

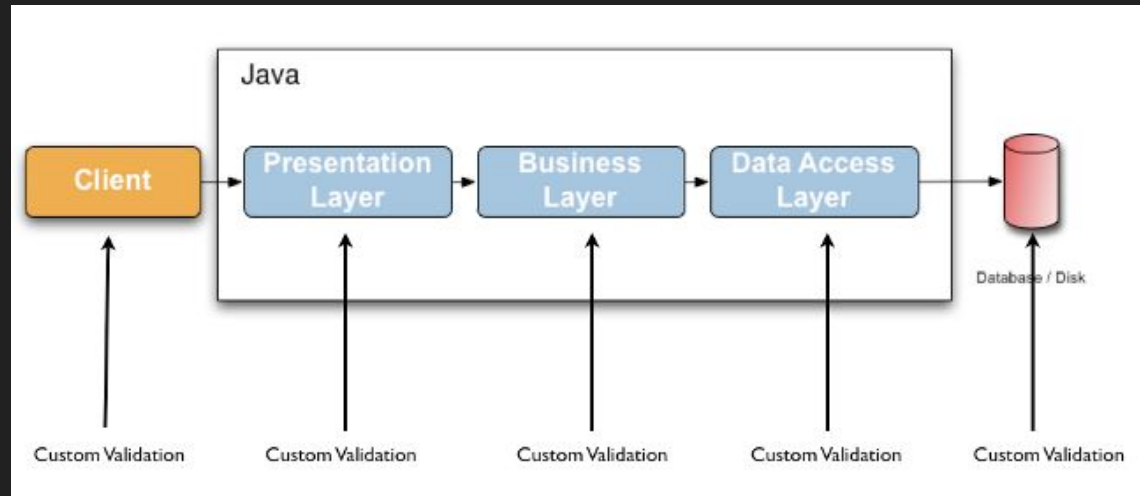
Java Advanced

**#05**

# BEAN VALIDATION

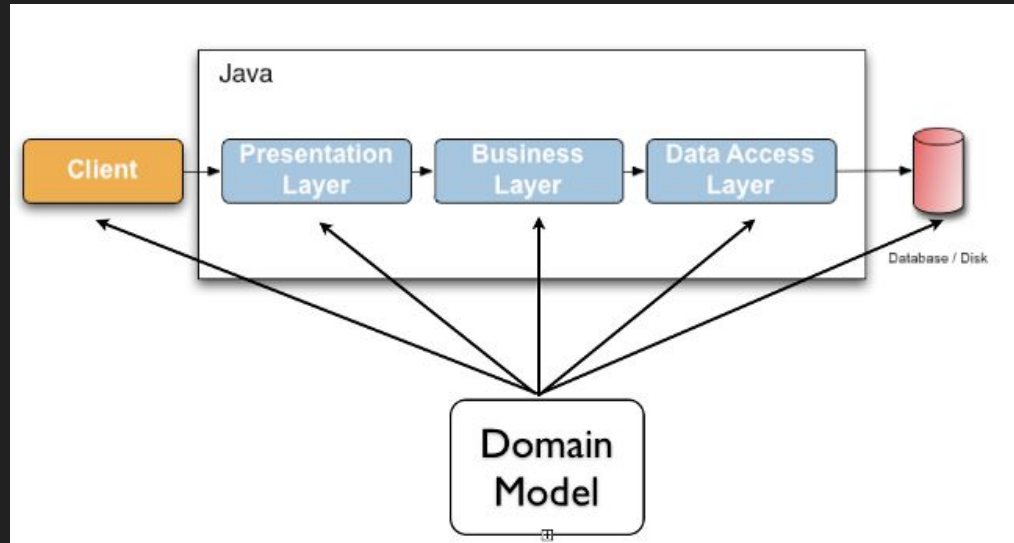
João Carlos Lima

# BEAN VALIDATION



fonte: <https://docs.jboss.org/hibernate/validator>

# BEAN VALIDATION



fonte: <https://docs.jboss.org/hibernate/validator>

# BEAN VALIDATION



1. Adicionar o Bean Validation no Starter
2. Anotar as regras no model
3. Anotar parâmetro com `@Valid`

# BEAN VALIDATION



```
@NotNull
```

```
@Size (min=3, max=30)
```

```
@Pattern(regex = "")
```

# BEAN VALIDATION

@NotNull

@NotEmpty

@NotBlank

```
String valor01 = null;  
String valor02 = "";  
String valor03 = " ";  
String valor04 = "valor";
```


```
String valor01 = null;  
String valor02 = "";  
String valor03 = " ";  
String valor04 = "valor";
```

```
String valor01 = null;  
String valor02 = "";  
String valor03 = " ";  
String valor04 = "valor";
```

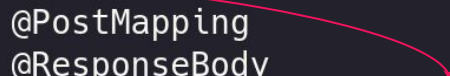
# BEAN VALIDATION



Aplica a  
validação nesse  
parâmetro



```
@PostMapping
@ResponseBody
public void create(@Valid User user) {
    System.out.println(user);
    repository.save(user);
}
```



# BEAN VALIDATION

representação  
dos dados  
validados



```
1 @PostMapping
2 public ResponseEntity<Despesa> create(@RequestBody @Valid Despesa despesa, BindingResult result){
3     log.info("cadastrando despesa " + despesa);
4     if (result.hasErrors()) {
5         log.error("erro ao cadastrar despesa " + result.getAllErrors()
6             .stream()
7             .map(e -> e.getDefaultMessage())
8             .reduce(" ", (a, b) -> a + ", " + b));
9     }
10    return ResponseEntity.badRequest().build();
11 }
```



# ! MENSAGENS DE VALIDAÇÃO



```
@Data
@Entity
public class User {

    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @NotBlank(message = "o campo nome é obrigatório")
    private String name;

    @Email(message = "informe um e-mail válido")
    private String email;

    @Size(min = 3, message = "informe uma senha com pelo menos 3 digitos")
    private String password;


}
```

# EXCEPTION HANDLER

```
1 @RestControllerAdvice
2 public class RestExceptionHandler {
3
4     Logger log = LoggerFactory.getLogger(RestExceptionHandler.class);
5
6     @ExceptionHandler({ConstraintViolationException.class})
7     public ResponseEntity<List<ValidationFieldError>> handleValidationException(ConstraintViolationException ex) {
8         List<ValidationFieldError> errors = new ArrayList<>();
9         ex.getConstraintViolations().forEach(
10             v -> {
11                 log.error(v.getMessage());
12                 errors.add(new ValidationFieldError(v.getPropertyPath().toString(), v.getMessage()));
13             })
14         ;
15         return ResponseEntity.badRequest().body(errors);
16     }
17 }
```

# RESPONSE STATUS EXCEPTION

```
1 Despesa despesaEncontrada = repository.findById(id)
2   .orElseThrow(() -> new ResponseStatusException(HttpStatus.NOT_FOUND, "Despesa não encontrada"));
3
```



```
1 @ExceptionHandler(ResponseStatusException.class)
2 public ResponseEntity<ResponseStatusError> handleResponseStatusException(ResponseStatusException ex){
3     ResponseStatusError error = new ResponseStatusError(
4         ex.getStatusCode().value(),
5         ex.getBody().getTitle(),
6         ex.getLocalizedMessage()
7     );
8
9     return ResponseEntity.status(ex.getStatusCode()).body(error);
10 }
```