança de *Person*

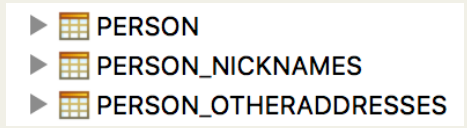
```
@Entity
@Inheritance(strategy=SINGLE_TABLE)
public class Person {
    @Id @GeneratedValue
    private int id;
    private String firstName;
    private String lastName;
    @Embedded
    private  Address adress;

    @ManyToOne
    private Family family;

    @ElementCollection
    private Set<String> nicknames;
```

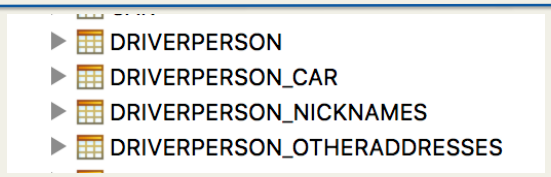
```
@Entity
public class DriverPerson extends Person {

    @OneToMany
    private Set<Car> cars;

            )riverPerson(){
```

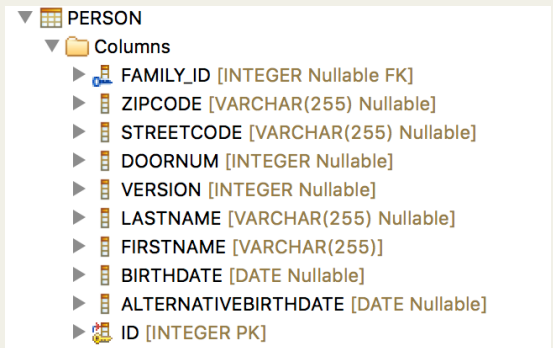


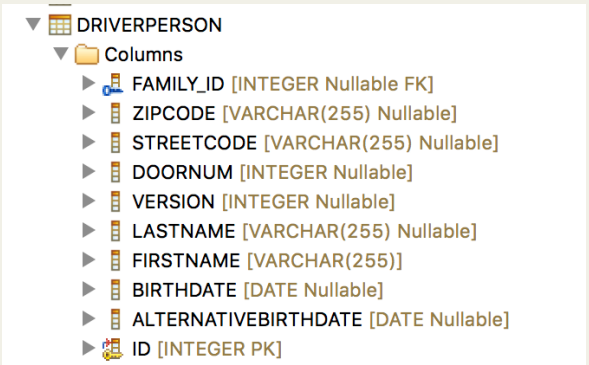
Se trocarmos de estratégia,
o que acontece?

Table per class









- Problemas com o uso do @JoinColumn no mapeamento de

  Person 1<-->* Job

  que só aparecem em run-time …

Joined

- CAR
- DRIVERPERSON
- DRIVERPERSON_CAR
- FAMILY
- JOB
- PERSON
- PERSON_NICKNAMES
- PERSON_OTHERADDRESSES

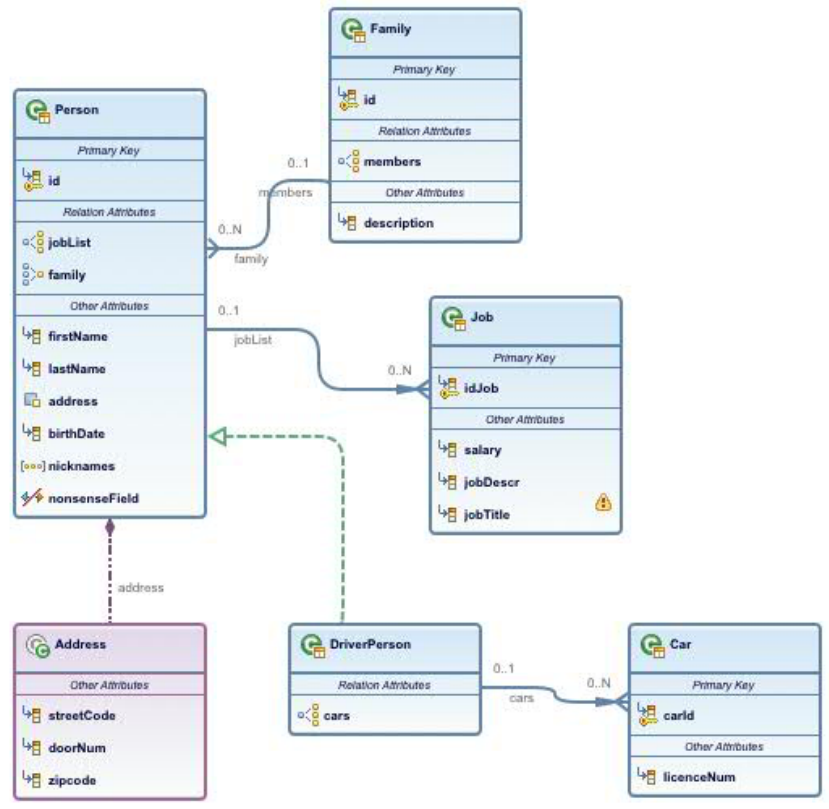
- PERSON
  - Columns
    - FAMILY_ID [INTEGER Nullable FK]
    - ZIPCODE [VARCHAR(255) Nullable]
    - STREETCODE [VARCHAR(255) Nullable]
    - DOORNUM [INTEGER Nullable]
    - VERSION [INTEGER Nullable]
    - LASTNAME [VARCHAR(255) Nullable]
    - FIRSTNAME [VARCHAR(255)]
    - BIRTHDATE [DATE Nullable]
    - ALTERNATIVEBIRTHDATE [DATE Nullable]
    - DTYPE [VARCHAR(31) Nullable]
    - ID [INTEGER PK]

- DRIVERPERSON
  - Columns
    - ID [INTEGER PK FK]

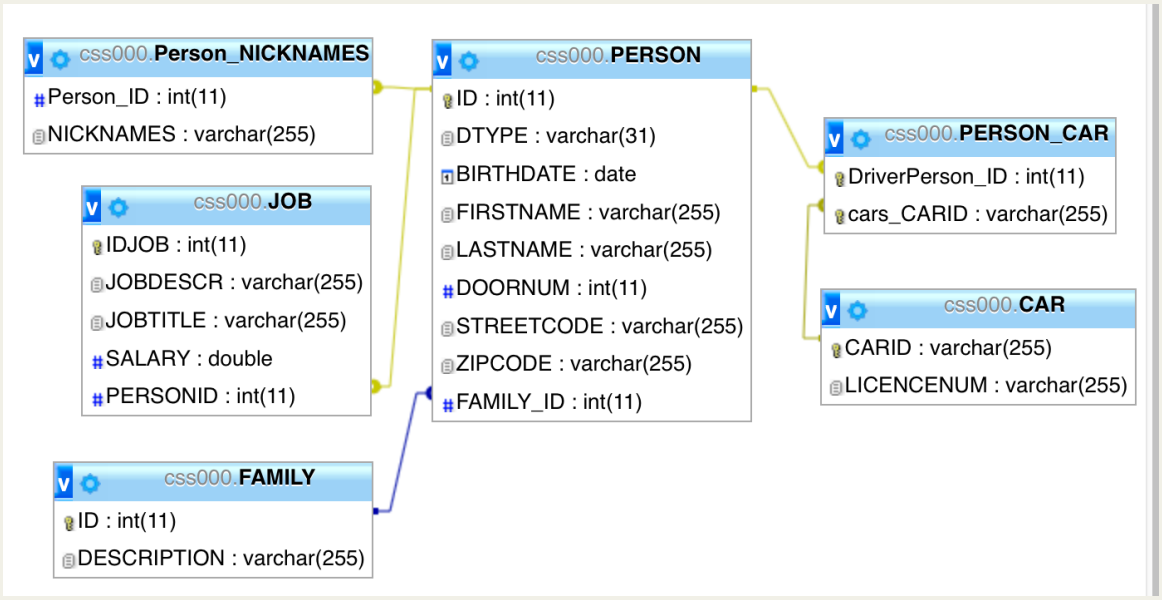
- DRIVERPERSON_CAR
  - Columns
    - CARS_CARID [VARCHAR(255) PK FK]
    - DRIVERPERSON_ID [INTEGER PK FK]

Ciências ULisboa | Informática

Diagrama que exibe ORM

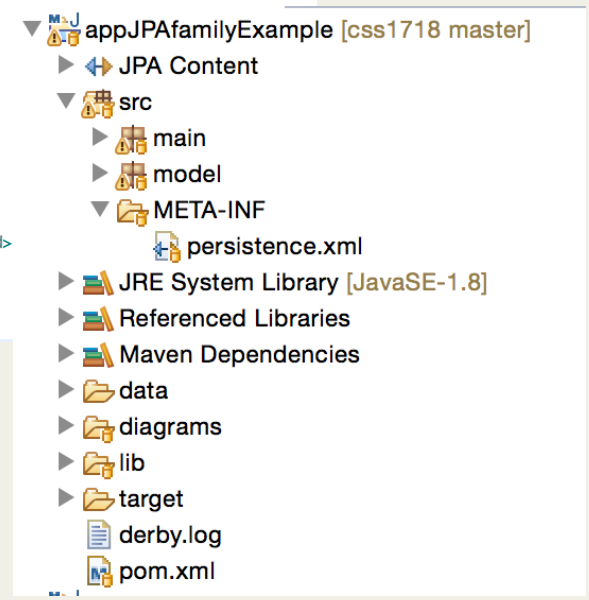


Modelo de Dados



Projeto JPA

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.
    <modelVersion>4.0.0</modelVersion>
    <groupId>pt.ulisboa.ciencia.di</groupId>
    <artifactId>family-jpa</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <description>An example of JPA project with resource local transactions
  </description>
    <properties>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.source>1.8</maven.compiler.source>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.apache.derby</groupId>
            <artifactId>derby</artifactId>
            <version>10.12.1.1</version>
        </dependency>
        <dependency>
            <groupId>org.eclipse.persistence</groupId>
            <artifactId>org.eclipse.persistence.jpa</artifactId>
            <version>2.6.4</version>
        </dependency>
    </dependencies>
</project>
```



Persistence.xml

PERSISTENCE UNIT NAME

Dados para a ligação

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persister
    version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">
    <persistence-unit name="people" transaction-type="RESOURCE_LOCAL">

        <class>model.Person</class>
        <class>model.DriverPerson</class>
        <class>model.Family</class>
        <class>model.Job</class>
        <class>model.Address</class>
        <class>model.Car</class>


        <properties>
            <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.EmbeddedDriver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:derby:data/relationsshipDb;create=true" />
            <property name="javax.persistence.jdbc.user" value="test" />
            <property name="javax.persistence.jdbc.password" value="test" />

            <!-- EclipseLink should create the database schema automatically -->
            <property name="eclipselink.ddl-generation" value="create-tables" />
            <property name="eclipselink.ddl-generation.output-mode" value="database" />
            <property name="eclipselink.exclude-eclipselink-orm" value="false"/>
            <property name="eclipselink.logging.level" value="INFO"/>
        </properties>

    </persistence-unit>
</persistence>
```


Gestor de Entidades

- Como o obter?

versão simples

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
EntityManager em = factory.createEntityManager();
```

- O que podemos fazer com ele?
  - obter objetos a partir da chave
  - persistir novos objetos
  - persistir a modificação de valores dos atributos de objetos
  - apagar objetos
  - …

  - find(Class entityClass,Object key)
  - persist(Object entity)
  - merge(Object entity)
  - remove(Object entity)

- Tudo dentro da transação que lhe está associada, delimitada por begin e commit

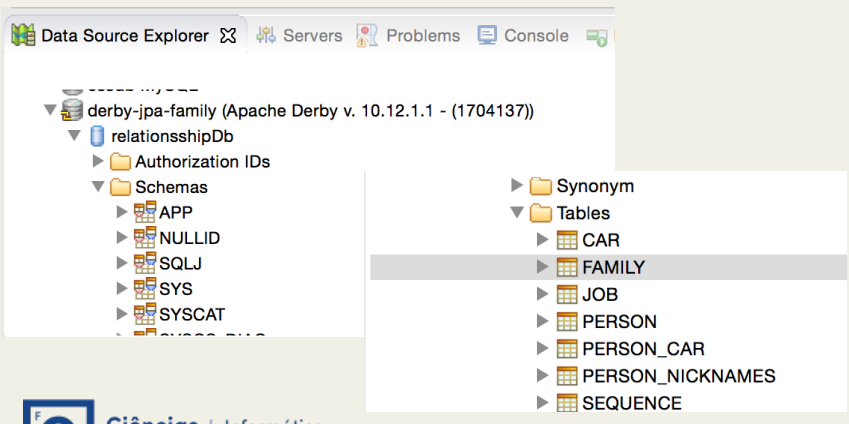
Exemplo Simples

```
EntityManagerFactory factory =
        Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
EntityManager em = factory.createEntityManager();

//create family
em.getTransaction().begin();
Family f = new Family();
f.setDescription("Martins");
em.persist(f);

Job j = new Job();
j.setJobDescr("CS Professor");
j.setTitle(Title.PROF);
em.persist(j);

Person p = new Person();
p.setFirstName("Francisco");
p.setLastName("Martins");
f.addMember(p);
em.persist(p);

ArrayList<Job> jl = new ArrayList<>();
jl.add(j);
p.setJobList(jl);

em.getTransaction().commit();
em.close();
```

```
//some queries
em = factory.createEntityManager();

Query q = em.createQuery("SELECT p FROM Person p");
q.setMaxResults(10);
System.out.println(q.getResultList());

Query qf = em.createQuery("SELECT f FROM Family f");
qf.setMaxResults(10);
System.out.println(qf.getResultList());

em.close();
```


Exemplo Simples

```
EL Info]: 2020-04-06 17:52:42.023--ServerSession(503642634)--EclipseLink,
version: Eclipse Persistence Services - 2.6.4.v20160829-44060b6
[EL Info]: connection: 2020-04-06 17:52:42.569--ServerSession(503642634)--
/file:/Users/antonialopes/Git/css/java/jpa-more-
examples/appJPAfamilyExample/target/classes/_people login successful

[Francisco Martins (202) * family: 201* jobs: {IndirectList: not
instantiated}* address: ]
[Martins (201) with members [ 202 ]]
```



Ciências ULisboa | Informática

Tutorial JPA + Eclipse

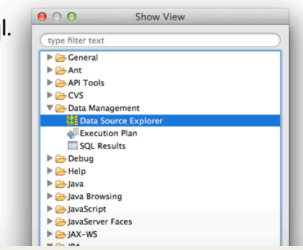
Construção de Sistemas de Software

Licenciatura em Engenharia Informática

Tutorial

Objetivos

- Familiarizar-se com a utilização de projetos JPA e do *EclipseLink* (a implementação de referência do JPA) no Eclipse IDE
- Configurar o Eclipse e criar uma ligação a uma base de dados MySql.
- Criar um pequeno projeto JPA que usa essa base de dados.



1. Configurar o Eclipse

O primeiro passo vai ser tornar visível as vistas de acesso à base de dados e de apoio à escrita de anotações JPA.

(a) Selecionar a opção do menu `Window -> Show View -> Other...`

Ciências ULisboa | Informática

JPA Editor