

MÓDULO

PROGRAMAÇÃO PHP

UNIDADE

INTERAÇÃO DO PHP COM FORMULÁRIOS HTML

ÍNDICE

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. OBJETOS \$_GET E \$_POST.....	5
2. SESSÕES.....	10
CONCLUSÃO.....	17
AUTOAVALIAÇÃO.....	19
SOLUÇÕES.....	23
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO.....	24
BIBLIOGRAFIA	25

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Interagir com formulários HTML e dados passados por URL.

INTRODUÇÃO

A principal característica das linguagens de programação como o PHP é a possibilidade de trabalhar com dados de formulários, ou seja, poder interagir com o utilizador para solicitar e entregar informações.

Graças a esta funcionalidade, o PHP não é utilizado apenas nas páginas web corporativas, mas também na criação de CRM (Customer Relationship Management) ou Intranet das empresas, uma vez que funciona em qualquer computador ligado à rede e tem a potência de qualquer ferramenta de software.

1. OBJETOS \$_GET E \$_POST

Estes objetos são duas variáveis do método HTTP.

No caso de \$_GET, é um array que contém os elementos passados pela URL para a página, ou seja, do link:

- www.paginaficticia.com/lista.php?pagina=1&idioma=pt

O array \$_GET teria dois elementos, um com a chave “pagina” com valor 1 e o outro com a chave “idioma” e o valor pt.

A diferença para \$_POST é que os valores do array tendo \$_POST não são passados por URL (nem são visíveis para o utilizador), mas são passados por formulário, e são cada um dos campos ou elementos do formulário. Por exemplo:

```
<form name="teste" method="post" action="lista.php">
<input type="text" name="nome">
<input type="checkbox" name="presente">
<input type="submit" name="enviar">
</form>
```

Os elementos que o array \$_POST irá conter são: a primeira chave “nome” e o valor que o utilizador escrever no campo, a segunda chave “presente” e valor on caso o utilizador o selecione ou, caso contrário, vazio.

Um formulário pode ser criado com o método GET em vez de POST. Nesse caso deverá recolher os valores com o array \$_GET.

Os dados recebidos nestes arrays já são tratados com a função urldecode e não é necessário passá-los pela mesma. Ao fazê-lo, mudar-se-á o conteúdo dos arrays para erro.

No PHP existe um segundo método para poder recolher os valores recebidos pela página web, seja através do método POST ou do método GET. Este método consiste em utilizar a variável genérica \$_REQUEST, que contém um array com os valores de \$_GET, \$_POST e \$_COOKIE.

O exemplo abaixo estará dividido em dois ficheiros: o primeiro será um ficheiro HTML que contém um formulário com vários campos, designado "form.html", e o segundo será um ficheiro PHP, utilizado para ler os dados enviados por meio do formulário da página anterior.

■ form.html:

```
<!Doctype html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
</head>
<body>
<p>
    <form method="post" action="exemplo.php" name="form">
        Nome <input type="text" name="nome">
        <br>
        Apellidos <input type="text" name="apelidos">
        <br>
        Idade<input type="text" name="idade">
        <br>
        Cor preferida
        <select name="cor">
            <option value="azul">azul</option>
```



```

        <option value="laranja">laranja</option>
    </select>
    <br>
    Gosto de cinema
    <input type="checkbox" name="cinema"><br>
    Descrição
    <textarea cols="20" rows="3" name="descricao">
    </textarea>
    <br>
    <input type="submit" value="enviar">
</form>
</p>
</body>
</html>

```

■ exemplo.php:

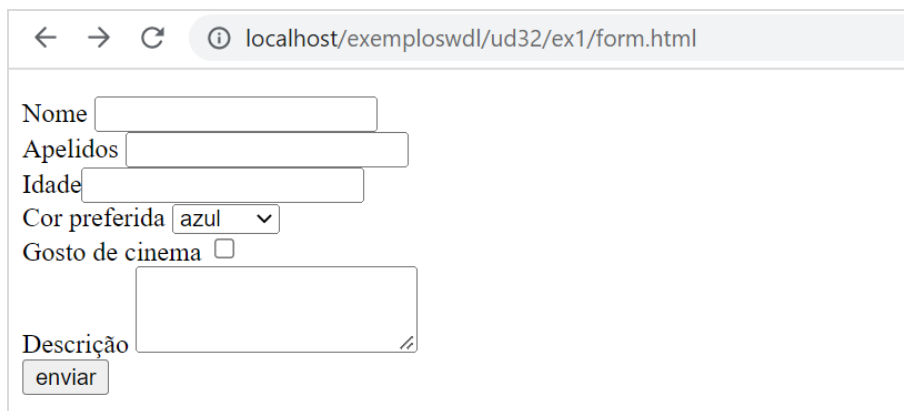
```

<html>
<body>
<p>
<?php
echo 'Olá ' . $_POST['nome'] . ' ' . $_POST['apelidos'] . '<br>';
echo 'Tem ' . $_POST['idade'] . ' anos' . '<br>';
echo 'A sua cor favorita é o ' . $_REQUEST['cor'] . '<br>';
if ($_POST['cinema'] == 'on'){
echo 'Gosta de cinema<br>';
}else{
echo 'Não gosta de cinema <br>';
}
echo ' E uma breve descrição sobre si seria
' . $_REQUEST['descricao']
?>
</p>

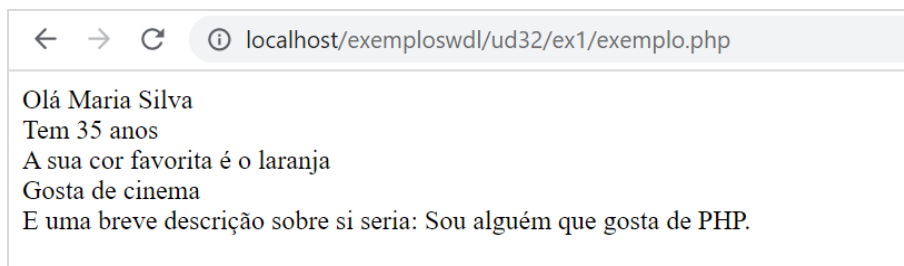
```

```
</body>
</html>
```

O resultado da visualização do ficheiro HTML antes e depois da inserção de dados e respetivo envio seria, por exemplo:



A screenshot of a web browser window. The address bar shows 'localhost/exemploswld/ud32/ex1/form.html'. The form contains the following fields: 'Nome' (text input), 'Apelidos' (text input), 'Idade' (text input), 'Cor preferida' (dropdown menu with 'azul' selected), 'Gosto de cinema' (checkbox), and 'Descrição' (text area). Below the text area is a button labeled 'enviar'.



A screenshot of a web browser window. The address bar shows 'localhost/exemploswld/ud32/ex1/exemplo.php'. The page displays the submitted data: 'Olá Maria Silva', 'Tem 35 anos', 'A sua cor favorita é o laranja', 'Gosta de cinema', and 'E uma breve descrição sobre si seria: Sou alguém que gosta de PHP.'

O PHP também permite trabalhar com parâmetros complexos, como uma seleção múltipla, sendo que receberia um array com os valores selecionados:

```
<?php
if ($_POST) {
    echo '<pre>';
    echo htmlspecialchars(print_r($_POST, true));
    echo '</pre>';
}
```

```
?>
<form action="" method="post">
Nome: <input type="text" name="nome"><br>
Email: <input type="text" name="email"><br>
Cores favoritas: <br>
<select multiplename="cores[]">
<option value="azul">azul</option>
<option value="laranja">laranja</option>
<option value="amarelo">amarelo</option>
</select><br>
<input type="submit" value="enviar">
</form>
```

No exemplo anterior, é utilizada a função **htmlspecialchars**, que converte caracteres especiais como o “e” comercial, aspas, etc., nas suas respectivas entidades HTML.

2. SESSÕES

Para começar a utilizar sessões em PHP terá de se inicializar a referida sessão. Para isso, utiliza-se a função **session_start()**, e uma vez que esta sessão for chamada é possível criar todas as variáveis de sessão que se pretender, sendo que estas são guardadas num array chamado `$_SESSION`, que pode ser acedido a partir de qualquer lugar da página web.

```
<?php
session_start();
if (!isset($_SESSION['contador'])){
    $_SESSION['contador'] = 0;
}else{
    $_SESSION['contador'] ++;
}
?>
```

O exemplo anterior inicia a sessão e verifica se a variável de sessão "contador" já está definida com a função `isset` negada com um ponto de exclamação (!); se não for definido, começa com o valor 0, caso contrário, adiciona 1 ao valor da variável.

Para "cancelar" o registo de uma variável de sessão, utiliza-se a função **unset** e como parâmetro é passada a variável que se pretende cancelar.

```
<?php unset($_SESSION['count']); ?>
```

O início da sessão **session_start()** deve ocorrer antes de qualquer definição HTML da página, uma vez que envia cabeçalhos HTTP para o navegador e, caso isso aconteça, ocorrerá um erro, posteriormente.

No exemplo abaixo, é criada uma sessão que exibe um valor armazenado na variável de sessão. Cada vez que a página for atualizada, o valor da referida variável aumentará um valor.

```
<?php session_start();  
if (!isset($_SESSION['contador'])) {  
    $_SESSION['contador'] = 0;  
} else {  
    $_SESSION['contador'] ++;  
}  
?>  
  
<html>  
<body>  
<p>  
<?php echo $_SESSION['contador']; ?>  
</p>  
</body>  
</html>
```

Este valor aumentará, contanto que a sessão do navegador não expire e o navegador não seja fechado, o que esvazia a cache da sessão. Este tempo é configurado no ficheiro .ini do PHP e pode ser modificado a partir deste ficheiro ou, temporariamente, da página web, utilizando a função `ini_set()`, que permite modificar alguns valores do ficheiro .ini.

```
<?php ini_set('session.cache_expire', $tempo); ?>
```

Com a instrução anterior é modificado o tempo, passando com a variável `$time` (em segundos) quanto tempo se pretende que a sessão dure. Por padrão (depende da instalação), esse tempo é de 180 segundos.

No PHP existem várias funções de sessão que ajudarão a trabalhar com as sessões. As mais utilizadas são:

- **session_cache_expire**: devolve o prazo da sessão atual. Também pode ser utilizada para atribuir o valor passando-o como parâmetro. Esta função trata os dados em minutos e não em segundos. Além disso, deve ser utilizada antes de iniciar a sessão, ou seja, antes da função `session_start()`.

```
<?php session_cache_expire(10); // termina em dez minutos ?>
```

- **session_cache_limiter**: recebe como parâmetro o nome do limitador de cache desejado e, de acordo, com o limitador envia alguns cabeçalhos HTTP. Podem ser:

Valor	Cabeçalhos enviados
public	<ul style="list-style-type: none"> · Expires: "valor de acordo com o <code>session.cache_expire</code>" · cache-control: public, max-age="de acordo com <code>cache_expire</code>" · Last-Modified: "marca a data da última sessão"
private_no_expire	<ul style="list-style-type: none"> · Cache-control: private, max-age="de acordo com <code>cache_expire</code>", pre-check="de acordo com <code>cache_expire</code>" · Last-Modified: "marca a data da última sessão"
private	<ul style="list-style-type: none"> · Expires: Thu, 19 Nov 1981 08:52:00 GMT · Cache-control: private, max-age="de acordo com <code>cache_expire</code>", pre-check="de acordo com <code>cache_expire</code>" · Last-Modified: "marca a data da última sessão"
nocache	<ul style="list-style-type: none"> · Expires: Thu, 19 Nov 1981 08:52:00 GMT · Cache-control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0 · Pragma: no-cache

```
<?php session_cache_limiter('private'); ?>
session_destroy, destrói todas as informações na sessão atual.
<?php session_destroy(); ?>
session_id, permite saber ou modificar o id da sessão atual.
<?php
session_start();
echo "SID: ".SID."<br>session_id(): ".session_id();
//devolve 39e9d094fdcd5cf027b2534d6867e3a7 ou similar
?>
```

- **session_name:** permite recuperar ou definir o nome da sessão atual.

```
<?php session_name('teste'); ?>
```

- **session_regenerate_id:** permite regenerar o número de id por um novo.

```
<?php session_regenerate_id(); ?>
```

- **session_unset:** liberta todas as variáveis de sessão.

```
<?php session_unset(); ?>
```

Segue-se um exemplo de como se pode guardar uma variável de sessão e aceder-lhe a partir de outra página. No exemplo1.php são criadas as variáveis de sessão e no exemplo2.php são mostradas.

- exemplo1.php:

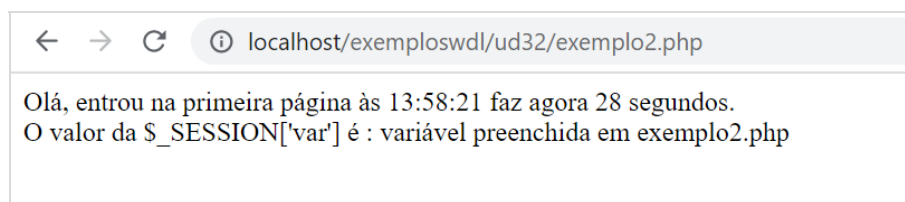
```
<?php
// session1.php
session_start();
// guardar numa variável de sessão a hora atual.
$_SESSION['tTime'] = time();
$_SESSION['var'] = 'variável preenchida em exemplo2.php';
?>
```

```
<html>
<body>
<p>
<?php echo 'Sessão guardada. Agora vá para <a href="exemplo2.php">
exemplo2.php</a>';?>
</p>
</body>
</html>
```

■ exemplo2.php:

```
<?php
// session2.php
session_start();
if(!empty($_SESSION['tTime'])) {
$hora = date('H:i:s',$_SESSION['tTime']);
$cont = (time() - $_SESSION['tTime']);
echo 'Olá, entrou na primeira página às '.$hora.' faz agora
'.$cont.' segundos.';
echo '<br>';
echo 'O valor da $_SESSION[\'var\'] é : '.$_SESSION['var'];
} else {
echo 'Ainda não foi guardada uma variável de sessão. Vá para o
exemplo1.php';
}
?>
```


As visualizações serão:



CONCLUSÃO

Trabalhar com formulários é essencial para desenvolver dados de utilizadores ou para trabalhar com os mesmos dados entre ecrãs. Para realizar este trabalho, passando dados entre ecrãs, também terá as variáveis de sessão, que poderão conter valores durante toda a execução do website.

AUTOAVALIAÇÃO

1. Qual é a diferença entre GET e POST?
 - a) Nenhuma, não existe diferença.
 - b) O GET envia os campos e os seus valores ocultos.
 - c) O POST envia os campos e os seus valores ocultos.
 - d) O GET obtém dados de uma página, enquanto o POST envia.

2. Qual é a variável global que os arrays \$_GET e \$_POST contêm?
 - a) \$REQUEST.
 - b) \$_REQUEST.
 - c) \$_GETPOST.
 - d) \$_POSTGET.

3. O que é utilizado para iniciar uma sessão?
 - a) session_open.
 - b) session_start.
 - c) session_begin.
 - d) session_run.

4. O que é utilizado para "cancelar o registo" de uma variável de sessão?
- a) `remove(variable)`.
 - b) `variable.remove`.
 - c) `unset(variable)`.
 - d) `variable.unset`.
5. Em que posição deve estar a instrução para abrir a sessão?
- a) Depois de escrever as tags DOCTYPE e HTML da página web.
 - b) No final da página.
 - c) Num ficheiro externo que é carregado no head do HTML.
 - d) No início da página antes de qualquer código HTML.
6. O que é que a função `ini_set()` permite realizar?
- a) Abrir várias sessões ao mesmo tempo.
 - b) Ler a partir do ficheiro PHP `.ini`.
 - c) Modificar os valores no ficheiro Apache `.ini`.
 - d) Modificar os valores no ficheiro PHP `.ini`.
7. Qual é a função utilizada para alterar a duração de uma sessão?
- a) `session_cache_expire`.
 - b) `session_expire`.
 - c) `expire_session`.
 - d) `Sessioncache_expire`.

- 8. O que é que permite a função `session_cache_limiter()`?**
- a) Limitar o tempo máximo de uma sessão.
 - b) Limitar o número de variáveis que pode declarar numa sessão.
 - c) Enviar alguns cabeçalhos HTTP de acordo com o nome do limitador passado.
 - d) Não existe essa função.
- 9. Qual é a função que devolve o id da sessão atual?**
- a) `idSession`.
 - b) `session_id`.
 - c) `sessionID`.
 - d) `getIdSession`.
- 10. O que se utiliza se se pretender destruir todas as informações da sessão?**
- a) `session_delete`.
 - b) `session_unset`.
 - c) `unset_session`.
 - d) `session_destroy`.

SOLUÇÕES

1.	c	2.	b	3.	b	4.	c	5.	d
6.	d	7.	a	8.	c	9.	b	10.	d

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para saber mais sobre todos os recursos de sessão existentes, visite o seguinte website:

- http://www.php.net/manual/pt_BR/ref.session.php

BIBLIOGRAFIA

- Cabezas, L. M. (2010), *Php 5*. Madrid: Anaya Multimedia.
- The PHP Group (2001), “Manipulação de Sessão”. Disponível em https://www.php.net/manual/pt_BR/book.session.php
Consultado a 17 de março de 2021.

