Introdução à Programação Orientada a Objetos (POO)

UNIVERSIDADE DO OESTE DE SANTA CATARINA - UNOESC

Ciencia da Computacao

Prof Leandro Otavio Cordova Vieira

Objetivos da Aula

Compreender a origem e os fundamentos da POO

Exploraremos o surgimento histórico e os conceitos básicos que fundamentam este paradigma de programação.

Identificar a importância da POO no desenvolvimento moderno

Analisaremos por que a POO se tornou dominante e quais benefícios ela traz para projetos contemporâneos.

Analisar a evolução da POO na linguagem PHP

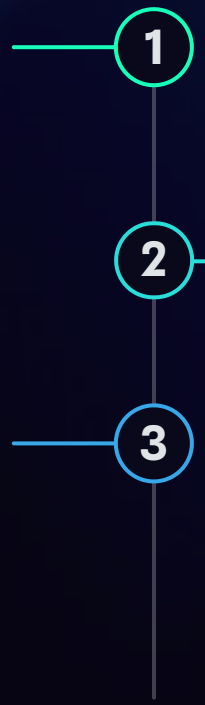
Veremos como o PHP incorporou e aprimorou os recursos de orientação a objetos ao longo de suas versões.

Pergunta Disparadora

Você já reutilizou algum trecho de código em outro projeto? Como?

Pense em suas experiências com reaproveitamento de código e como isso poderia ser melhorado através de técnicas estruturadas.

Panorama Histórico



Décadas de 60-70

Surgimento da POO com as linguagens Simula e Smalltalk, que introduziram os primeiros conceitos de classes e objetos.

1980

Desenvolvimento do C++, que popularizou a POO ao adicionar recursos orientados a objetos à linguagem C.

1995

Lançamento do Java e PHP, linguagens que ajudaram a consolidar o paradigma da orientação a objetos no desenvolvimento de software.

A evolução da POO marcou uma mudança fundamental na forma como pensamos e estruturamos o código.

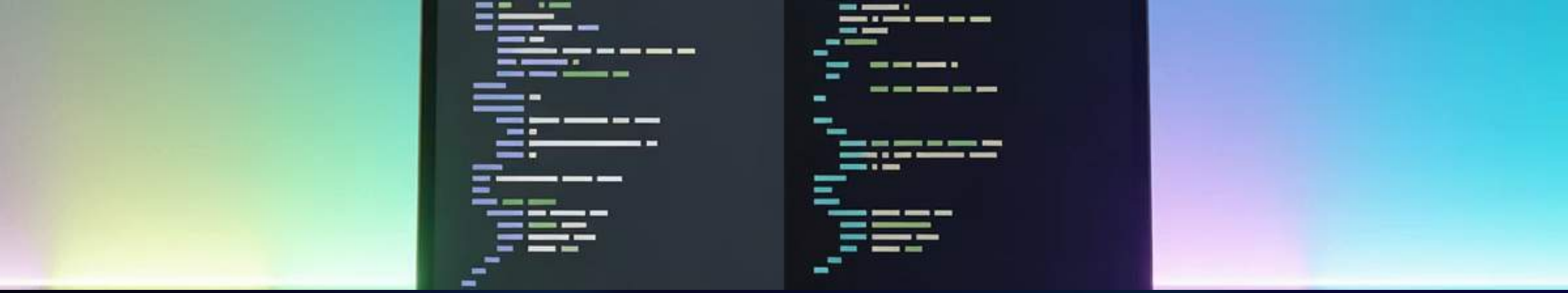
Definição de POO

Paradigma que organiza o código em torno de objetos

Objetos são entidades que combinam:

- **Dados (atributos)**: características do objeto
- **Comportamentos (métodos)**: ações que o objeto pode realizar

Esta abordagem permite modelar o software de forma mais próxima ao mundo real.



Paradigmas Comparados

Programação Procedural	Programação Orientada a Objetos
• Foco em funções e procedimentos	• Foco em objetos que encapsulam dados e comportamentos
• Dados e comportamentos separados	• Métodos associados a objetos específicos
• Variáveis globais frequentes	• Atributos encapsulados em classes
• Código sequencial linear	• Estrutura modular e hierárquica

Por que usar POO?

Organização

Estrutura o código em unidades lógicas e coesas, facilitando a compreensão e manutenção do sistema.

Reutilização

Permite aproveitar componentes em diferentes partes do sistema ou em outros projetos, reduzindo duplicação.

Manutenção

Facilita alterações e correções localizadas, com menor impacto no sistema como um todo.

Escalabilidade

Proporciona base sólida para crescimento do sistema, com adição de novas funcionalidades de forma modular.

Evolução da POO no PHP

PHP 4

Suporte limitado à POO, com implementação básica e incompleta de classes e objetos.

Ausência de recursos importantes como visibilidade de propriedades e métodos.

PHP 5

Revolução na POO com implementação completa de classes, herança, interfaces, métodos abstratos e tratamento de exceções.

Introdução dos modificadores de acesso (public, private, protected).

PHP 7+

Melhorias significativas de performance e introdução de tipagem de parâmetros e retornos.

Adição de classes anônimas e operador de resolução de escopo (::).

PHP 8+

Introdução de atributos, constructor property promotion, tipos de união e match expressions.

Aprimoramentos na sintaxe para escrever código orientado a objetos de forma mais concisa.

Pilares da POO

Encapsulamento

Oculta detalhes internos e expõe apenas o necessário

Protege os dados e garante que sejam acessados de forma controlada

Herança

Permite que classes derivem de outras classes

Promove reutilização de código e estabelece hierarquias

Polimorfismo

Permite que objetos de diferentes classes respondam à mesma mensagem

Possibilita tratar objetos de tipos diferentes de maneira uniforme

Estes três pilares formam a base conceitual da programação orientada a objetos.

O que é uma Classe?

Molde para criação de objetos

Uma classe é como uma planta arquitetônica que define:

- **Atributos**: variáveis que representam as características dos objetos
- **Métodos**: funções que definem os comportamentos dos objetos

Classes permitem definir um tipo personalizado que encapsula dados e comportamentos relacionados.

O que é um Objeto?

Instância de uma classe

Representa uma entidade do mundo real com características e comportamentos específicos.

Enquanto a classe é o modelo, o objeto é a materialização desse modelo com valores concretos. Por exemplo, a classe "Carro" define as características gerais, enquanto um objeto específico pode ser "Meu Fiat Uno vermelho de 2010".

Exemplo em PHP (Classe Produto)

```
class Produto {
    public $nome;
    public $preco;

    function exibir() {
        echo "$this->nome: R$ $this->preco";
    }
}
```

Componentes da classe:

- **Atributos**: $nome e $preco (públicos)
- **Método**: exibir() - mostra os dados do produto
- **$this**: referência ao próprio objeto

Esta classe define um modelo para produtos com nome, preço e a capacidade de exibir suas informações.

Criando um Objeto

Instanciação

```
$p = new Produto();
```

Criamos uma nova instância da classe Produto e a armazenamos na variável $p.

Definição de Atributos

```
$p->nome = "Caneta";
$p->preco = 3.5;
```

Atribuímos valores específicos aos atributos do objeto.

Chamada de Método

```
$p->exibir();
```

Executamos o método que mostrará "Caneta: R$ 3.5" na tela.

Discussão em Grupo

Qual vantagem de criar uma classe em vez de usar funções soltas?

Discuta com seus colegas como a organização em classes pode melhorar:

- A estrutura do código
- A manutenibilidade
- A reutilização
- A legibilidade
- O trabalho em equipe
- A modelagem do problema

Conceitos-chave

Classe ≠ Objeto

Classe é o modelo/planta

Objeto é a instância concreta

Ex: Classe Carro vs. Objeto MeuCarro

Atributos

São as variáveis da classe

Representam características

Ex: $marca, $modelo, $ano

Métodos

São as funções da classe

Representam comportamentos

Ex: ligar(), acelerar(), frear()

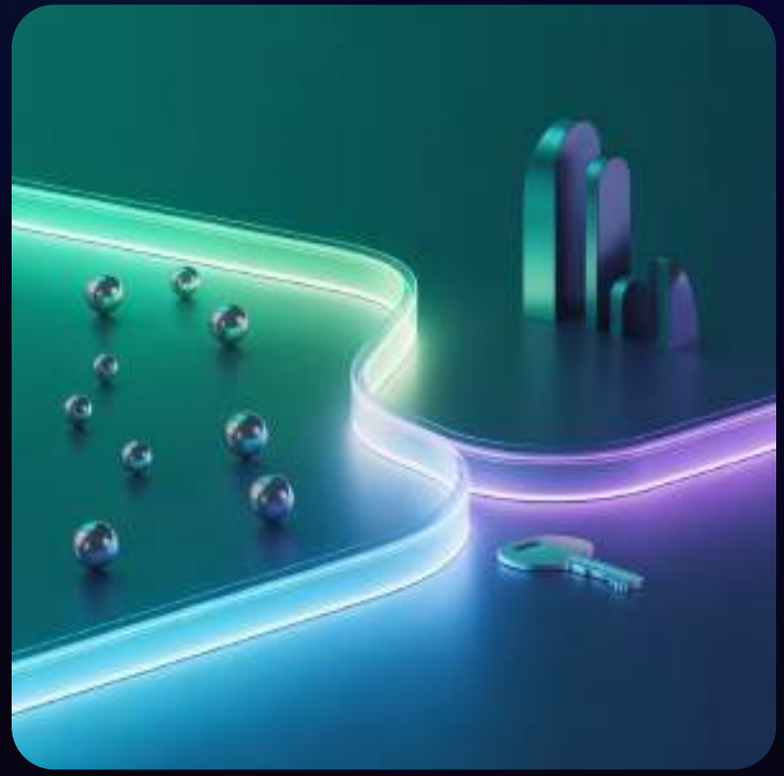
Compreender estas distinções é fundamental para aplicar corretamente a POO em seus projetos.

Sintaxe Básica de Classe

```
class Produto {
    // Modificadores de acesso
    public $nome;      // Acessível por todos
    private $estoque;  // Só acessível internamente
    protected $custo;  // Acessível por subclasses

    // Construtor
    public function __construct($nome, $estoque, $custo) {
        $this->nome = $nome;
        $this->estoque = $estoque;
        $this->custo = $custo;
    }
}
```

Elementos importantes:

- **Modificadores de acesso**: controlam a visibilidade dos membros
- **Construtor**: método especial executado na criação do objeto
- **$this**: referência ao objeto atual



Desafio Rápido

Converta o código procedural abaixo em uma classe:

Código Procedural

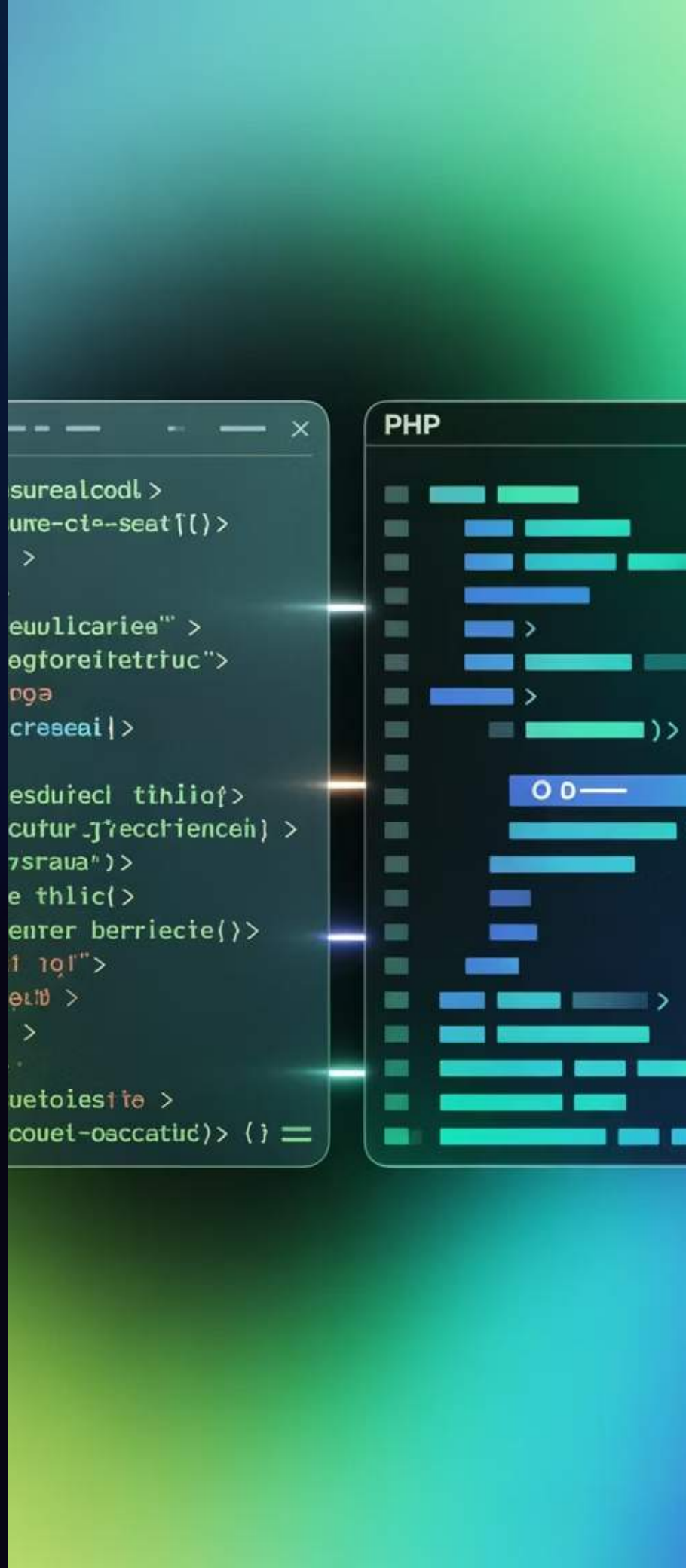
```
// Dados do usuário
$nome = "João";
$email =
"joao@exemplo.com";
$idade = 25;

// Função para exibir dados
function exibirDados($n, $e,
$i) {
    echo "Nome: $n, Email: $e,
Idade: $i";
}

// Função para verificar
maioridade
function ehMaiorDeIdade($i) {
    return $i >= 18;
}

// Chamadas
exibirDados($nome, $email,
$idade);
$maioridade =
ehMaiorDeIdade($idade);
```

Versão Orientada a Objetos

Como ficaria este código usando uma classe Usuario?

- Defina os atributos apropriados
- Crie os métodos necessários
- Inclua um construtor
- Demonstre a instanciação e uso

Tempo: 5 minutos



Resumo

Modelo Realista

A POO permite representar entidades do mundo real de forma mais natural e intuitiva no código.

Evolução do PHP

O PHP evoluiu de suporte limitado para amplo suporte a POO, tornando-se uma linguagem moderna e poderosa.

Estruturação

A POO organiza o código em unidades coesas e com baixo acoplamento, facilitando desenvolvimento e manutenção.

A programação orientada a objetos não é apenas uma técnica, mas uma forma de pensar sobre problemas e suas soluções computacionais.

Quiz Rápido

1. **O que é encapsulamento?**

   Explique como este pilar da POO ajuda a proteger os dados de uma classe.

2. **Como instanciar um objeto?**

   Demonstre a sintaxe correta para criar uma instância de uma classe em PHP.

3. **Diferença entre método e função?**

   Explique a principal distinção entre estes dois conceitos.

4. **Por que usar construtor?**

   Descreva o propósito e as vantagens de implementar um método construtor.

5. **Nome de uma boa prática na POO?**

   Mencione uma prática recomendada ao desenvolver código orientado a objetos.

Preview da Próxima Aula

Classes e Objetos em Detalhes

Aprofundaremos nos conceitos fundamentais:

- Propriedades e métodos avançados
- Métodos mágicos do PHP
- Construtores e destrutores
- Membros estáticos
- Namespaces e autoloading

Exercício Prático

Desenvolveremos um mini-sistema com reuso de componentes e estruturação orientada a objetos.

Venha preparado para colocar em prática os conceitos vistos hoje e expandir seu conhecimento em POO com PHP!