1 أنواع العلاقات في JPA – JPA Relationship

العلاقة، في سياق قواعد البيانات، هي حالة توجد بين جدولي قاعدة بيانات علاقية عندما يحتوي أحدهما على مفتاح خارجي يشير إلى المفتاح الأساسي للجدول الآخر. تسمح العلاقات لقواعد البيانات العلاقية بتقسيم البيانات وتخزينها في جداول مختلفة، مع ربط عناصر البيانات.
هناك 4 أنواع للعلاقات، وهي: OneToOne – OneToMany – ManyToOne - ManyToMany.
في البداية، لنعرف بعض العلاقات في المثال:

- Teacher – Course
- Teacher – Address
- Student – Course

1.1 OneToMany/ManyToOne

كما يوحي اسمها، فهي علاقة تربط كيانًا بالعديد من الكيانات الأخرى.
في مثالنا، سيكون هذا مدرسًا ودوراته التدريبية. يمكن للمدرس أن يعطي دورات متعددة، لكن الدورة يتم تقديمها من قبل مدرس واحد فقط (من منظور ManyToOne - العديد من الدورات لمعلم واحد).

قبل الغوص في تفاصيل كيفية تعيين هذه العلاقة، دعنا ننشئ كياناتنا:

```
@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Teacher {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Name must not be null")
    @Column(nullable = false)
    private String name;

}

@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Title must not be null")
    @Column(nullable = false)
    private String title;

}
```

الآن، يجب أن تتضمن حقول Teacher Class قائمة بالـ Courses نظرًا لأننا نرغب في تعيين هذه العلاقة في قاعدة بيانات، ويجب تعيين حقل Teacher في Course Class ايضاً، فسنقوم باستخدام التعليق التوضيحي @OneToMany في Teacher Class واستخدام @ManyToOne في Course Class:

```
@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Teacher {

    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Name must not be null")
    @Column(nullable = false)
    private String name;

    @OneToMany(mappedBy = "teacher")
    private Set<Course> courses;
}

@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Title must not be null")
    @Column(nullable = false)
    private String title;

    @ManyToOne
    @JoinColumn(name = "teacher_id" , referencedColumnName = "id")
    @JsonIgnore
    private Teacher teacher;

}
```

كيف تعكس JPA هذه العلاقة في قاعدة البيانات؟ بشكل عام، لهذا النوع من العلاقات، يجب علينا استخدام مفتاح خارجي في جدول.
يقوم JPA بهذا من أجلنا، بالنظر إلى مدخلاتنا حول كيفية التعامل مع العلاقة. يتم ذلك عبر التعليق التوضيحي @JoinColumn ، سيؤدي استخدام هذا التعليق التوضيحي إلى إخبار JPA بأن جدول Course يجب أن يحتوي على عمود مفتاح خارجي teacher_id يشير إلى عمود معرف جدول Teacher.

المهم أن نلاحظ هنا استخدام علامة mappedBy في التعليق التوضيحي OneToMany. بدونها، لن يكون لدينا علاقة ثنائية الاتجاه. سيكون لدينا علاقتان في اتجاه واحد. يقوم كلا الكيانين بتعيين المفاتيح الخارجية للكيان الآخر.
باستخدامه، نخبر JPA أن الحقل تم تعيينه بالفعل بواسطة كيان آخر. تم تعيينه بواسطة حقل Teacher في كيان Course.

@JsonIgnore يتم استخدامه على مستوى الحقل لتمييز خاصية أو قائمة بالخصائص التي سيتم تجاهلها.

OneToOne 1.2

إنها علاقة تربط كيان واحد بسجل واحد بالضبط في كيان آخر.
في مثالنا، سيكون هذا مدرسًا وعنوانه. لكل مدرس عنوان واحد فقط، وكل عنوان تابع لمدرس واحد فقط.

```
@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Teacher {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Name must not be null")
    @Column(nullable = false)
    private String name;

    @OneToMany(mappedBy = "teacher")
    private Set<Course> courses;

    @OneToOne(cascade = CascadeType.ALL , mappedBy = "teacher")
    @PrimaryKeyJoinColumn
    private Address address;

}


@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Address {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Name must not be null")
    @Column(nullable = false)
    private String name;

    @OneToOne
    @JoinColumn(name = "teacher_id" , referencedColumnName = "id")
    @MapsId
    @JsonIgnore
    private Teacher teacher;
}
```

نقوم بتعيين ارتباط حقيقي OneToOne باستخدام المفاتيح الأساسية المشتركة.
يوضح التعليق التوضيح @PrimaryKeyJoinColumn أنه يتم استخدام المفتاح الأساسي للكيان كقيمة مفتاح خارجي للكيان المرتبط.
يتم استخدام @MapsId إذا كان المفتاح الأساسي للكيان المرتبط هو نفسه المفتاح الأساسي للكيان الاصلي. من خلال تحديد التعليق التوضيحي، يمكنك ملء المفتاح الأساسي للكيان المرتبط تلقائيًا من للكيان الأصلي.
تعني كلمة Cascading عندما نقوم بتنفيذ بعض الإجراءات على الكيان المستهدف، فسيتم تطبيق نفس الإجراء على الكيان المرتبط.

ManyToMany 1.3

في قاعدة البيانات، تتضمن علاقة ManyToMany جدولًا متوسطًا يشير إلى كلا الجدولين الآخرين.
لحسن الحظ بالنسبة لنا ، تقوم JPA بمعظم العمل ، وعلينا فقط إلقاء بعض التعليقات التوضيحية هناك ، وهي تتولى الباقي نيابةً عنا.
لذلك، على سبيل المثال، ستكون علاقة ManyToMany هي العلاقة بين الطالب والدورة التدريبية حيث يمكن للطالب حضور دورات متعددة ، ويمكن أن يتبع الدورة العديد من الطلاب.

```
@Data @NoArgsConstructor @AllArgsConstructor @Entity
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name;

    @ManyToMany(mappedBy = "students")
    @JsonIgnore
    private List<Course> courses;
}

@Setter @Getter
@AllArgsConstructor @NoArgsConstructor @Entity
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    @NotEmpty(message = "Title must not be null")
    @Column(nullable = false)
    private String title;

    @ManyToOne
    @JoinColumn(name = "teacher_id" , referencedColumnName = "id")
    @JsonIgnore
    private Teacher teacher;

    @ManyToMany
    private List<Student> students;


}
```