Aula 6 – PHP

Prof. Dr. Muriel Figueredo Franco
muriel.franco@ufcspa.edu.br

UFCSPA
A FEDERAL DA SAÚDE

Informática Biomédica
UFCSPA

Agenda

- Introdução ao PHP
- Atividade em Aula
- Banco de Dados + PHP
- Segurança básica em PHP
- Atividade em Aula

PHP

- Linguagem de programação interpretada, criada em 1994
- Principais características:
  - Muito usada em back-end web
  - Fácil integração com HTML e banco de dados
  - Suporta paradigmas estruturado e orientado a objetos
  - Grande comunidade e vasta quantidade de frameworks
  - Base de sistemas famosos como o Wordpress, Moodle, MediaWiki e Drupal

PHP

- Linguagem de programação interpretada, criada em 1994
- Principais características:
  - Muito usada em back-end web
  - Fácil integração com HTML e banco de dados
  - Suporta paradigmas estruturado e orientado a objetos
  - Grande comunidade e vasta quantidade de frameworks
  - Base de sistemas famosos como o Wordpress, Moodle, MediaWiki e Drupal

PHP é uma linguagem de servidor que transforma **páginas estáticas em sistemas dinâmicos** na web.

[HTML + CSS + JS] + PHP

- PHP atua como backend, gerando HTML, CSS e Javascript dinamicamente e enviando o resultado já pronto para o navegador

O código PHP

```
<?php
  $nome = “InfoBio”;
  echo “<h1>$nome existe na UFCSPA, UFPR e USP”</h1>”;

?>
```

O que o navegador vê

```
<h1>InfoBio existe na UFCSPA, UFPR e USP</h1>
```

Variáveis

- As variáveis no PHP são representadas por um cifrão seguido pelo nome da variável

```
<?php
    $var = “Teste”;
    $nome = “InfoBio”;
    echo “Meu curso é $nome”;
?>
```

https://www.php.net/manual/pt_BR/langref.php

Arrays

- Um array em PHP é um mapa ordenado
  - Relação chave-valor

Índices Numéricos

```
<?php
    $valores = [10, “Valor”, 30];
    $outros = array(40, 50, 60);
?>
```

Arrays

- Um array em PHP é um mapa ordenado
  - Relação chave-valor

Índices Numéricos

```
<?php
    $valores = [10, “Valor”, 30];
    $outros = array(40, 50, 60);
?>
```

Índices Associativos

```
<?php
    $valores = array(
        “chave” => “valor”,
        chave2 => “valor2”,
        …
    );
?>
```

Condicionais e Laços

- PHP oferece os principais tipos de laços que já conhecemos de outras linguagens
  - For, while, do … while, foreach (melhor forma para percorrer arrays), switch
  - If, else, elseif

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
?>
```

Condicionais e Laços

- PHP oferece os principais tipos de laços que já conhecemos de outras linguagens
  - For, while, do … while, foreach (melhor forma para percorrer arrays), switch
  - If, else, elseif

```php
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
?>
```

```php
<?php
$inst = ["UFCSPA", "IFRS", "UFRGS"];
foreach ($nome as $inst) {
    echo "A $nome fica em POA!<br>";
}
?>
```

Classes e Objetos

- PHP também suporta Programa Orientada a Objetos (POO)!
  - Útil para projetos médios ou grande onde precisamos organizar o código em módulos paa reuso, criação de APIs, frameworks e bibliotecas
    - Frameworks modernos usam POO: Laravel, Symfony, Wordpress, etc.

Classes e Objetos

- PHP também suporta Programa Orientada a Objetos (POO)!
  - Útil para projetos médios ou grande onde precisamos organizar o código em módulos paa reuso, criação de APIs, frameworks e bibliotecas
    - Frameworks modernos usam POO: Laravel, Symfony, Wordpress, etc.

```
<?php
Class Pessoa {
    public string $nome
    public function apresentar(): void {
        echo “Nome: “ . $this-nome;
}
?>
```

[HTML + CSS + JS] + PHP

Form/Index.php e
Form/Receba.php

Exercício

- Ajustar o exemplo para receber os dados via GET e mostrar em uma nova página em formato JSON
  - Dica: utilizar a função json_encode($dados)
    - https://www.php.net/manual/pt_BR/function.json-encode.php

Banco de Dados

- A **conexão com banco de dados** (BD) acontece utilizando o PHP Data Objects (PDO), uma abstração de acesso ao BD
  - É o jeito **moderno e seguro** de conectar PHP a qualquer BD

Banco de Dados

- A **conexão com banco de dados** (BD) acontece utilizando o PHP Data Objects (PDO), uma abstração de acesso ao BD
  - É o jeito **moderno e seguro** de conectar PHP a qualquer BD
- As principais vantagens do PDO incluem:
  - Funciona com vários BD com a mesma interface
    - MySQL, PostgreSQL, SQLite, Oracle, SQL Server, etc.
  - Usa **prepared statements** (mais seguros)
  - Facilitada para tratar erros e exceções
  - Código mais organizado e reuso

Criar conexão e Preparar

- A conexão acontece criando um **objeto do tipo PDO**

```
$pdo = new PDO("mysql:host=localhost;dbname=nome;charset=utf8", "user", "senha");

$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

Criar conexão e Preparar

- A conexão acontece criando um **objeto do tipo PDO**

```
$pdo = new PDO("mysql:host=localhost;dbname=nome;charset=utf8", "user", "senha");

$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

- **prepare()** diz ao BD que uma query será executada várias vezes, com valores diferentes
  - O banco irá sempre analisar e compilar a instrução antes de receber os dados

```
$sql = "INSERT INTO usuarios (nome, email) VALUES (:nome, :email)";
$stmt = $pdo->prepare($sql);
```

Anlisando o prepare($sql)

- O PHP manda para o BD apenas o esqueleto da SQL, com os places holders adicionados (por exemplo, :nome e :email)
  - O BD analisa a sintaxe e compila internamente, o **:nome e :email** são convertidos para **marcadores internos “?”** pelo driver do BD

Anlisando o prepare($sql)

- O PHP manda para o BD apenas o esqueleto da SQL, com os places holders adicionados (por exemplo, :nome e :email)
  - O BD analisa a sintaxe e compila internamente, o **:nome e :email** são convertidos para **marcadores internos “?”** pelo driver do BD
- O BD reserva espaço na memória e já decide como o comando irá rodar (mas ainda sem os dados)
  - Caso a mesma query seja rodada várias vezes, **não precisa recompilar internamente, apenas passar os dados**
  - Garante segurança contra SQL injection :)

Executar

- Precisamos definir **quais valores serão executados na query**

```
$sql = "INSERT INTO usuarios (nome, email) VALUES (:nome, :email)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    "nome"  => "Maria",
    "email" => "maria@email.com"
]);
```

Executar

- Precisamos definir **quais valores serão executados na query**

```
$sql = "INSERT INTO usuarios (nome, email) VALUES (:nome, :email)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    "nome"  => "Maria",
    "email" => "maria@email.com"
]);
```

- O **bindValue()** também pode ser utilizado antes do execute()

```
$stmt->bindValue(":nome", $nome);
$stmt->bindValue(":email", $email);
$stmt->execute();
```

Executar

- Precisamos definir **quais valores serão executados na query**

```
$sql = "INSERT INTO usuarios (nome, email) VALUES (:nome, :email)";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    "nome"  => "Maria",
    "email" => "maria@email.com"
]);
```

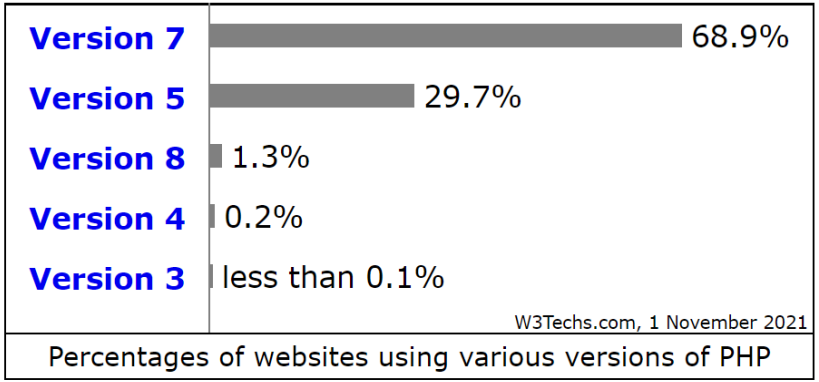
BD/index.php
BD/adicionar.php
BD/db.php
BD/processa.php

- O **bindValue()** também pode ser utilizado antes do execute()

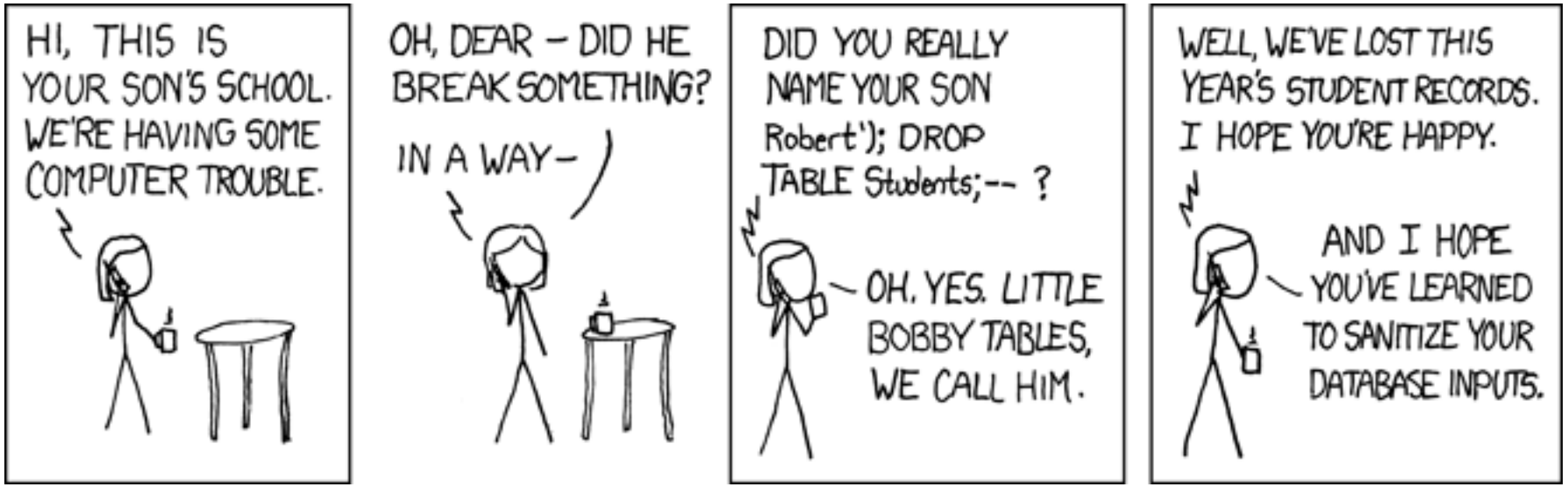
```
$stmt->bindValue(":nome", $nome);
$stmt->bindValue(":email", $email);
$stmt->execute();
```

Segurança: Cuidado!

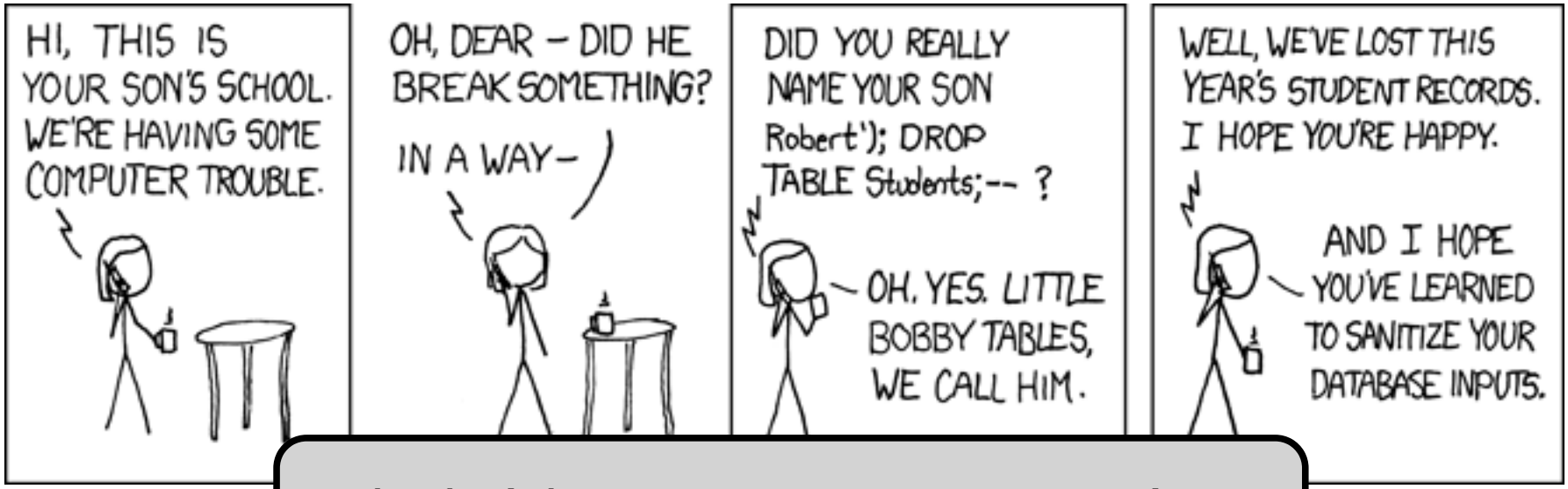
- PDO está consolidado desde o PHP 7 (2015), quando as outras funções fora removidas de vez
  - PDO está integrado ao core desde o PHP 5.1 (2005)
- O PDO facilita a escrita de código seguro, **mas não impede que o programador cometa erros**



SQL Injection!



SQL Injection!



/BD/Vuln/ProcessaMOSTRAR-QUERY.php
/BD/Vuln/ProcessaSQL-INJECTION.php

CRUD

- Representa as quatro operações fundamentais em sistemas que manipulam dados em um banco
  - C → Create (Criar): Inserir novos registros
  - R → Read (Ler): Consultar ou listar os registros
  - U → Update (Atualizar): Alterar os dados de um registro
  - D → Delete (Excluir): Remover um registro do banco
- É a base de todo sistema web, pois ajuda a organizar a lógica to sistema e o primeiro passo para aprender frameworks

Atividade em Aula

- Vamos desenvolver agora um CRUD básico
  - Create, Read, Update, Delete
- Utilize o **clinica.sql** disponível no Moodle e crie uma interface de CRUD para pacientes e consultas no banco de dados
- **Desafio extra:** Criar um endpoint em PHP puro (sem uso de frameworks) que retorne as consultas cadastradas em um JSON e utilizar o JSON para gerar gráficos
  - Consultas por mês, consultas por especialidades, etc.
  - **Exemplo:** Ao fazer um GET para /api/consultas.php?especialidade=Cardiologia, recebemos um JSON com todos os dados de consultas de Cardiologia.

Boa noite!

Prof. Dr. Muriel Figueredo Franco
muriel.franco@ufcspa.edu.br

UFCSPA
A FEDERAL DA SAÚDE

0101
101
001
110
100
Informática Biomédica
UFCSPA