

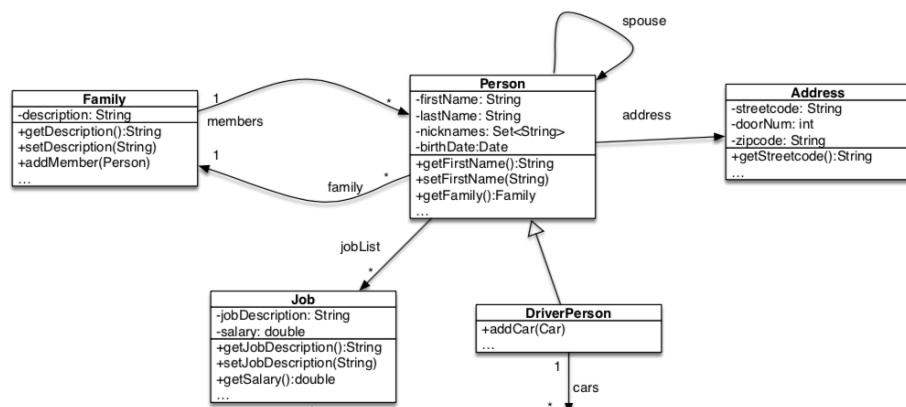
CONSTRUÇÃO DE SISTEMAS DE SOFTWARE

JPA — THE JAVA PERSISTENCE API
ORM Structural Patterns



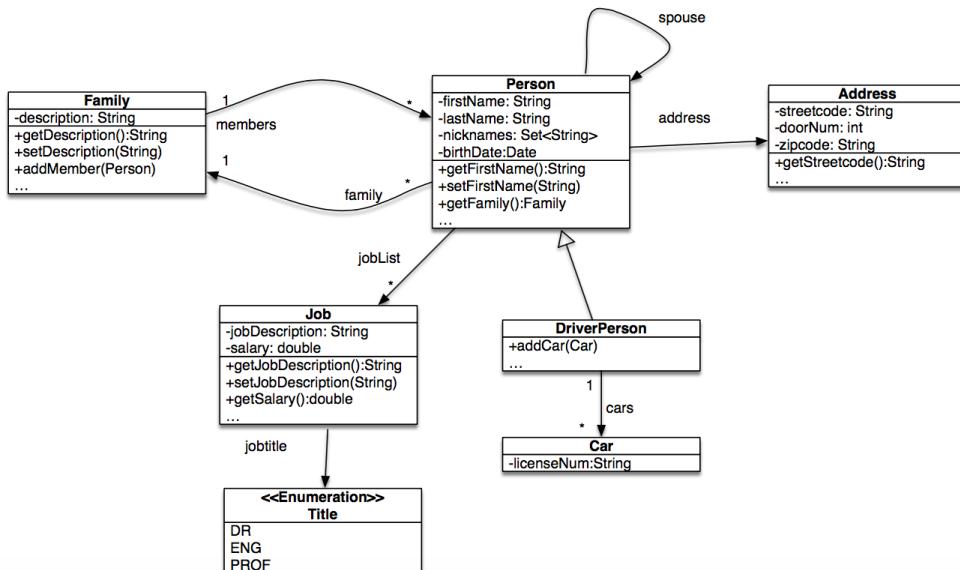
JPA Mapper (ORM Structural Patterns)

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.

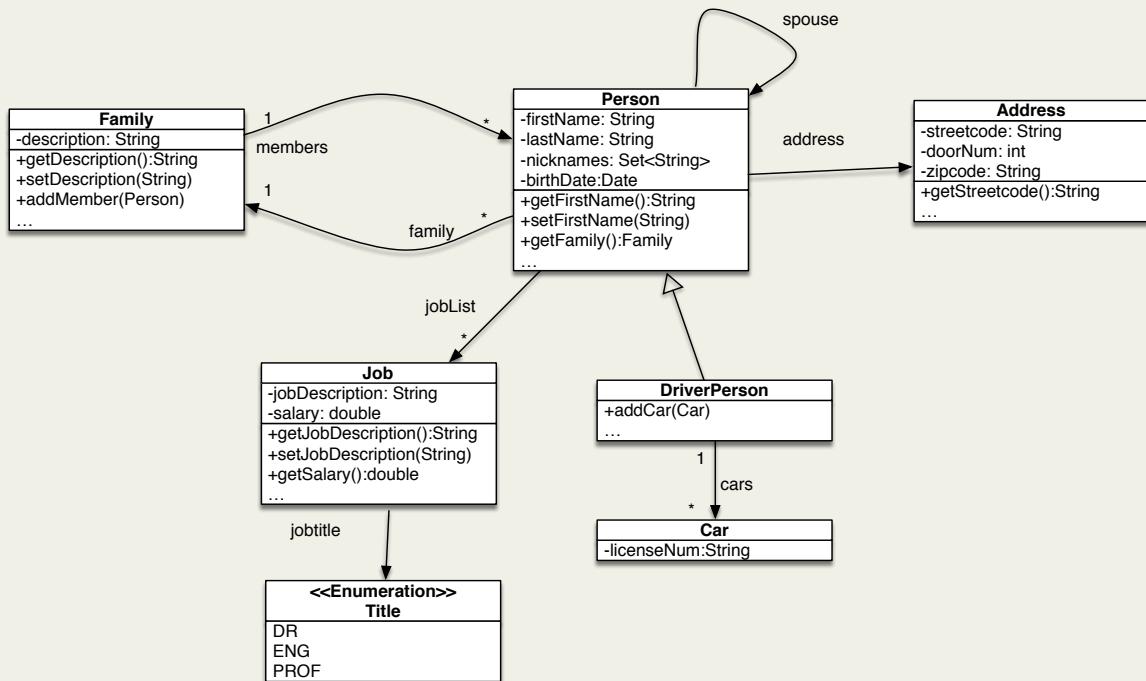


Exercício

Considere o diagrama de classes abaixo, que faz parte da camada de negócios 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



Ciências
ULisboa

| Informática

Entidades, Chaves e Propriedades de Tipo Básico

The screenshot shows the Eclipse IDE interface with two open windows:

- Person.java**: The Java code for the Person entity, including annotations for the primary key, first name, last name, and address.
- JPA Details**: A configuration dialog for the firstName attribute. It shows that 'firstName' is mapped as `default_(basic)`. The basic properties are set to 'Default (firstName)' and 'Default (Person)'. Under details, 'Insertable' and 'Updatable' are set to true, while 'Unique' is false and 'Nullable' is checked. The length is set to 255 and precision to 0.

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

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

JPA Provider: Eclipse Link



Ciências
ULisboa

| Informática

JPA De

Type 'Job' is mapped as [entity](#).

Entity

Name: Default (Job)

Catalog: Default

Schema: Default

Name: Default (Job)

Access: Default (Field)

ID class: <None> [Browse...](#)

[Caching](#)

[Queries](#)

[Inheritance](#)

[Attribute Overrides](#)

```
@Entity
public class Job {
    @Id @GeneratedValue
    private int idJob;
    private double salary;
    private String jobDescr;
```

JPA Details

Attribute 'idJob' is mapped as [ID](#).

ID

Column

Name: Default (idJob)

Table: Default (Job)

[Details](#)

Type

Primary Key Generation

Primary key generation

Strategy: [Default \(Auto\)](#)

Generator name:

Table Generator

Sequence Generator

JPA Structure

Job

- idJob
- salary
- jobDescr

Propriedades de Tipo Data e Enumerado

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

    private double salary;

    private String jobDescr;

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

```
@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;
```



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;

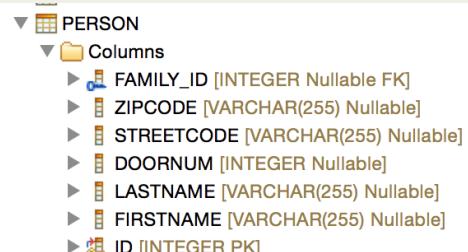
    private String firstName;
    private String lastName;

    @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;
```



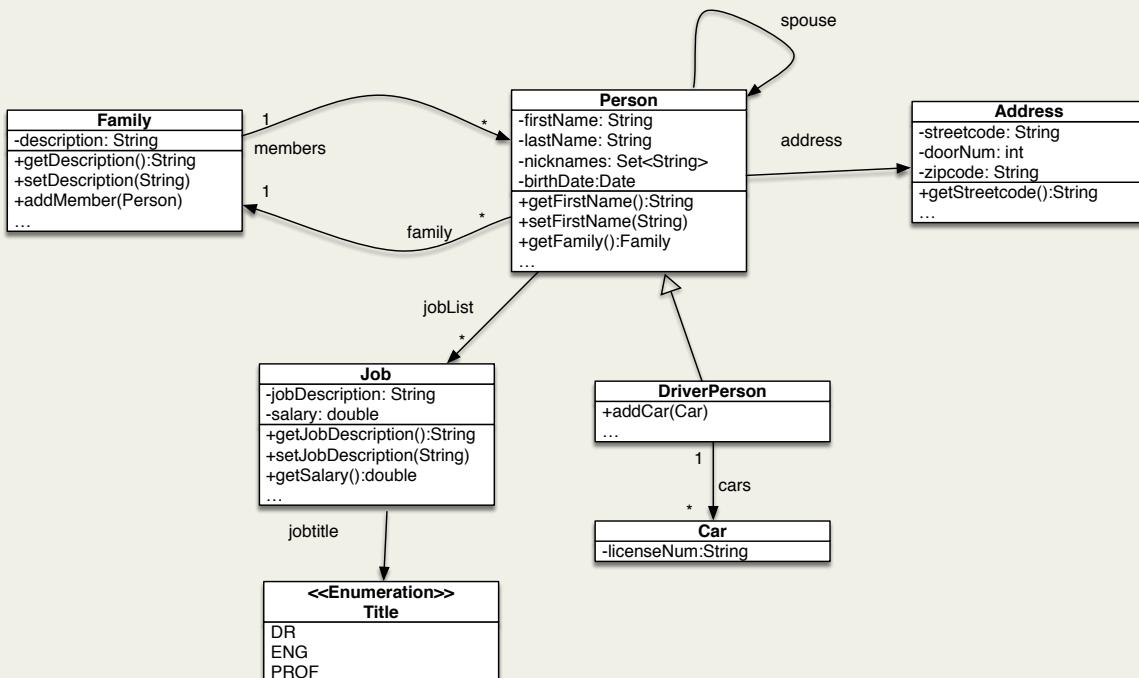
Propriedades de tipo *Collection* e *User-Defined*

```
@Entity  
public class Person {  
  
    @Id @GeneratedValue  
    private int id;  
  
    private String firstName;  
    private String lastName;  
  
    @Embedded  
    private Address address;  
  
    @ElementCollection  
    private Set<Address> otherAddresses;
```

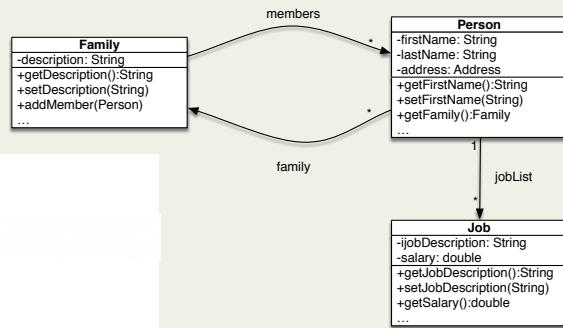
```
@Embeddable  
public class Address {  
  
    private String streetCode;  
    private int doorNum;  
    private String zipcode;
```

▼ PERSON_OTHERADDRESSES
 ▼ Columns
 ► PERSON_ID [INTEGER Nullable FK]
 ► ZIPCODE [VARCHAR(255) Nullable]
 ► STREETCODE [VARCHAR(255) Nullable]
 ► DOORNUM [INTEGER Nullable]

Mapeamento das associações



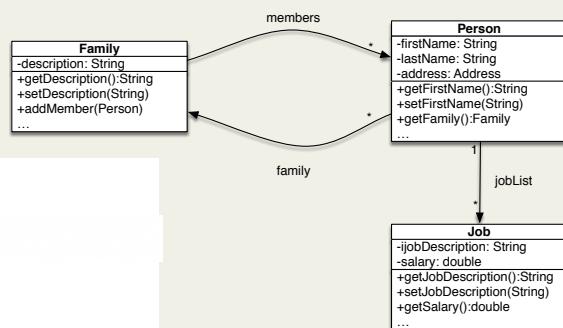
Associação Unidirecional entre Person e Job (v1)



```
@Entity  
public class Person {  
    @Id @GeneratedValue  
    private int id;  
    private String firstName;  
    private String lastName;  
  
    @OneToMany  
    private List<Job> jobList = new ArrayList<Job>();
```

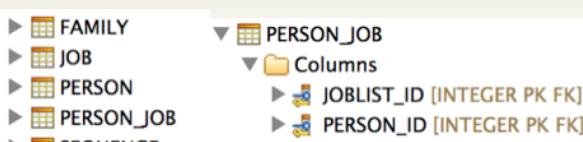
```
@Entity  
public class Job {  
    @Id @GeneratedValue  
    private int idJob;  
    private double salary;  
    private String jobDescr;
```

Associação Unidirecional entre Person e Job (v1)



```
@Entity  
public class Person {  
    @Id @GeneratedValue  
    private int id;  
    private String firstName;  
    private String lastName;  
  
    @OneToMany  
    private List<Job> jobList = new ArrayList<Job>();
```

```
@Entity  
public class Job {  
    @Id @GeneratedValue  
    private int idJob;  
    private double salary;  
    private String jobDescr;
```



Associação Unidirecional entre Person e Job (v2)

```

classDiagram
    class Family {
        -description: String
        +getDescription():String
        +setDescription(String)
        +addMember(Person)
        ...
    }
    class Person {
        -firstName: String
        -lastName: String
        -address: Address
        +getFirstName():String
        +setFirstName(String)
        +getFamily():Family
        ...
    }
    class Job {
        -jobDescription: String
        -salary: double
        +getJobDescription():String
        +setJobDescription(String)
        +getSalary():double
        ...
    }
    Family "members" --> "*" Person
    Person "family" --> "*" Job
    Person "jobList" --> "1" Job
  
```

`@Entity`

```

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

    @OneToMany @JoinColumn
    private List<Job> joblist = new ArrayList<>();
}

```

JPA Details

Attribute 'jobList' is mapped as [one to many](#).

- One to Many
- Joining Strategy
- Mapped by
- Join columns
- Default (jobList_id) -> Default (id)
- Join table

Ciências ULisboa | Informática

Associação Bidirecional entre Person e Family

```

classDiagram
    class Family {
        -description: String
        +getDescription():String
        +setDescription(String)
        +addMember(Person)
        ...
    }
    class Person {
        -firstName: String
        -lastName: String
        -address: Address
        +getFirstName():String
        +setFirstName(String)
        +getFamily():Family
        ...
    }
    class Job {
        -jobDescription: String
        -salary: double
        +getJobDescription():String
        +setJobDescription(String)
        +getSalary():double
        ...
    }
    Family "members" <--> "*" Person
    Person "family" <--> "*" Job
    Person "jobList" --> "1" Job
  
```

`@Entity`

```

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

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

```

`@Entity`

```

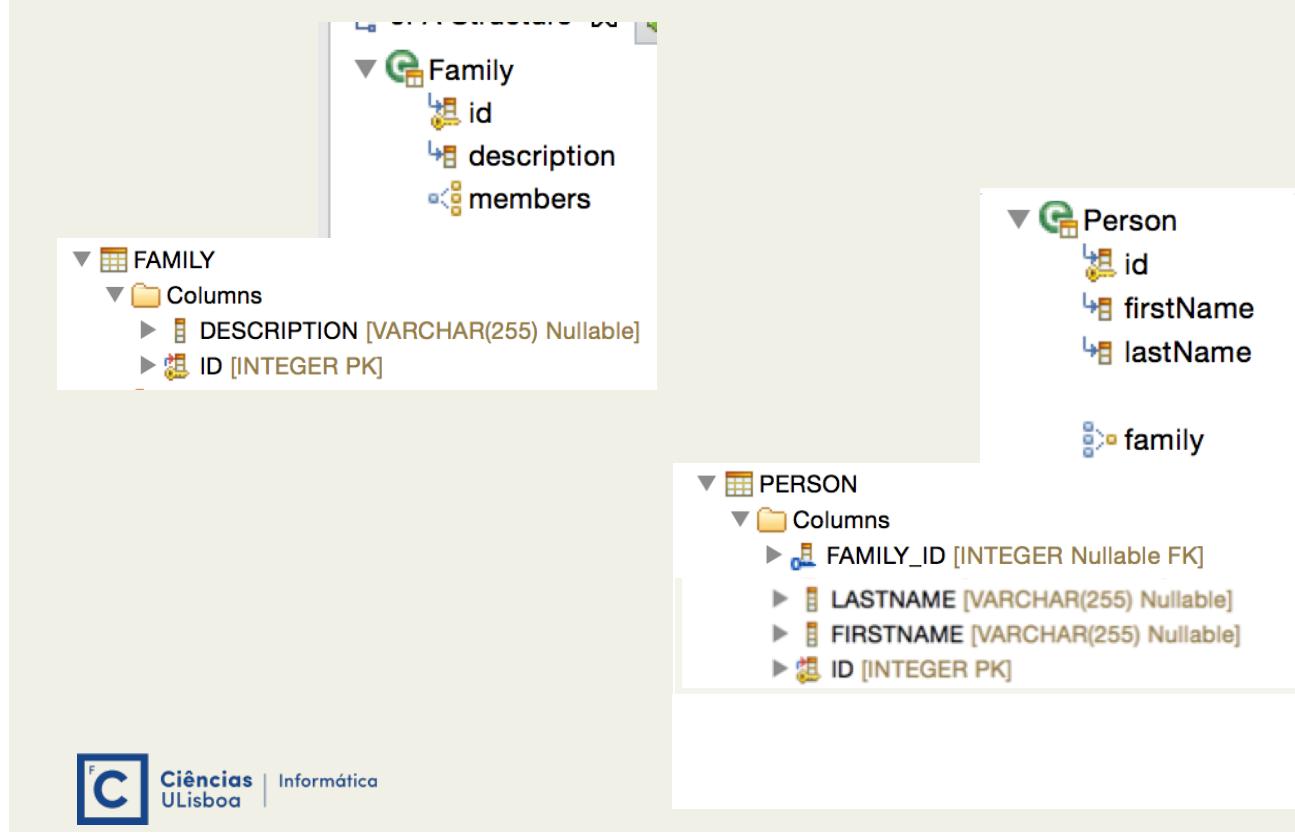
public class Person {
    @Id @GeneratedValue(strategy = GenerationType.TABLE)
    private int id;
    private String firstName;
    private String lastName;

    @ManyToOne
    private Family family;
}

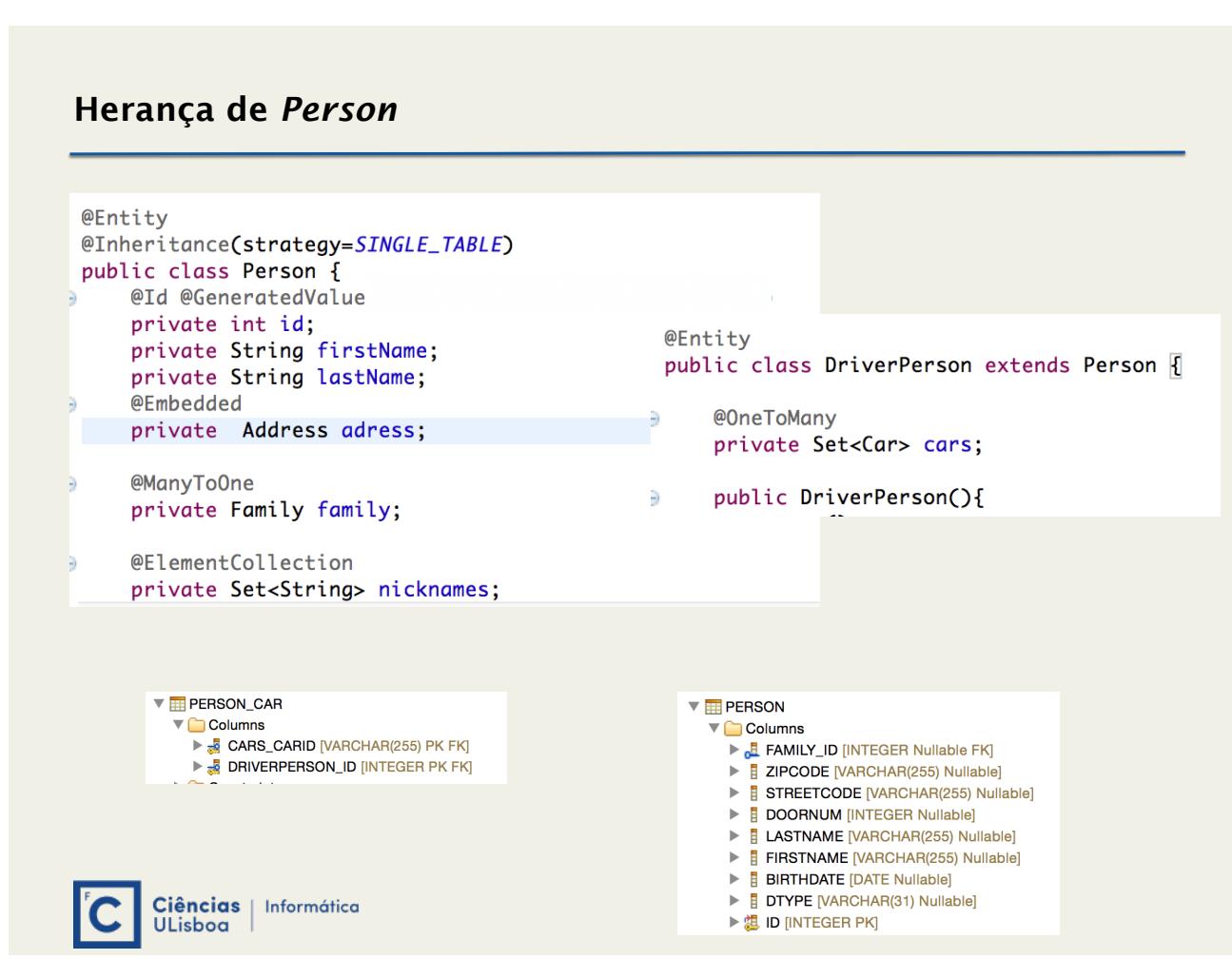
```

Ciências ULisboa | Informática

Associação Bidirecional entre *Person* e *Family*



Herança de *Person*



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;  
  
    DriverPerson(){
```

JPA Details Type 'Person' is mapped as entity.

Entity

Queries

Inheritance

Strategy:

Discriminator value: Default (Person)

Default (Single Table)

Joined

Single Table

Table per Class

Discriminator column

Name: Default (DTYPE)

Type: Default (String)

Details

Primary key join columns

Override default

Add...
Edit...
Remove

Se trocarmos de estratégia,
o que acontece?



Ciências
ULisboa | Informática

Table per class

- ▶ 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]
 - ▶ ID [INTEGER PK]

- ▶ DRIVERPERSON
- ▶ DRIVERPERSON_CAR
- ▶ DRIVERPERSON_NICKNAMES
- ▶ DRIVERPERSON_OTHERADDRESSES

▼ DRIVERPERSON

- ▼ 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]
 - ▶ ID [INTEGER PK]

- Problemas com o uso do @JoinColumn no mapeamento de

Person 1<->* Job

que só aparecem em run-time ...



Ciências
ULisboa | Informática

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]
 - ▶ DTYP [VARCHAR(31) Nullable]
 - ▶ ID [INTEGER PK]

▼ DRIVERPERSON

- ▼ Columns
 - ▶ ID [INTEGER PK FK]

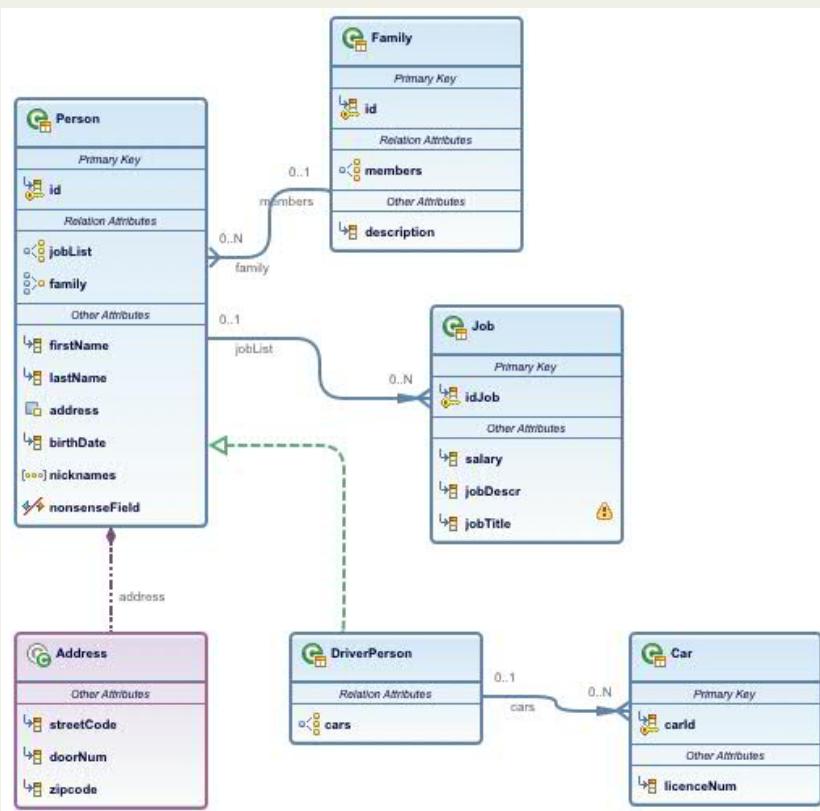
▼ DRIVERPERSON_CAR

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

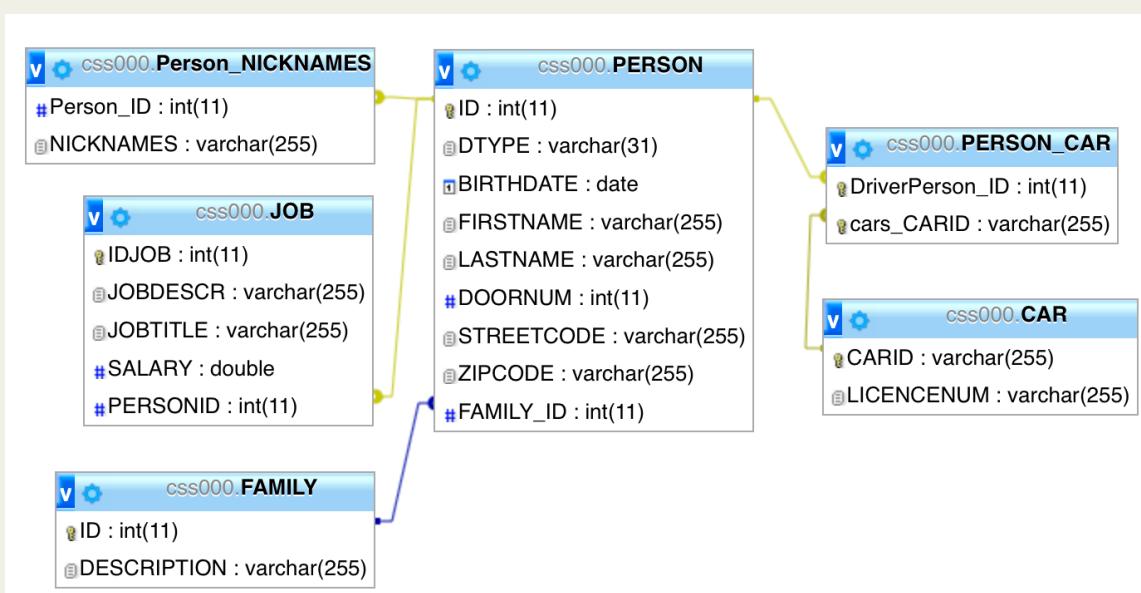


Ciências
ULisboa | Informática

Diagrama que exibe ORM



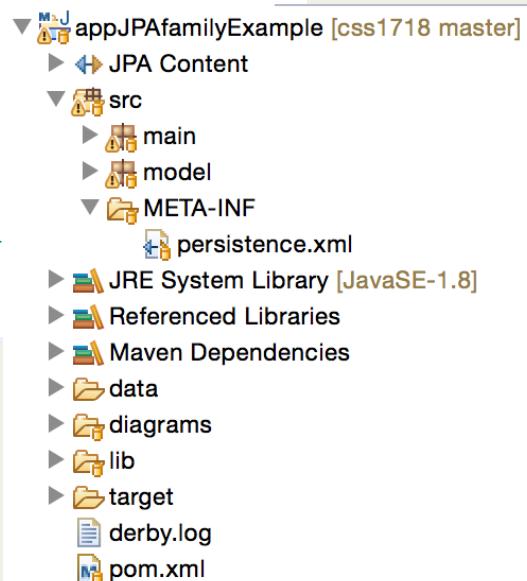
Modelo de Dados



Ciências
ULisboa

Projeto JPA

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <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>
```



Ciências
ULisboa

Persistence.xml

```
<?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/persistence_2_0.xsd"
    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>
        <br/>
        <properties>
            <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.EmbeddedDriver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:derby:data/relationshipDb;create=true" />
            <property name="javax.persistence.jdbc.user" value="test" />
            <property name="javax.persistence.jdbc.password" value="test" />
            <br/>
            <!-- 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>
```



Ciências
ULisboa | Informática

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`



Ciências | Informática

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 ]]
```

The screenshot shows the Eclipse IDE's Data Source Explorer view. A tree structure displays a database named 'derby-jpa-family'. Under this database, there is a 'relationshipDb' schema which contains several tables: APP, NULLID, SQLJ, SYS, SYSCAT, and SYSCONS. To the right of the tree, a detailed list of tables is shown, including Synonym, Tables, and specific tables like CAR, FAMILY, JOB, PERSON, PERSON_CAR, PERSON_NICKNAMES, and SEQUENCE. The 'FAMILY' table is currently selected.





Construção de Sistemas de Software

Licenciatura em Engenharia Informática

Tutorial

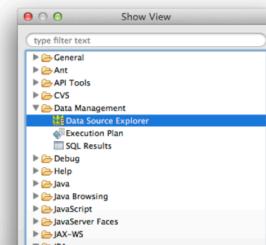
Objetivos

- 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.

I. 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...



JPA Editor

