



Spring Data JPA

O que é a para que Serve



O que é?

O Spring Data JPA é um **Framework** de **Persistência de Dados** lançado nos primeiros meses de 2011 com a finalidade de facilitar o processo de implementação das classes direcionadas a operações de **CRUD** (Create, Read, Update, Delete) e a criação de repositórios



O que é um Framework?

Framework uma ferramenta que ajuda o desenvolvedor a codificar melhor e mais rápido, unindo códigos comuns entre vários projetos de software, servindo como um suporte ou guia para a construção de algo.




O que significa o termo: Persistência de Dados ?

Persistência de Dados é um termo utilizado para dizer que os dados foram gravados em algum lugar e que não se perderão quando o computador for desligado. Ou seja, qualquer tipo de aplicação que grave um estado possível de ser recuperado, está fazendo persistência.



Para que serve o Spring Data JPA?

O Spring Data JPA libera o programador de ter que implementar as interfaces referentes aos repositórios e também deixa pré-implementado algumas funcionalidades como ordenação das consultas e paginação de registros. Já conta também com algumas classes e interfaces que reduzem a complexidade e a quantidade de código fonte.




Um projeto que não utiliza **Spring Data JPA**, geralmente tem uma interface com alguns métodos de consulta definidos que posteriormente são codificados em uma classe concreta. Quando se faz uso de **Spring Data**, basta incluir a assinatura do método de pesquisa na interface e adicionar a esta assinatura, uma anotação do tipo **@Query**, com a respectiva consulta no formato **JPQL**. A implementação deste método será realizada automaticamente pelo **Spring Data** em tempo de execução.



Alguns exemplos de utilização

No **Spring Data** já existem todos os métodos necessários para fazer um **CRUD**. Mas antes, temos construir a entidade:


```
1  @Entity
2  public class Produto implements Serializable {
3
4      private static final long serialVersionUID = 1L;
5
6      @Id
7      @GeneratedValue
8      private Long id;
9
10
11     private String nome;
12
13     private String descricao;
14
15     private boolean ativo;
16
17     @Temporal(TemporalType.TIMESTAMP)
18     private Date cadastro;
19
20
21     private int quantidade;
22
23     // getters e setters omitidos
24
25 }
```

Exemplo de como criar o repositório para a entidade Produto e nome **Produtos**:


```
1 | public interface Produtos extends JpaRepository<Produto, Long> {  
2 |  
3 | }
```

A implementação da interface **Produtos**, será disponibilizada pelo próprio **Spring Data JPA** em tempo de execução.



Exemplo de **Ordenação de Registros** com um método recebendo o tipo Sort como parâmetro e permitindo receber da requisição, a propriedade pela qual a consulta será ordenada e em qual direção (ascendente ou descendente):


```
1  @GetMapping
2  public List<Produto> pesquisar(
3      @RequestParam(defaultValue = "nome") String ordenacao,
4      @RequestParam(defaultValue = "ASC") Sort.Direction direcao) {
5      return produtos.findAll(new Sort(direcao, ordenacao));
6  }
```



A facilidade mais interessante desse framework, é a criação de consultas a partir da **Assinatura do Método**. Sendo assim, exemplificaremos uma consulta para retornar o produto pelo nome:

```
1 public interface Produtos extends JpaRepository<Produto, Long> {  
2  
3     Produto findByName(String nome);  
4  
5 }
```

Observe também que não necessário dar uma implementação para esse método *findByName*



O recurso de criar consultas pela **assinatura do método** é bastante útil mas não resolverá todos os problemas. Eventualmente, ainda será necessário executar consultas com JPQL. Os desenvolvedores do **Spring Data JPA** sabendo disso, incluíram a anotação **@Query** para os usuários do Framework



Segue um exemplo que também não precisa dar a implementação para o método:

```
1 public interface Produtos extends JpaRepository<Produto, Long> {  
2  
3     ...  
4  
5     @Query("from Produto where nome like concat(?1, '%')")  
6     List<Produto> pesquisarProdutos(String nome);  
7  
8 }
```

Repare que a consulta e o método recebem um parâmetro para que o **Spring Data** possa utilizá-lo na hora a busca. O nome **pesquisarProdutos**, não precisa seguir uma regra pois a consulta válida é a que está na anotação.

src > main > java > com > example > demo3 > model > Aluno.java > Aluno

```
1  package com.example.demo3.model;
2
3  import javax.persistence.Entity;
4  import javax.persistence.GeneratedValue;
5  import javax.persistence.GenerationType;
6  import javax.persistence.Id;
7
8  @Entity
9  public class Aluno {
10     @Id
11     @GeneratedValue(strategy = GenerationType.AUTO)
12     private Integer id;
13     private String firstName;
14     private String lastName;
15
16     public Integer getId() {
17         return id;
18     }
19     public void setId(Integer id) {
20         this.id = id;
21     }
22     public String getFirstName() {
23         return firstName;
24     }
25     public void setFirstName(String firstName) {
26         this.firstName = firstName;
27     }
28     public String getLastName() {
29         return lastName;
30     }
31     public void setLastName(String lastName) {
32         this.lastName = lastName;
33     }
34     public Aluno(String firstName, String lastName) {
35         this.firstName = firstName;
36         this.lastName = lastName;
37     }
38     public Aluno() {
39
40     }
41     @Override
42     public String toString() {
43         return "Aluno [firstName=" + firstName + ", id=" + id + ", lastName=" + lastName + "];";
44     }
45
46 }
```

Aluno.java • AlunoRepository.java X

src > main > java > com > example > demo3 > repository > AlunoRepository.java > ...

```
1 package com.example.demo3.repository;
2
3 import com.example.demo3.model.Aluno;
4
5 import org.springframework.data.repository.CrudRepository;
6
7 // This will be AUTO IMPLEMENTED by Spring into a Bean called AlunoRepository
8 // CRUD refers Create, Read, Update, Delete
9
10 public interface AlunoRepository extends CrudRepository<Aluno, Integer> {
11
12     Aluno findByLastName(String lastName);
13
14     Aluno findById(long id);
15
16 }
17
```

```
src > main > java > com > example > demo3 > controller > AlunoController.java > ...
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Controller;
6 import org.springframework.ui.Model;
7 import org.springframework.web.bind.annotation.GetMapping;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RequestParam;
11 import org.springframework.web.bind.annotation.ResponseBody;
12 import org.springframework.web.servlet.view.RedirectView;
13 @Controller
14 @RequestMapping("/")
15 public class AlunoController {
16     @Autowired // Which is auto-generated by Spring, we will use it to handle the data
17     private AlunoRepository alunoRepository;
18     @GetMapping(path = "/index")
19     public String showPageNow(Model model, @RequestParam(defaultValue = "0") int page) {
20         model.addAttribute("alunos", alunoRepository.findAll());
21         return "index";
22     }
23     @GetMapping(path = "/")
24     public String showPage(Model model) {
25         model.addAttribute("alunos", alunoRepository.findAll());
26         return "index";
27     }
28     @GetMapping(path = "/all")
29     public @ResponseBody RedirectView Aluno(Model model) {
30         model.addAttribute("alunos", alunoRepository.findAll());
31         // return "index";
32         // return new ModelAndView("forward:/index", model);
33         return new RedirectView("index");
34     }
35     @PostMapping(path = "/add")
36     public @ResponseBody RedirectView addNewAluno(@RequestParam String firstName, @RequestParam String lastName) {
37
38         Aluno a = new Aluno();
39         a.setFirstName(firstName);
40         a.setLastName(lastName);
41         alunoRepository.save(a);
42         // return "redirect:/";
43         // return "redirect:/index";
44         return new RedirectView("index");
45     }
46     @PostMapping(path = "/delete")
47     public @ResponseBody RedirectView delAluno(Integer id) {
48         alunoRepository.deleteById(id);
49         // return "redirect:/index";
50         return new RedirectView("index");
51     }
52 }
```


AccessingDataMysqlApplication.java X

src > main > java > com > example > demo3 > AccessingDataMysqlApplication.java > AccessingDataMysqlApplication

```
1  package com.example.demo3;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.context.annotation.ComponentScan;
6  import org.springframework.context.annotation.Configuration;
7
8  @Configuration
9  @ComponentScan("package com.example.demo3.controller")
10 @SpringBootApplication
11 public class AccessingDataMysqlApplication {
12
13     Run | Debug
14     public static void main(String[] args) {
15         SpringApplication.run(AccessingDataMysqlApplication.class, args);
16     }
17 }
```

 WebConfig.java ×

src > main > java > com > example > demo3 >  WebConfig.java >  WebConfig

```
1  package com.example.demo3;
2
3  import org.springframework.context.annotation.ComponentScan;
4  import org.springframework.context.annotation.Configuration;
5
6  @Configuration
7  @ComponentScan("package com.example.demo3.controller")
8  public class WebConfig {
9      
10  }
```

src > main > resources > application.properties

```
1  ## MySQL
2  #spring.datasource.url=jdbc:mysql://localhost:3306/springdatajpa?useTimezone=true&serverTimezone=UTC
3  #spring.datasource.username=root
4  #spring.datasource.password=entrar
5
6  #`hibernate_sequence` doesn't exist
7  #spring.jpa.hibernate.use-new-id-generator-mappings=false
8
9  # drop n create table, good for testing, comment this in production
10 # spring.jpa.hibernate.ddl-auto=create
11
12 #2 versão
13 spring.jpa.hibernate.ddl-auto=update
14 spring.datasource.url=jdbc:mysql://localhost:3306/alunoslp2?useTimezone=true&serverTimezone=UTC
15 spring.datasource.username=root
16 spring.datasource.password=entrar
17
18
19
20
21 # /* sql      code */
22
23
24 #      create database alunoslp2;
25
26 #      use alunoslp2;
27
28 #      show tables;
29
30 #      select * from aluno;
31
32 #      drop table aluno;
33
34 #
```

```
30 </head>
31
32 <body>
33   <div class="container .container-fluid">
34     <section class="container .container-fluid">
35       <div style="margin: 2em; margin-bottom: 4em;">
36         <h1>Linguagem de Programação 2<span></h1>
37       </div>
38     </section>
39     <section class="container .container-fluid">
40       <table class="table table-striped">
41         <thead class="thead-dark">
42           <th scope="col">#</th>
43           <th scope="col">Nome</th>
44           <th scope="col">Sobrenome</th>
45           <th scope="col"><span style="color: black;"></span></th>
46         </thead>
47         <tbody>
48           <tr th:if="${alunos.empty}">
49             <td colspan="2">Cadê os alunos...? </td>
50             <td colspan="2"></td>
51           </tr>
52           <tr th:each="alunos : ${alunos}">
53             <td><span th:text="${alunos.id}"> id</span></td>
54             <td><span th:text="${alunos.firstName}"> id</span></td>
55             <td><span th:text="${alunos.lastName}"> id</span></td>
56             <td>
57               <form action="/delete" method="post">
58                 <input type="text" name="id" th:value="${alunos.id}" style="display:none;">
59                 <input class="btn btn-primary" type="submit" value="delete">
60               </form>
61             </td>
62           </tr>
63         </tbody>
64       </table>
65     </section>
66     <section class="container .container-fluid">
67       <div class="form-group">
68         <form action="/add" method="post">
69           <div><input type="text" class="form-control" name="firstName" placeholder="First name"><br></div>
70           <div><input type="text" class="form-control" name="lastName" placeholder="Last name"><br></div>
71           <div><input type="submit" value="add" class="btn btn-primary"
72             style="width: 20%; margin: auto; margin-top: 1em;"></div>
73         </form>
74       </div>
75     </section>
76
77 </div>
```



File Edit Selection View Go Debug Terminal Help

EXPLORER

OPEN EDITORS

JAVA DEPENDENCIES

DEMO3

- > .mvn
- > .vscode
- > src
 - main
 - java
 - com
 - example
 - demo3
 - controller
 - AlunoController.java
 - model
 - Aluno.java
 - repository
 - AlunoRepository.java
 - view
 - AccessingDataMysqlApplication.java
 - WebConfig.java
 - resources
 - static
 - MyWebApplicationInitializer.java
 - templates
 - index.html
 - application.properties
 - web.xml
 - .gitignore
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

TOMCAT SERVERS

0 2 Connect Live Share 05 - Spring Data JPA - D2 (WebApp MyS

SQL File 3* Administration - Startup / Shutdo... SQL File 3* x

Limit to 1000 rows

```
1
2
3  /* sql code */
4  create database alunoslp2;
5
6
7  /* sql code */
8  use alunoslp2;
9
10
11
12  select * from aluno;
13
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

Filter Rows: Export: Wrap Cell Contents: IS

Tables_in_alunoslp2

- aluno
- hibernate_sequence

Result 1 x Read Only Context Help Snippets

Output

#	Time	Action	Message	Duration / Fetch
41	15:35:07	drop table aluno	0 row(s) affected	0.031 sec
42	15:35:13	create database alunosdb	Error Code: 1007. Can't create database 'alunosdb'; database exists	0.000 sec
43	15:35:15	select * from aluno LIMIT 0, 1000	Error Code: 1146. Table 'alunoslp2.aluno' doesn't exist	0.015 sec
44	15:35:48	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
45	16:29:25	use alunoslp2	0 row(s) affected	0.000 sec
46	16:29:27	show tables	2 row(s) returned	0.000 sec / 0.000 sec

Linguagem de Programação 2

#	Nome	Sobrenome
Cadê os alunos...?		
<input type="text"/>		
<input type="text"/>		
<input type="button" value="add"/>		

SQL File 3* Administration - Startup / Shutdo... SQL File 3* x

Limit to 1000 rows

```
1
2
3 /* sql code */
4 create database alunoslp2;
5
6
7 /* sql code */
8 use alunoslp2;
9
10 show tables;
11
12 aluno
13
```

Result Grid

id	first_name	last_name
26	Rafael	Lapa Valgas

aluno 3 x

Output

#	Time	Action	Message	Duration / Fetch
43	15:35:15	select * from aluno LIMIT 0, 1000	Error Code: 1146, Table 'alunoslp2.aluno' doesn't exist	0.015 sec
44	15:35:48	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
45	16:29:25	use alunoslp2	0 row(s) affected	0.000 sec
46	16:29:27	show tables	2 row(s) returned	0.000 sec / 0.000 sec
47	16:30:39	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
48	16:30:55	select * from aluno LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec

Linguagem de Programação 2

#	Nome	Sobrenome	
26	Rafael	Lapa Valgas	<button>delete</button>
<input type="text" value="First name"/>			
<input type="text" value="Last name"/>			
<button>add</button>			

SQL File 3* Administration - Startup / Shudo... SQL File 3* x

Limit to 1000 rows

```
1
2
3 /* sql code */
4 create database alunoslp2;
5
6
7 /* sql code */
8 use alunoslp2;
9
10 show tables;
11
12 aluno
13
```

Result Grid

id	first_name	last_name
26	Rafael	Lapa Valgas
27	Patrícia	Kraus
28	Diego	Silva
29	Lucas	Medeiros
30	Osmar	Conha
31	Help	Me
32	Fernando	Abreu

Form Editor

Field Types

Query Data

Execution Plan

Apply Context Help Snippets

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 44	15:35:48	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
✓ 45	16:29:25	use alunoslp2	0 row(s) affected	0.000 sec
✓ 46	16:29:27	show tables	2 row(s) returned	0.000 sec / 0.000 sec
✓ 47	16:30:39	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
✓ 48	16:30:55	select * from aluno LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
✓ 49	16:33:12	select * from aluno LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec

Linguagem de Programação 2

#	Nome	Sobrenome	
26	Rafael	Lapa Valgas	<button>delete</button>
27	Patrícia	Kraus	<button>delete</button>
28	Diego	Silva	<button>delete</button>
29	Lucas	Medeiros	<button>delete</button>
30	Osmar	Conha	<button>delete</button>
31	Help	Me	<button>delete</button>
32	Fernando	Abreu	<button>delete</button>

add

 **Captura de tela salva**
A captura de tela foi adicionada ao OneDrive.
OneDrive

SQL File 3* Administration - Startup / Shudo... SQL File 3* x

Limit to 1000 rows

```
1
2
3 /* sql code */
4 create database alunoslp2;
5
6
7 /* sql code */
8 use alunoslp2;
9
10 show tables;
11
12 aluno
13
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

	id	first_name	last_name
▶	26	Rafael	Lapa Valgas
*	27	Patricia	Kraus

Form Editor

Field Types

Query Data

Execution Plan

aluno 5 x

Apply Context Help Snippets

Output

#	Time	Action	Message	Duration / Fetch
✓ 45	16:29:25	use alunoslp2	0 row(s) affected	0.000 sec
✓ 46	16:29:27	show tables	2 row(s) returned	0.000 sec / 0.000 sec
✓ 47	16:30:39	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
✓ 48	16:30:55	select * from aluno LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
✓ 49	16:33:12	select * from aluno LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
✓ 50	16:34:26	select * from aluno LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Linguagem de Programação 2

#	Nome	Sobrenome	
26	Rafael	Lapa Valgas	<button>delete</button>
27	Patricia	Kraus	<button>delete</button>

add

SQL File 3* Administration - Startup / Shutdo... SQL File 3* x

Limit to 1000 rows

```
1
2
3 /* sql code */
4 create database alunoslp2;
5
6
7 /* sql code */
8 use alunoslp2;
9
10 show tables;
11
12 -- aluno
13
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

id	first_name	last_name
26	Rafael	Lapa Valgas
33	José	Pedrão

aluno 6 x

Output

#	Time	Action	Message	Duration / Fetch
46	16:29:27	show tables	2 row(s) returned	0.000 sec / 0.000 sec
47	16:30:39	select * from aluno LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
48	16:30:55	select * from aluno LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
49	16:33:12	select * from aluno LIMIT 0, 1000	7 row(s) returned	0.000 sec / 0.000 sec
50	16:34:26	select * from aluno LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec
51	16:35:26	select * from aluno LIMIT 0, 1000	2 row(s) returned	0.000 sec / 0.000 sec

Linguagem de Programação 2

#	Nome	Sobrenome	
26	Rafael	Lapa Valgas	<button>delete</button>
33	José	Pedrão	<button>delete</button>

add



VLW GALERA!! ;)



Spring Data JPA

O que é a para que Serve

