**DTO (Data Transfer Object):**

**Padrão ideal** em aplicações Spring (e em qualquer app com boas práticas de arquitetura em camadas):

**🔄 Fluxo correto de uma requisição:**

1. **O cliente (usuário, app, Postman, etc.) envia JSON →**  
   O @RequestBody converte esse JSON em um **DTO** (PersonDTO).
2. **O Controller chama o Service →**  
   O Service recebe o DTO e o **converte em uma entidade** (Person) para poder:
   * Salvar no banco (repository.save()), ou
   * Atualizar os dados, ou
   * Buscar informações.
3. **O resultado (entidade)** é então **convertido de volta em DTO**, e esse DTO é devolvido ao cliente como resposta JSON.

**📊 Esquematicamente:**

Cliente → JSON → DTO → Entity → Banco

Banco → Entity → DTO → JSON → Cliente

**🚫 O que *não* deve acontecer:**

* ❌ O usuário nunca deve enviar ou receber uma **entidade JPA** diretamente.
* ❌ A entidade nunca deve "vazar" para fora da camada de serviço ou repositório.
* ❌ O controller nunca deve manipular a entidade diretamente.

**⚙️ Motivos:**

| **Motivo** | **Explicação** |
| --- | --- |
| **Segurança** | Evita expor campos sensíveis da entidade. |
| **Desempenho** | Evita problemas com lazy loading e relacionamentos. |
| **Flexibilidade** | Você pode alterar o banco sem mudar a API. |
| **Organização** | Cada camada faz só o que precisa (Controller → entrada/saída, Service → regras de negócio, Repository → banco). |

**✅ Em resumo:**

O usuário **nunca tem acesso à entidade ou ao banco**.  
Ele interage apenas com **DTOs**, que são versões seguras e controladas dos dados.