

VRaptor 3

Felipe Sobreira Cassimiro

Criando o projeto

- Blank project
- Projeto padrão + JARs
- Maven project
- Scaffold *
 - Utiliza o ant para gerenciar a build
 - Utiliza o ivy para gerenciar as dependências

O Projeto?

The image shows two side-by-side browser windows from a web application running on localhost:8080.

Left Window: Formulário
URL: `http://localhost:8080/livraria-admin/livros/formulario`
Title: **Formulário de cadastro de livros**
Fields:

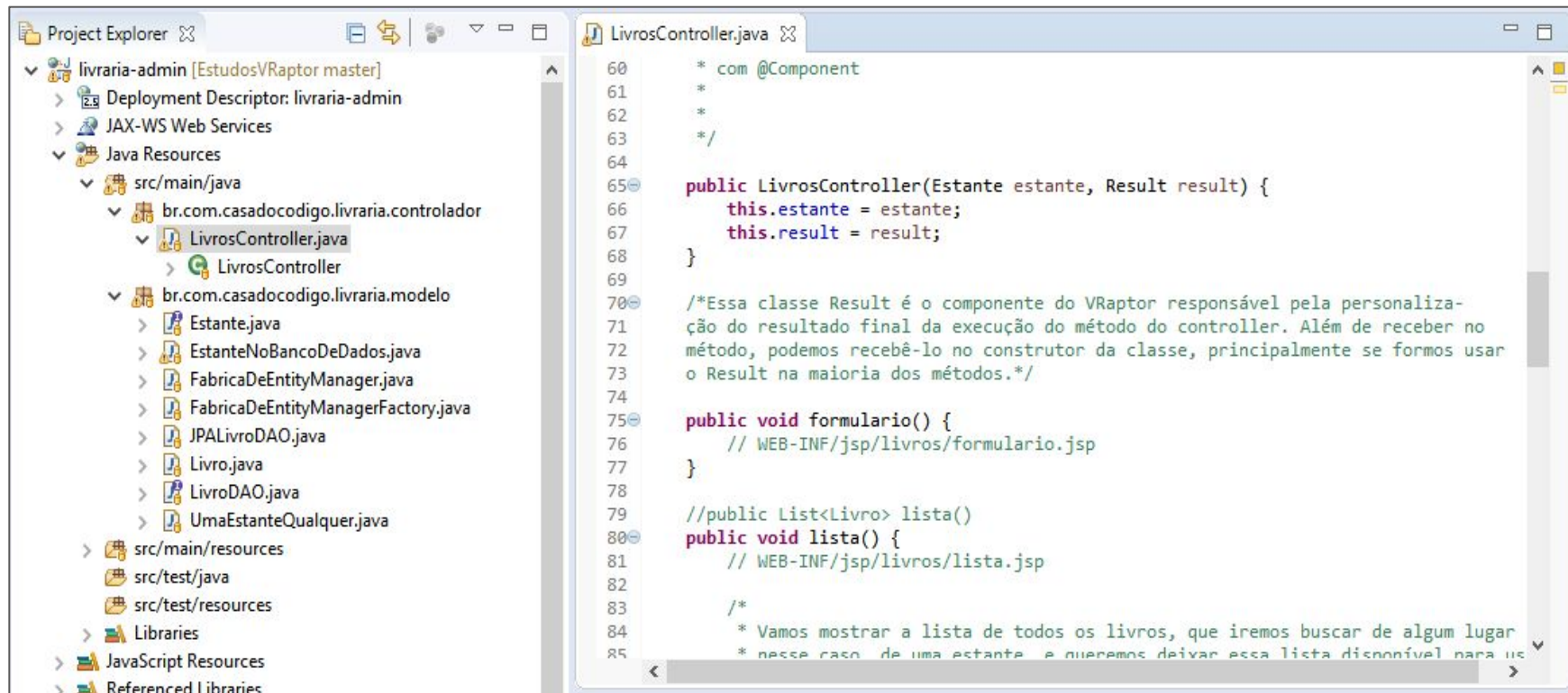
- Titulo:
- Descrição:
- ISBN:
- Preço:
- Data de publicacao:

Button:

Right Window: Lista de livros
URL: `http://localhost:8080/livraria-admin/livros/lista`
Message: Livro salvo com sucesso!
Title: **Lista de livros**
List:

- VRaptor 3- Uma introdução [Modificar](#)
- JPA- Uma introdução [Modificar](#)
- Spring- mais introdução [Modificar](#)

O Projeto?



The image shows a screenshot of an IDE with two panels. The left panel displays the Project Explorer, and the right panel displays the source code of the `LivrosController.java` file.

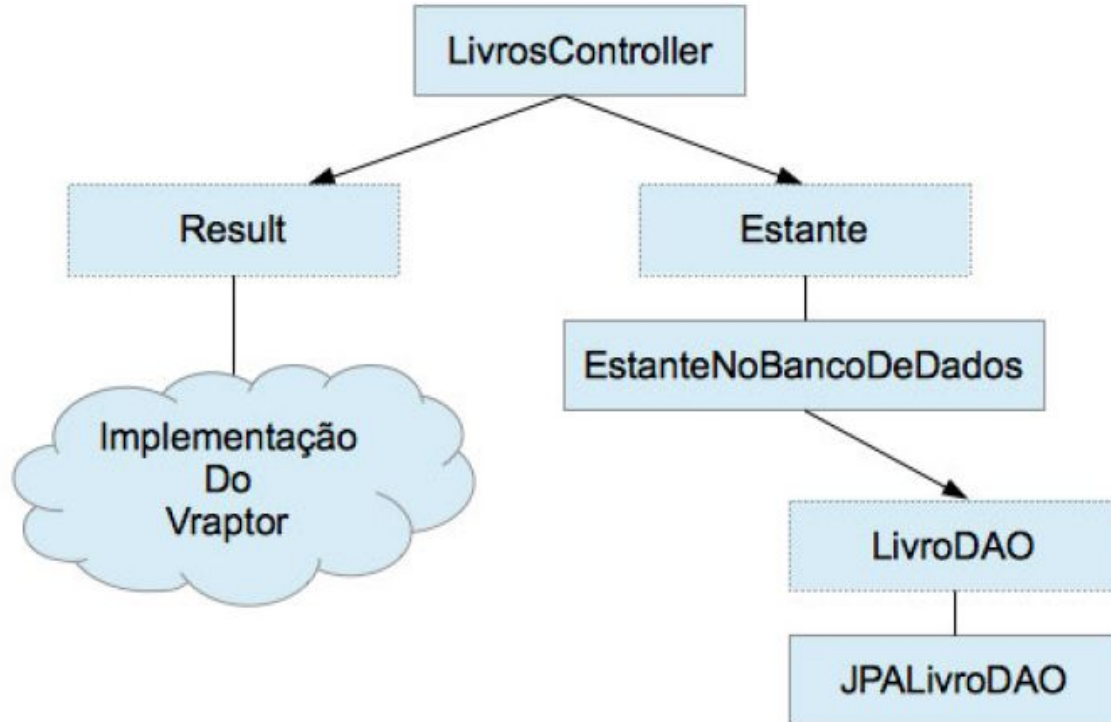
Project Explorer (Left Panel):

- livraria-admin [EstudosVRaptor master]
 - Deployment Descriptor: livraria-admin
 - JAX-WS Web Services
 - Java Resources
 - src/main/java
 - br.com.casadocodigo.livraria.controlador
 - LivrosController.java (selected)
 - LivrosController
 - br.com.casadocodigo.livraria.modelo
 - Estante.java
 - EstanteNoBancoDeDados.java
 - FabricaDeEntityManager.java
 - FabricaDeEntityManagerFactory.java
 - JPALivroDAO.java
 - Livro.java
 - LivroDAO.java
 - UmaEstanteQualquer.java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - Libraries
 - JavaScript Resources
 - Referenced Libraries

LivrosController.java (Right Panel):

```
60  * com @Component
61  *
62  *
63  */
64
65  public LivrosController(Estante estante, Result result) {
66      this.estante = estante;
67      this.result = result;
68  }
69
70  /*Essa classe Result é o componente do VRaptor responsável pela personaliza-
71  ção do resultado final da execução do método do controller. Além de receber no
72  método, podemos recebê-lo no construtor da classe, principalmente se formos usar
73  o Result na maioria dos métodos.*/
74
75  public void formulario() {
76      // WEB-INF/jsp/livros/formulario.jsp
77  }
78
79  //public List<Livro> lista()
80  public void lista() {
81      // WEB-INF/jsp/livros/lista.jsp
82
83      /*
84      * Vamos mostrar a lista de todos os livros, que iremos buscar de algum lugar
85      * nesse caso de uma estante e queremos deixar essa lista disponível para us
```

Arquitetura de dependências



@Resource - Um controlador

- Uma classe para ser determinada como um controlador, precisa ser anotada com @Resource.
- Métodos públicos desse controlador ficarão disponíveis para receber requisições.
- Os parâmetros utilizados nos métodos serão instanciados com os valores da requisição.
- O construtor é utilizado para injetar dependências.
- O retorno de um método é exportado para a view.
- Path + Métodos HTTP

@Resource - Um controlador

```
@Resource
public class ClienteController {

    @Path("/cliente")
    @Post
    public void adiciona(Cliente cliente) {

    }

}
```

```
@Resource
@Path("/clientes")
public class ClienteController {

    //URI: /clientes/lista
    public void lista() {...}

    //URI: /clientes/salva
    @Path("salva")
    public void adiciona() {...}

    //URI: /clientes/todosOsClientes
    @Path("/todosOsClientes")
    public void listaTudo() {...}

}
```

```
@Resource
public class ClienteController {

    @Post("/cliente")
    public void adiciona(Cliente cliente) {

    }

    @Path("/")
    public List<Cliente> lista() {
        return ...
    }

    @Get("/cliente")
    public Cliente visualiza(Cliente cliente) {
        return ...
    }

    @Delete("/cliente")
    public void remove(Cliente cliente) {
        ...
    }

    @Put("/cliente")
    public void atualiza(Cliente cliente) {
        ...
    }

}
```

@Component - Uma dependência

- Classes que são utilizadas como dependências por outras classes.
- A boa prática determina criar uma interface para seus componentes.

```
@Component
public class ClienteDao {

    private final Session session;
    public ClienteDao(Session session) {
        this.session = session;
    }

    public void adiciona(Cliente cliente) {
        session.save(cliente);
    }

}
```


Outros conceitos

- @Entity - JPA
- Result - Controle dos resultados

```
public void edita(String isbn) {  
    Livro livroEncontrado = this.estante.buscaPorIsbn(isbn);  
    result.include(livroEncontrado);  
  
    result.of(this).formulario();  
}
```

Considerações

- Tempo desperdiçado com configuração e bugs.
- Falha na leitura.
- Utilizar da persistência em memória.
- Utilizar documentação para desenvolvimento em “passos”.

01	VRaptor3 - O guia inicial de 1 minuto
 	
02	VRaptor3 - O guia inicial de 10 minutos
 	
03	Resources-Rest
 	
04	Componentes
 	
05	Conversores
 	
06	Interceptadores
 	
07	Validação
 	
08	View e Ajax
 	
09	Injeção de dependências
 	
10	Download e Upload
 	
11	Componentes utilitários opcionais
 	

Referências

- CAVALCANTI, L. VRaptor: Desenvolvimento ágil para web com java.
- VRaptor 3 Documentação. Disponível em: <http://vraptor3.vraptor.org/pt/docs>.