(resultados.cfm?action=list)

2403 - LINGUAGENS DE SERVIDOR - Resultados

1 Lucas é um engenheiro de software com experiência em vários paradigmas de programação, mas recentemente, ele se juntou a uma empresa que trabalha principalmente com PHP para desenvolver aplicações web. Neste novo cargo, Lucas é responsável pelo desenvolvimento de um sistema de gestão de projetos online. Uma das principais características desse sistema é a capacidade de manter os usuários autenticados para que possam acessar seus projetos após o login.

No entanto, Lucas percebeu que alguns usuários estavam tendo problemas com suas sessões, perdendo o acesso enquanto estavam trabalhando em seus projetos. Preocupado com a segurança e a experiência do usuário, Lucas decidiu investigar mais a fundo a forma como as sessões foram implementadas.

Ao analisar o código, Lucas notou que o desenvolvedor anterior não estava usando uma abordagem padrão de gerenciamento de sessões em PHP. Em vez de usar as funções nativas de sessão do PHP, o código tinha uma implementação personalizada que usava cookies diretamente, sem qualquer tipo de criptografia ou validação adequada.

Lucas entende que, para corrigir o problema, ele precisará reescrever a maneira como o sistema lida com as sessões, incorporando as práticas recomendadas para garantir que as sessões sejam seguras e estáveis. Ele decide começar lendo a documentação oficial do PHP sobre sessões e implementa uma solução usando as funções nativas de sessão.

Com base neste cenário, qual das seguintes alternativas representa a abordagem mais apropriada e segura para iniciar e verificar uma sessão em PHP?

a
```
session_activate();
if ($_SESSION['user'] == "") {
    redirect('login.php');
}
```

b
```
if (!isset($_SESSION)) {
    session_initialize();
}
```

c
```
if (!session_id()) {
    session_start();
}
```

d
```
session_start();
if (!isset($_SESSION['loggedin'])) {
    header('Location: login.php');
}
```

e
```
if (!isset($_COOKIE['PHPSESSID'])) {
    session_create();
}
```

Pontuação: 1

2 Na hora de listar os produtos da loja, que vem dentro de uma *array* $produtos, o seu colega chegou ao seguinte código:

```
for($i = 0; $i < count($produtos); $i++) {
    echo '<p>'. $produtos[$i]['nome'] .'</p>';
}
```

Você identifica que o loop pode ser facilmente substituído por um *foreach.* Assim, sugere ao seu colega que substitua os trechos em negrito por:

a *foreach($produtos as $i)* e *$produtos[$i]['nome']*

b *foreach($produtos)* e $p['nome']

c *foreach($produtos as $p)* e *$p['nome']*

d *foreach($p in $produtos)* e *$p['nome']*

e *foreach($produtos)* e *$produtos['nome']*

Pontuação: 1

3 Leonardo é um desenvolvedor de sistemas web e, recentemente, foi contratado por uma organização não governamental (ONG) dedicada à proteção animal. Esta ONG tem um site onde os usuários podem adotar animais, doar recursos e se voluntariar para diversas atividades. O site era mantido por um antigo desenvolvedor que usava funções mysql_* para conectar-se ao banco de dados, mas, com o passar do tempo, esse tipo de conexão tornou-se obsoleto e inseguro. Sabendo disso, a diretoria da ONG decidiu que era o momento de atualizar o sistema.

Dentre as preocupações da diretoria, a segurança era a principal. Eles haviam lido sobre SQL Injection e estavam alarmados com a possibilidade de um ataque deste tipo. Leonardo, então, decidiu implementar o PDO (PHP Data Objects) no sistema, pois sabia que ele proporciona uma camada de acesso ao banco de dados mais segura e permite a utilização de prepared statements, que ajudam a prevenir ataques de SQL Injection.

Leonardo começou a refatorar o código. Ele teve que transformar várias consultas diretas ao banco, que eram vulneráveis, em consultas seguras usando PDO. Uma das tarefas era pegar os detalhes de um animal específico para mostrar na página de detalhes.

A antiga consulta era algo como:

```
$query = "SELECT * FROM animais WHERE id = " . $_GET['id'];
$result = mysql_query($query);
$animal = mysql_fetch_assoc($result);
```

Leonardo sabe que este código é vulnerável a ataques de SQL Injection por usar diretamente o valor de `$_GET['id']` na consulta.

Com base nessa situação, qual das seguintes alternativas representa a maneira correta e segura de reescrever a consulta acima usando PDO e prepared statements?

a
```
$pdo = new PDO('mysql:host=localhost;dbname=ong', 'user', 'password');
$query = "SELECT * FROM animais WHERE id = $_GET['id']";
$stmt = $pdo->prepare($query);
$stmt->execute();
$animal = $stmt->fetch();
```

b
```
$pdo = new PDO('mysql:host=localhost;dbname=ong', 'user', 'password');
$query = "SELECT * FROM animais WHERE id = " . $_GET['id'];
$stmt = $pdo->query($query);
$animal = $stmt->fetch();
```

c
```
$pdo = new PDO('mysql:host=localhost;dbname=ong', 'user', 'password');
$query = "SELECT * FROM animais WHERE id = ?";
$stmt = $pdo->prepare($query);
$stmt->execute($_GET['id']);
$animal = $stmt->fetch();
```

d
```
$pdo = new PDO('mysql:host=localhost;dbname=ong', 'user', 'password');
$query = "SELECT * FROM animais WHERE id = :id";
$stmt = $pdo->prepare($query);
$stmt->execute([':id' => $_GET['id']]);
$animal = $stmt->fetch();
```

e
```
$pdo = new PDO('mysql:host=localhost;dbname=ong', 'user', 'password');
$query = "SELECT * FROM animais";
$stmt = $pdo->query($query);
$animal = $stmt->fetch();
```

Pontuação: 1

4 Você foi contratado por uma empresa para desenvolver uma aplicação de *e-commerce*. Na sua primeira semana, você foi incumbido de fazer programação em dupla com um programador *back-end* júnior. Juntos, vocês vão fazer o formulário de login do e-commerce. No entanto, o seu colega não sabe muito bem qual seria o método mais adequado para o formulário. Você responde que o mais adequado é o método:

a GET, pois mantém as informações explícitas para o usuário, ideal para um formulário de login;

b POST, pois mantém as informações escondidas do usuário, ideal para um formulário de login;

c GET ou POST seriam métodos igualmente adequados para este formulário.

d POST, pois mantém as informações explícitas para o usuário, ideal para um formulário de login;

e GET, pois mantém as informações escondidas do usuário, ideal para um formulário de login;

Pontuação: 1

5 Lucas, um desenvolvedor PHP, compreendeu a importância da programação orientada a objetos (POO) após frequentar um workshop sobre boas práticas de codificação. Ele aprendeu que a POO oferece uma abordagem modular e reutilizável, permitindo que os sistemas sejam mais escaláveis e manuteníveis. Inspirado, Lucas decidiu refatorar um projeto de e-commerce que estava desenvolvendo, convertendo suas funções soltas em classes bem definidas.

Ele deseja criar uma classe chamada "Produto" para representar os itens disponíveis na loja. Esta classe deverá ter propriedades para o nome do produto, preço e uma função para exibir as informações do produto. Qual das alternativas a seguir é a implementação correta de uma classe "Produto" em PHP?

a
```
class Produto {
    var $nome;
    var $preco;

    function mostrarProduto() {
        return "Nome: " . $nome . ", Preço: " . $preco;
    }
}
```

b
```
Produto {
    public nome;
    public preco;

    public function mostrarProduto() {
        return "Nome: " . $this->nome . ", Preço: " . $this->preco;
    }
}
```

c
```
Produto {
    var nome;
    var preco;

    public mostrarProduto() {
        echo "Nome: " . $this->nome . ", Preço: " . $this->preco;
    }
}
```

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

    function mostrarProduto() {
        echo "Nome: " . $nome . ", Preço: " . $preco;
    }
}
```

e

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

    public function mostrarProduto() {
        echo "Nome: " . $this->nome . ", Preço: " . $this->preco;
    }
}
```

Pontuação: 1

6 Fernanda é uma desenvolvedora backend que trabalha em uma start-up de comércio eletrônico. Ela foi designada para desenvolver a seção de "checkout" do site, onde os clientes inserirão detalhes do cartão de crédito para finalizar a compra.

Durante uma reunião de planejamento, Fernanda e sua equipe discutiram sobre quais métodos HTTP, especificamente GET e POST, deveriam ser usados para diferentes operações no site. Fernanda defendeu que a submissão de detalhes do cartão de crédito dos clientes deveria usar um método em particular por razões de segurança e para não expor informações sensíveis na URL do navegador.

Qual método HTTP Fernanda deveria escolher para a submissão de detalhes do cartão de crédito dos clientes ao finalizar a compra?

a Ambos, GET e POST, porque a escolha do método não afeta a segurança da transação.

b POST, porque mantém os dados enviados ocultos na URL e é mais seguro para informações sensíveis.

c POST, porque pode enviar grandes quantidades de dados.

d GET, porque é mais rápido e eficiente.

e GET, porque pode manter os detalhes do cartão de crédito na URL para referência futura.

Pontuação: 1

7 A recursividade é um conceito fundamental em computação e programação, referindo-se à capacidade de uma função chamar a si mesma para resolver um problema. Esta técnica pode ser extremamente útil ao lidar com estruturas de dados hierárquicas ou problemas que podem ser divididos em subproblemas mais simples, semelhantes ao problema original. Embora a recursividade possa ser uma ferramenta poderosa, os desenvolvedores devem ser cautelosos ao utilizá-la, pois o uso inadequado pode levar a um consumo excessivo de recursos ou mesmo a uma falha no programa devido a uma "recursão infinita".

No contexto do PHP, assim como em muitas outras linguagens de programação, a recursividade é implementada permitindo que uma função chame a si mesma. Uma aplicação clássica da recursividade é o cálculo do fatorial de um número.

Dado este contexto, escreva um código em PHP que implementa uma função recursiva para calcular o fatorial de um número.

```php
<?php
// Função recursiva para calcular o fatorial de um número
function fatorial($n) {
    // Caso base: o fatorial de 0 ou 1 é 1
    if ($n == 0 || $n == 1) {
        return 1;
    }
    // Caso recursivo: n! = n * (n-1)!
    return $n * fatorial($n - 1);
}
```

Conceito: Certo - Pontuação: 4
Explicação:
Sugestão de solução;

```php
function fatorial($n) {
    if ($n == 0) {
        return 1;
    } else {
        return $n * fatorial($n - 1);
    }
}
```

Legenda:

Alternativa correta

Resposta do aluno

Pontuação total: **9**