Relacionamento

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Modos de relacionamento
- Modos de carregamento

Relacionamentos

No JPA temos como mapear entidades conforme seus relacionamentos de foreign key. As anotações disponíveis são:

- One-to-Many
- Many-to-One
- One-to-One
- Many-to-Many

@OneToMany

O relacionamento @OneToMany, como o nome diz, é o relacionamento onde uma entidade pode ter relacionamento com muitas entidades, neste caso, com essa anotação, ao buscar a entidade, retorna a lista de entidades marcadas no “many”.

```
@OneToMany
private List<Course> courses;
```

@ManyToOne

Ao contrário da anotação anterior, o @ManyToOne faz referência de que a entidade responde a várias outras mas que nela é direcionada um único objeto.

```
@ManyToOne
@JoinColumn(name = "TEACHER_ID", referencedColumnName = "ID")
private Teacher teacher;
```

```
@Entity
public class Teacher {
    // ...

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

@Entity
public class Course {
    // ...

    @ManyToOne
    @JoinColumn(name = "TEACHER_ID", referencedColumnName = "ID")
    private Teacher teacher;
}
```

@OneToOne

Com o @OneToOne, ao invés de ter um relacionamento entre uma entidade de um lado e um monte de entidades do outro, teremos no máximo uma entidade de cada lado.

```
@OneToOne(optional = false)
@JoinColumn(name = "COURSE_ID", referencedColumnName = "ID")
private Course course;
```

@ManyToMany

Agora, por último, mas não menos importante: relacionamentos muitos-para-muitos. Deixamos estes para o final porque exigem um pouco mais de trabalho do que os anteriores.

Efetivamente, em um banco de dados, um relacionamento Muitos-para-Muitos envolve uma tabela do meio referenciando ambas as outras tabelas.

Felizmente para nós, o JPA faz a maior parte do trabalho, só precisamos lançar algumas anotações e ele cuida do resto para nós.

Portanto, para o nosso exemplo, o relacionamento Muitos-para-Muitos será aquele entre as instâncias Aluno e Curso, pois um aluno pode frequentar vários cursos e um curso pode ser seguido por vários alunos.

Para mapear um relacionamento Muitos-para-Muitos, usaremos a notação @ManyToMany. No entanto, desta vez, também usaremos uma anotação @JoinTable para configurar a tabela que representa o relacionamento.

@ManyToMany

```
@ManyToMany
@JoinTable(
  name = "STUDENTS_COURSES",
  joinColumns = @JoinColumn(name = "COURSE_ID", referencedColumnName =
"ID"),
  inverseJoinColumns = @JoinColumn(name = "STUDENT_ID",
referencedColumnName = "ID")
)
private List<Student> students;
```

Lazy Loading

Vamos começar estudando o Lazy loading que é o mais comum de ser encontrado nos mapeamentos realizados pelo Hibernate, afinal, todos os mapeamentos do Hibernate são Lazy por padrão, se você não especificar o tipo de “fetch” (busca) que será realizada.

Enfim, o Lazy Loading faz com que determinados objetos não sejam carregados do banco até que você precise deles, ou seja, são carregados 'on demand' (apenas quando você solicitar explicitamente o carregamento destes).

```
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "id_endereco")
private Endereco endereco;
```

Eager Loading

Oposto ao Lazy Loading, o Eager Loading carrega os dados mesmo que você não vá utilizá-los, mas é óbvio que você só utilizará esta técnica se de fato você for precisar com muita frequência dos dados carregados.

```
@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "id_endereco")
private Endereco endereco;
```

AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

Clique aqui ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>