DTO’s

DTO significa **Data Transfer Object(Objeto de transferência de dados)**, a ideai por trás de utilizar um DTO é controlar **quais dados você vai expor** da sua entidade. Um DTO nada mais é do que um POJO (objeto simples do Java) com os **atributos espelhados da Entidade que ele representa:**

```java
@Entity
public class Topico {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String titulo;
    private String mensagem;
    private LocalDateTime dataCriacao = LocalDateTime.now();

    @Enumerated(EnumType.STRING)
    private StatusTopico status = StatusTopico.NAO_RESPONDIDO;

    @ManyToOne(fetch = FetchType.LAZY)
    private Usuario autor;

    @ManyToOne(fetch = FetchType.LAZY)
    private Curso curso;

    @OneToMany(mappedBy = "topico")
    private List<Resposta> respostas = new ArrayList<>();
```

```java
public class TopicoDto {

    private final Long id;
    private final String titulo;
    private final String mensagem;
    private final LocalDateTime dataCriacao;
    private final String nomeCurso;
```

Por exemplo, numa listagem geral da **entidade Tópico** eu posso **expor como resposta o DTO** apenas com os dados que eu quero mostrar:

```java
@GetMapping
public List<TopicoDto> buscarCollectionResourceTopicos(){
    List<Topico> topicos = topicoRepository.findAll();
    return TopicoDto.converter(topicos);
}
```

DTO’s podem ser múltiplos, se eu quiser expor um **Singleton Resource de um Tópico** com muito **mais detalhes**, eu posso ter um **DTO detalhado sobre o Tópico**. Se eu quiser receber um **payload pra cadastro de um tópico**, eu posso bindar essas informações em um **DTO que serve de formulário para o Tópico** e etc....:

```java
@GetMapping("/{id}")
public ResponseEntity<DetalhesTopicoDto> buscarSingletonTopico(@PathVariable Long id){
    Optional<Topico> topico = topicoRepository.buscarTopicoDetalhe(id);
    if(topico.isEmpty()){
        return ResponseEntity.notFound().build();
    }
    return ResponseEntity.ok().body(new DetalhesTopicoDto((topico.get())));
}
```

```java
@PostMapping
@Transactional
public ResponseEntity<TopicoDto>  cadastrarTopico(@RequestBody @Valid TopicoForm topicoForm,
                                                  UriComponentsBuilder uriBuilder){

    Topico topico = topicoForm.converterParaEntidade(cursoRepository);
    topicoRepository.save(topico);

    URI uri = uriBuilder.path("/topicos/{id}").buildAndExpand(topico.getId()).toUri();
    return ResponseEntity.created(uri).body(new TopicoDto(topico));
}
```