TEORIA DO CODIGO JAVA

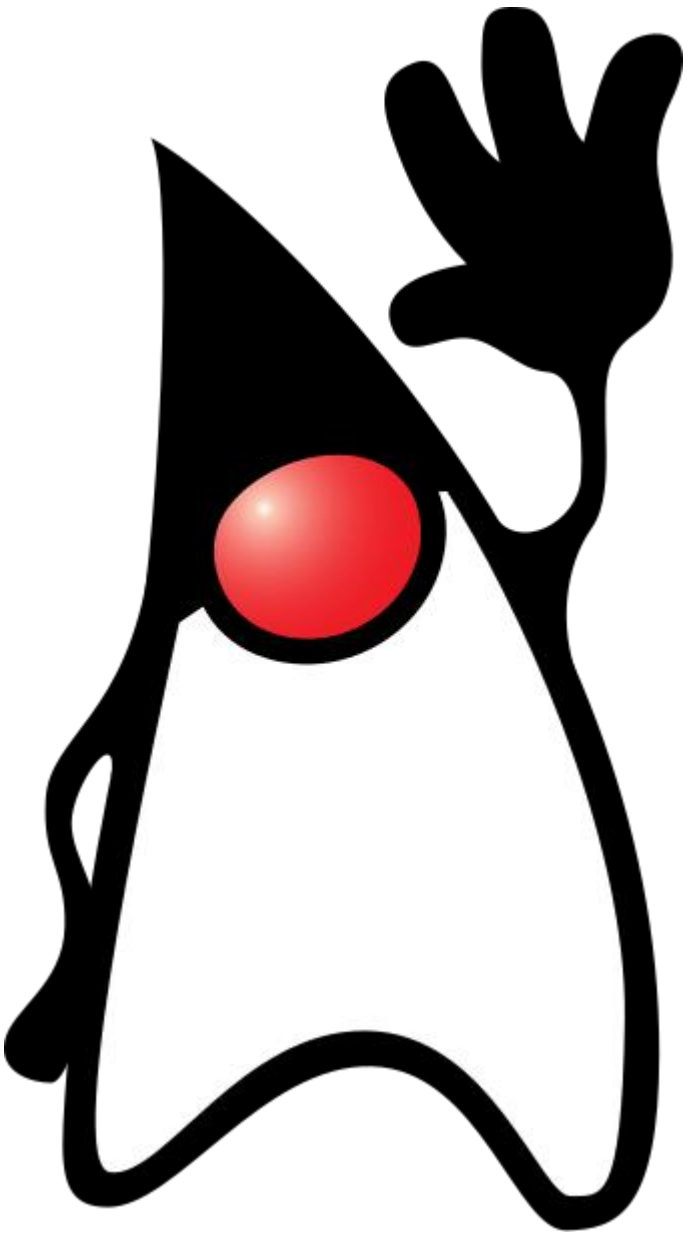
A SINGULARIDADE DA MULTIPLATAFORMA

LUAN SILVA

Java: Uma Jornada Multiplataforma

Explore exemplos de aplicações Java.

O Java é uma linguagem de programação que está em todos os lugares: na web, nos aplicativos de celular, nos programas de desktop e muito mais. Com uma sintaxe clara e uma comunidade ativa, ele oferece ferramentas poderosas para desenvolver praticamente qualquer tipo de software. Neste ebook, vamos ver onde o Java brilha e aprender com exemplos simples e diretos!



01

Desenvolvimento Web com Java

Aprenda como o Java cria sites dinâmicos e interativos usando tecnologias como Servlets e JSP. Um ótimo ponto de partida para quem quer começar no mundo web.

Desenvolvimento Web com Java

O Java é muito usado para criar sites dinâmicos. Usando Servlets e JSP, podemos criar páginas que mudam de acordo com a interação do usuário.

Servlet que exibe uma saudação simples

```java
@WebServlet("/saudacao")
public class SaudacaoServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        resp.getWriter().println("Olá, visitante!");
    }
}
```

Aqui, criamos um servlet que exibe a mensagem "Olá, visitante!" sempre que alguém acessar o link ```/saudacao```.

Desenvolvimento Web com Java

Além de exibir páginas, um Servlet pode receber dados enviados por um formulário HTML. Isso é muito usado para cadastros e buscas.

Servlet que lê o nome de um formulário

```java
@WebServlet("/recebe-nome")
public class FormularioServlet extends HttpServlet {
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        String nome = req.getParameter("nome");
        resp.getWriter().println("Olá, " + nome + "!");
    }
}
```

Assim, ao enviar o formulário com o campo ```nome```, o servlet responde com uma saudação personalizada.

02

Aplicações Desktop com Swing

Descubra como criar programas de desktop com interfaces gráficas amigáveis e ricas usando o Swing.

Aplicações Desktop com Swing

Para criar programas que rodam no computador, o Java tem o Swing, uma biblioteca para criar janelas, botões e campos de texto de forma simples.

Janela com um botão

```java
import javax.swing.*;

public class AppDesktop {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Meu App");
        JButton botao = new JButton("Clique aqui");
        frame.add(botao);
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

Esse código cria uma janela com um botão. Ao clicar, podemos programar novas ações!

Aplicações Desktop com Swing

Com o Swing, podemos exibir mensagens usando caixas de diálogo para interações simples.

Exibindo Mensagens para o Usuário

```java
import javax.swing.*;

public class AlertaSimples {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null, "Bem-vindo ao meu app!");
    }
}
```

Essa janela exibe a mensagem "Bem-vindo ao meu app!" em um pop-up.

03

Java no Android

Veja como o Java ainda está presente no desenvolvimento de aplicativos móveis para Android, desde a criação de telas até a integração com o sistema.

Java no Android

O Java também é a base de muitos aplicativos para Android. Com ele, podemos criar telas e interagir com os recursos do celular.

Tela de Boas-vindas no Android

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Essa Activity carrega o layout da tela inicial de um aplicativo Android.

Java no Android

Um app Android fica mais interativo quando adicionamos botões e tratamos seus cliques.

Botão que exibe uma mensagem ao ser clicado

```java
Button meuBotao = findViewById(R.id.botao);
meuBotao.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, "Botão clicado!", Toast.LENGTH_SHORT).show();
    }
});
```

Esse código mostra um alerta (Toast) ao clicar no botão.

04

APIS E MICROSERVICES COM SPRING BOOT

Explore como o Java facilita a criação de APIs e microsserviços robustos usando o poderoso framework Spring Boot.

APIs e Microservices com Spring Boot

Para criar APIs (interfaces que permitem a comunicação entre sistemas), o Spring Boot é a ferramenta ideal no Java. Ele facilita a criação de microsserviços, que são pequenos pedaços de software que se comunicam entre si.

API simples que retorna uma mensagem

```java
@RestController
public class SaudacaoController {
    @GetMapping("/api/saudacao")
    public String saudacao() {
        return "Bem-vindo à API!";
    }
}
```

Essa API responde "Bem-vindo à API!" quando acessamos ```/api/saudação```.

APIs e Microservices com Spring Boot

O Spring Boot permite que as respostas da API sejam em JSON, um formato muito usado na comunicação entre sistemas.

API que retorna um objeto JSON

```java
@RestController
public class ProdutoController {
    @GetMapping("/api/produto")
    public Produto produto() {
        return new Produto("Notebook", 2500.00);
    }
}

class Produto {
    public String nome;
    public double preco;

    public Produto(String nome, double preco) {
        this.nome = nome;
        this.preco = preco;
    }
}
```

Essa API retorna um objeto JSON com nome e preço de um produto.

05

INTEGRAÇÃO COM BANCO DE DADOS: JDBC

Entenda como conectar suas aplicações Java a bancos de dados usando JDBC e realizar operações como consultas e atualizações.

Integração com Banco de Dados

Com o JDBC, podemos conectar aplicações Java a bancos de dados como MySQL ou PostgreSQL. Isso é essencial para guardar e buscar informações.

Buscando nomes em um banco de dados MySQL

```java
import java.sql.*;

public class ConexaoBanco {
    public static void main(String[] args) throws SQLException {
        Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost/meubanco", "usuario", "senha");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT nome FROM clientes");
        while (rs.next()) {
            System.out.println(rs.getString("nome"));
        }
        conn.close();
    }
}
```

Esse programa busca e exibe nomes de clientes salvos em um banco de dados.

Integração com Banco de Dados

Além de buscar dados, também podemos inserir novas informações no banco.

Inserindo um novo cliente no MySQL

```java
import java.sql.*;

public class InserirCliente {
    public static void main(String[] args) throws SQLException {
        Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost/meubanco", "usuario", "senha");
        String sql = "INSERT INTO clientes (nome) VALUES (?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, "João");
        stmt.executeUpdate();
        System.out.println("Cliente inserido!");
        conn.close();
    }
}
```

Esse programa insere um novo cliente chamado "João" no banco.

AGRADECIMENTOS

OBRIGADO POR LER ATÉ AQUI

O conteúdo desse ebook foi gerado integralmente por ia e auxiliado com ferramentas de edição para fins didáticos e não comerciais.

Fique a vontade para compartilhar com amigos e familiares. O ebook não traz uma introdução ao mundo da programação com Java, apenas demonstrar a usabilidade dessa linguagem em diferentes plataformas.

Repositório GitHub