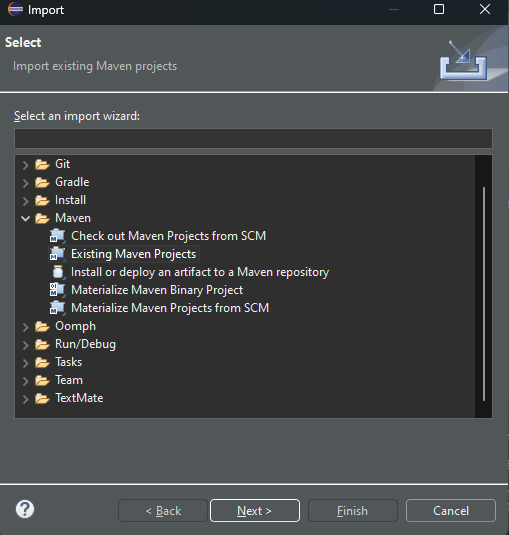
**Entity:**

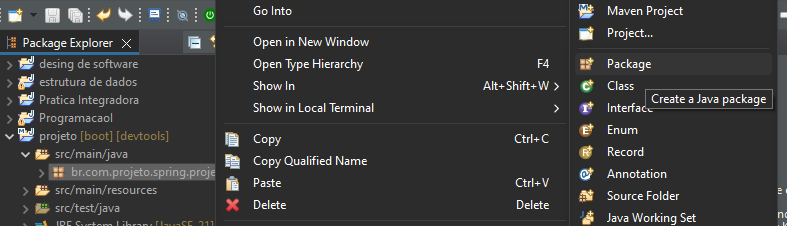
As camadas de nossa API são criadas a partir de “packages” JAVA.

Criando o primeiro package na nossa spring:

1ª como eu usei o eclipse, eu importo o spring feito no site spring.io assim:



Depois de importado, agora vamos criar o package:



.entity pois estamos fazendo o pacote de entidades.

**Criando o repository:**

Semelhante, crio um novo package e .repository

No Spring, a camada Repository é responsável pela persistência dos dados, trafegados ao longo da aplicação através da(s) instância(s) de uma ou mais Entidades no banco de dados. Logo, todas as operações relacionadas ao banco ficam sob responsabilidade dessa camada – sejam elas de recuperação, persistência, atualização ou deleção de registros.

Diferente das Entities, no Repositório temos **interfaces** ao invés de classes. Além disso, **cada Entidade deverá possuir sua própria interface** na camada em questão, ou seja, se criarmos uma classe chamada Aluno na camada Entity, deveremos então criar uma interface, na camada Repository, chamada AlunoRepository. Repare, aqui, na convenção de se utilizar a palavra “Repository” como sufixo ao nome da Entidade. Isso ajuda a identificar, pelo nome do arquivo, a qual entidade um repositório se refere.

**Criando o controller:**

Semelhante, crio um novo package e .controller

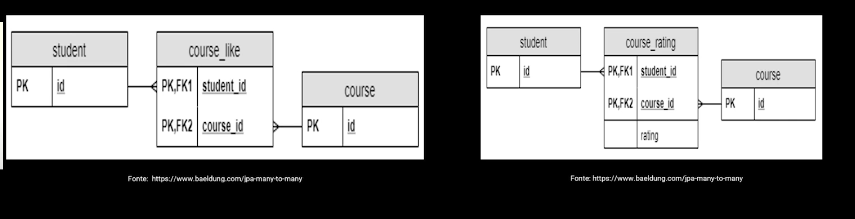
A camada Controller, no Spring, contém classes responsáveis por responder a requisições HTTP/HTTPS. Logo, nela inserimos os **recursos que queremos expor em nossa API/ e que queremos disponibilizar para serem utilizados por clientes**, como outros sistemas de software, ou mesmo por outras aplicações em nosso próprio ecossistema, por exemplo, podemos ter um sistema composto por uma aplicação web e por um aplicativo mobile, em que ambos consomem serviços disponíveis em nossa API desenvolvida utilizando Spring, fazendo requisições sobre o protocolo HTTP, tendo como alvo a camada Controller. Isso posto, e seguindo o padrão do framework, devemos ter, não obrigatoriamente, mas de forma recomendada, uma classe na camada Controller para cada uma das entidades que compõem nossa API.

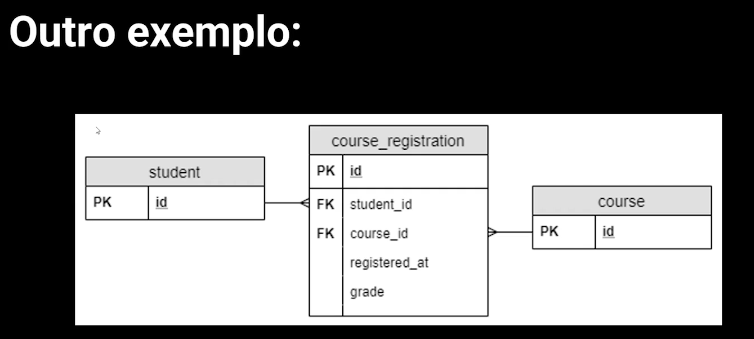
Além disso, tais classes recebem algumas anotações especiais – anotações essas que, a nível de classe, informam que a classe em questão é um Controller do tipo REST (@RestController); e, a nível de recursos e serviços disponibilizados, representados pelos métodos que compõem a classe, o tipo de acesso a cada um deles (@GetMapping, @PostMapping, @PutMapping, @DeleteMapping e @PathMapping).

**Relacionamentos complexos**:

Relacionamentos many-to-many.

A complexidade, se dá pelas diferentes características possíveis num relacionamento muito-para-muito, como a existência de atributos adicionais na Entidade de relacionamento, por exemplo.





Na pratica:

