# Variáveis externas

As **variáveis externas** no PHP são variáveis que chegam ao script de **fontes externas**, como requisições HTTP, arquivos, ambiente do sistema operacional, ou argumentos passados pela linha de comando. Elas são amplamente usadas para interagir com o mundo externo, seja por meio de dados enviados por um usuário, variáveis de ambiente do servidor ou outros contextos.

# Superglobais X Globais

### Superglobais

**O Que São?**

As **superglobais** são um conjunto de variáveis **pré-definidas** pelo PHP, disponíveis automaticamente em **todos os escopos** (global, local ou em funções). Você não precisa usar global ou passar essas variáveis explicitamente para funções.

**Características:**

* Sempre disponíveis, independentemente do escopo.
* Usadas para acessar informações do ambiente, servidor, requisições HTTP, sessões, etc.
* Exemplos de superglobais: $\_GET, $\_POST, $\_FILES, $\_REQUEST, $\_SERVER, $\_SESSION, $\_COOKIE, $\_ENV.

### Globais

**O Que São?**

As **variáveis globais** são variáveis que estão no **escopo global do script**, ou seja, declaradas fora de qualquer função ou classe. No entanto, **não são automaticamente acessíveis dentro de funções ou métodos**. Para usá-las em funções, você deve declará-las explicitamente como global ou acessá-las via $GLOBALS.

**Características:**

* Acessíveis no escopo global (fora de funções/classes).
* São variáveis criadas pelo desenvolvedor mais definidas como global com a palavra chave “global”
* Para serem acessadas dentro de funções, é necessário:
  1. Declarar com a palavra-chave global.
  2. Texto

     Descrição gerada automaticamenteUsar $GLOBALS, que é uma superglobal.

# categorizando & se aprofundando

Podemos organizar essas variáveis de acordo com **categorias baseadas em sua funcionalidade** e **origem dos dados**.

**1. Variáveis Globais**

* Variáveis que referenciam ou armazenam todas as informações disponíveis globalmente no script.
  + **$GLOBALS**: Referencia todas as variáveis disponíveis no escopo global.

**2. Variáveis HTTP**

* Usadas para capturar informações vindas de requisições HTTP (GET, POST, Cookies, etc.).
  + **$\_GET**: Dados enviados via método HTTP GET.
  + **$\_POST**: Dados enviados via método HTTP POST.
  + **$\_FILES**: Informações sobre arquivos enviados por upload.
  + **$\_REQUEST**: Combinação de dados de $\_GET, $\_POST e $\_COOKIE.
  + **$\_COOKIE**: Dados enviados por cookies HTTP.
  + **$http\_response\_header**: Cabeçalhos de resposta HTTP.

**3. Variáveis de Sessão**

* Usadas para armazenar e gerenciar informações relacionadas à sessão do usuário.
  + **$\_SESSION**: Dados armazenados na sessão.

**4. Variáveis de Ambiente**

* Informações sobre o ambiente de execução ou o sistema.
  + **$\_ENV**: Variáveis de ambiente definidas no sistema ou no servidor.
  + **$\_SERVER**: Informações do servidor e ambiente de execução.

**5. Variáveis CLI (Linha de Comando)**

* Específicas para scripts executados via CLI (linha de comando).
  + **$argc**: Número de argumentos passados ao script.
  + **$argv**: Array contendo os argumentos passados ao script.

# Exemplos

**1. Variáveis Globais**

Texto

Descrição gerada automaticamenteA superglobal $GLOBALS é um array associativo especial no PHP que contém todas as variáveis definidas no **escopo global**. É útil quando você deseja acessar ou modificar variáveis globais dentro de funções, sem usar a palavra-chave global.Texto

Descrição gerada automaticamente com confiança baixa

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**2. Variáveis de Ambiente**

$\_SERVER

$\_SERVER é uma superglobal no PHP que contém informações sobre os cabeçalhos HTTP, caminhos, e detalhes do ambiente de execução do servidor. Esses dados são disponibilizados pelo servidor web ou diretamente pelo PHP. A variável é preenchida automaticamente e é usada para acessar detalhes como:

Forma

Descrição gerada automaticamente com confiança média$\_SERVER é um array associativo, onde cada chave representa um aspecto específico do ambiente do servidor.

Como $\_SERVER é um array, podemos acessar seus valores com suas chaves, por mais que os valores sejam diferente de servidor pra servidor:

Texto

Descrição gerada automaticamenteSe nós apenas darmos um print\_r em $\_SERVER, teremos muitas informações sobre o servidor.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

>>>

Podemos acessar qualquer uma desses valores através de suas chaves, veremos sobre o que eles são a seguir, cada um deles.

#### Configurações de Ambiente do PHP e Servidor

Essas variáveis configuram o ambiente do PHP e a infraestrutura do servidor.

Texto

Descrição gerada automaticamente

#### Configurações de caminhos do sistema

Texto

Descrição gerada automaticamente  
Variáveis relacionadas aos diretórios e caminhos do sistema operacional.

#### informações do servidor

Variáveis fornecidas pelo servidor para identificar sua configuração e detalhes de execução.

Texto

Descrição gerada automaticamente

#### informações de requisição http

Interface gráfica do usuário, Texto

Descrição gerada automaticamenteVariáveis relacionadas à requisição HTTP feita pelo cliente.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

#### informações sobre o script em execução

Texto

Descrição gerada automaticamenteVariáveis que fornecem detalhes sobre o script PHP em execução.

#### informações sobre o cliente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamenteVariáveis que identificam o cliente que fez a requisição.

#### informações de tempo

Interface gráfica do usuário, Texto

Descrição gerada automaticamenteVariáveis que indicam o momento em que a requisição foi processada.

#### informações sobre a interface de comunicação

Variáveis relacionadas à interação entre o servidor e o cliente.

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

$\_ENV

$\_ENV é uma **variável superglobal** no PHP. Isso significa que ela está sempre disponível em todos os escopos do script sem a necessidade de declará-la como global. É um **array associativo** que armazena as **variáveis de ambiente** passadas para o script PHP no momento da execução.

As variáveis de ambiente geralmente contêm informações sobre o **sistema operacional**, **servidor** ou **aplicação**, podendo incluir configurações específicas do ambiente, como diretórios de sistema, configuração de usuário, etc.

***Como as variáveis de ambiente funcionam no PHP ?***

* Essas variáveis são carregadas no ambiente onde o PHP está sendo executado.
* Elas são definidas pelo **sistema operacional**, **servidor web** ou até mesmo pelo **programador** ao configurar o ambiente de execução.
* Por exemplo, em servidores Linux/Unix, as variáveis de ambiente podem ser configuradas no .bashrc, .bash\_profile, ou diretamente no terminal.

### **Exemplo Completo**

#### Configurando uma variável de ambiente em Linux:

Texto

Descrição gerada automaticamente

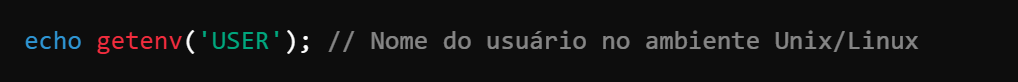
Texto

Descrição gerada automaticamente

* **Evite expor informações sensíveis em $\_ENV.**
  + Dados como senhas e chaves de API devem ser tratados com cuidado.
* **Prefira getenv() quando o $\_ENV estiver vazio.**
  + Em algumas configurações, o $\_ENV pode ser desativado por segurança.

### Funções Relacionadas

**getenv()**

Recupera o valor de uma variável de ambiente específica ou todas.

Diferença entre $\_ENV e getenv():

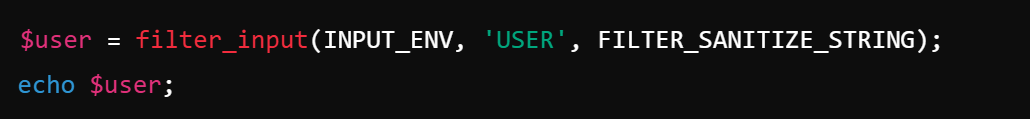
* $\_ENV depende de variáveis de ambiente carregadas no momento da inicialização do PHP.
* getenv() consulta o ambiente diretamente, mesmo que o $\_ENV esteja vazio.

**putenv()**

Texto

Descrição gerada automaticamenteDefine ou modifica uma variável de ambiente.

**filter\_input()**

Pode ser usado para acessar variáveis de ambiente com filtro de validação:

**3. Variáveis de Sessão**

$\_SESSION

$\_SESSION é uma **superglobal** no PHP que armazena **dados da sessão do usuário**.

É usado para manter informações entre diferentes **requisições HTTP**.

As sessões permitem que você identifique usuários únicos e armazene informações relacionadas a eles enquanto navegam no site.

As informações da sessão são armazenadas no servidor e cada usuário recebe um identificador exclusivo (geralmente em um cookie chamado PHPSESSID).

Os dados armazenados na sessão ficam disponíveis enquanto a sessão está ativa (até ser destruída ou expirar).

É usado principalmente para **persistir informações** (como login, carrinhos de compras, configurações do usuário, etc.).

***Como Funciona?***

* **Início da Sessão**
  + Para usar $\_SESSION, você deve iniciar a sessão no início do script, com a função session\_start().
* **Armazenamento de Dados**
  + O $\_SESSION é um array associativo onde você pode armazenar e recuperar informações.
* **Persistência de Dados**
  + Os dados armazenados em $\_SESSION persistem enquanto a sessão do usuário estiver ativa.
* **Encerrando sessão:**
  + Texto

    Descrição gerada automaticamenteTemos que incerrar a sessão depois de um tempo com session\_destroy( );

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

***Funções Relacionadas a Sessões***

Texto

Descrição gerada automaticamenteAqui estão as principais funções que você pode usar com $\_SESSION:

***Estrutura de Sessão***

As sessões geralmente utilizam cookies para armazenar o **identificador único do usuário** (PHPSESSID).

Os dados da sessão são armazenados no servidor, geralmente em arquivos ou banco de dados.

* Texto

  Descrição gerada automaticamenteExemplo de conteúdo de $\_SESSION

**Exemplo Avançado: Carrinho de Compras**

**Texto

Descrição gerada automaticamenteCódigo do Carrinho:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* A **sessão** ($\_SESSION) armazena os dados no servidor e o navegador mantém o ID da sessão em um cookie (geralmente chamado PHPSESSID).
* Os **cookies** (setcookie()) são armazenados no cliente (navegador), e podem conter dados úteis, como preferências ou tokens de autenticação.

Se você deseja armazenar dados específicos da sessão em cookies, você pode fazer isso manualmente.

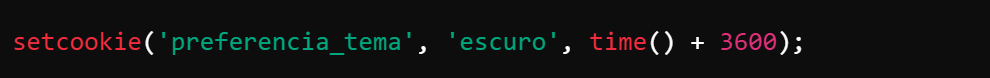
O navegador do cliente **não vem com cookies preenchidos por padrão**. Ele começa vazio e apenas armazenará cookies se o servidor ou o próprio cliente os criar explicitamente.

**Como os cookies são criados e usados?**

1. **Navegador vazio por padrão:**
   * Quando um usuário acessa um site pela primeira vez, o navegador não tem cookies específicos para esse domínio (a menos que já tenha visitado o site antes e os cookies ainda estejam válidos).
2. **Servidor cria cookies (opcional):**
   * O servidor pode **instruir o navegador a armazenar cookies**, usando setcookie() no PHP ou cabeçalhos HTTP apropriados. Esses cookies podem conter qualquer dado que você queira.
3. **Sessões usam cookies automaticamente:**
   * Quando você usa session\_start() no PHP, ele automaticamente gera um cookie com o **ID da sessão** (chamado PHPSESSID por padrão). Esse cookie serve para identificar o usuário de forma única e guardar informações dele.

**Você pode criar seus próprios cookies:**

* Além do PHPSESSID, você pode criar cookies adicionais para armazenar informações extras no navegador do cliente.



Cookies são pequenos arquivos de texto armazenados no navegador do cliente. Eles consistem em pares **nome/valor** que são enviados pelo servidor para o navegador ou criados diretamente pelo JavaScript no cliente. O navegador armazena esses cookies e os envia automaticamente ao servidor em cada requisição subsequente para o mesmo domínio.

É como se fosse uma tabela, essa tabela tem nomes e valores, esses são passados do site para o navegador e o navegador armazena eles. Sempre que o cliente entra no site novamente, esses nomes e valores são passados para o site e o site pega essas informações e cria uma sessão para o usuário com as informações contidas nos cookies.

Os **cookies adicionais** (se você criar outros) podem ser usados para armazenar informações que não precisam de processamento pelo servidor.

***Como funciona a relação entre cookies e sessões?***

Texto

Descrição gerada automaticamente

Cookies podem conter valores **em texto simples** ou **criptografados**.

Dados sensíveis nunca devem ser armazenados diretamente nos cookies.   
Em vez disso:

Use tokens (como tokens de sessão) que o servidor sabe interpretar.

O **PHPSESSID** é um cookie especial criado pelo PHP para gerenciar sessões. Ele possui tanto um **nome** (PHPSESSID) quanto um **valor**, que é um identificador único de sessão gerado automaticamente pelo PHP.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Se vc não achar PHPSESSID em uma página é pq aquele site não usa a linguagem PHP no back-end.

* Dados são armazenados no array associativo $\_SESSION no servidor.
* O navegador envia o cookie PHPSESSID para o servidor.
* O PHPSESSID apenas serve como uma chave para vincular o navegador do cliente aos dados salvos no array associativo $\_SESSION, que irá conter os dados daquela sessão
* Sempre que você **inicia uma sessão** com session\_start(), o **array superglobal $\_SESSION** é criado para armazenar os dados daquela específicos daquela sessão, mas ele não tem acesso a todas as sessões existentes.

1. **Variáveis de Linha de Comando (CLI)**

No PHP, **$argc** e **$argv** são variáveis predefinidas que permitem trabalhar com argumentos de linha de comando quando o script é executado no modo **CLI (Command Line Interface)**.

$\_argc

**O que é $argc?**

* $argc (argument count) é uma variável predefinida que contém o número de argumentos passados para o script na linha de comando.
* O valor de $argc **inclui o nome do script como o primeiro argumento**.

Interface gráfica do usuário, Texto

Descrição gerada automaticamenteExemplo:

Se você executar o seguinte script com os argumentos:

O valor de $argc será **3**:

1. Nome do script: meu\_script.php.
2. Primeiro argumento: arg1.
3. Segundo argumento: arg2.

$\_argv

**O que é $argv?**

* $argv (argument values) é uma variável predefinida que contém um **array de argumentos** passados para o script.
* O **primeiro elemento** do array $argv[0] é sempre o nome do script.
* Os **outros elementos** correspondem aos argumentos passados na linha de comando.

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

>>>

Isso parece não ter utilidade no Desenvolvimento Web, e não tem mesmo!

Como o PHP foi escrito em C, ele é meio que uma “linguagem ferramenta de C”, então ele pode ser usado para algumas coisas fora do Desenvolvimento Web, como no caso de:

- Scripts de automação

- Processamento de Arquivos e Dados e etc...

Claro, sempre teremos linguagens melhores para isso.

1. **Variáveis HTTP**

As **variáveis HTTP** em PHP são superglobais que permitem capturar e trabalhar com informações vindas de requisições feitas pelo cliente ao servidor.

Essas informações podem ser enviadas por métodos **GET**, **POST**, **Cookies** ou em forma de uploads de arquivos.

**$\_GET**

* **Descrição:**  
  Contém os dados enviados para o servidor usando o método HTTP **GET**, geralmente adicionados na URL como parâmetros.
* **Características:**
  + Os dados são visíveis na URL.
  + Útil para enviar informações não sensíveis, como filtros ou IDs.
  + Possui limite de tamanho (depende do servidor).

Texto

Descrição gerada automaticamente**Exemplo:** URL: [**http://example.com?nome=Joao&idade=25**](http://example.com?nome=Joao&idade=25)

**$\_POST**

* **Descrição:**  
  Contém os dados enviados para o servidor usando o método HTTP **POST**, geralmente em formulários.
* **Características:**
  + Os dados não aparecem na URL.
  + Ideal para envio de informações sensíveis, como senhas (com HTTPS)
  + Texto

    Descrição gerada automaticamenteTexto

    Descrição gerada automaticamenteSem limite estrito de tamanho (mas depende do servidor).

**$\_FILES**

* **Descrição:**  
  Contém informações sobre os arquivos enviados via upload HTTP.
* **Características:**
  + Permite acessar o nome, tipo, tamanho e local temporário dos arquivos enviados.
  + Estruturado como um array associativo.
  + Texto

    Descrição gerada automaticamenteRequer configuração no formulário (enctype="multipart/form-data").

Texto

Descrição gerada automaticamente

**$\_REQUEST**

* **Descrição:**  
  Contém dados combinados de **$\_GET**, **$\_POST** e **$\_COOKIE**.
* **Características:**
  + Útil para capturar dados de qualquer tipo de requisição HTTP.
  + Pode ser confuso em situações onde os nomes de parâmetros em **GET** e **POST** colidem.
* Texto

  Descrição gerada automaticamente**Exemplo:**   
  URL: <http://example.com?nome=Joao>  
    
  Formulário com POST:

Interface gráfica do usuário, Diagrama

Descrição gerada automaticamente com confiança média

**$http\_response\_header**

O $http\_response\_header é uma **variável superglobal** disponível no PHP que armazena os **cabeçalhos da resposta HTTP** recebidos após a execução de funções que fazem requisições HTTP, como file\_get\_contents() ou fopen().

***Características Importantes***

1. **Quando é Populada**:
   * Esta variável é preenchida automaticamente **após uma requisição HTTP** ser feita usando funções que leem URLs remotas (como file\_get\_contents, fopen com uma URL, ou file()).
   * Se nenhuma requisição HTTP for feita, a variável não estará disponível.
2. **Tipo de Dados**:
   * É um **array associativo**, onde cada índice contém uma linha do cabeçalho de resposta.
3. **Escopo**:
   * Apesar de ser uma variável "superglobal", ela **não está disponível por padrão em todos os contextos**. Ela só é preenchida no escopo onde a requisição HTTP foi feita.

**Texto

Descrição gerada automaticamente**Texto

Descrição gerada automaticamente

**$\_COOKIE**

$\_COOKIE é uma superglobal no PHP que representa um array associativo contendo **os cookies enviados pelo navegador do cliente ao servidor** em cada requisição HTTP. Esses cookies são armazenados no navegador do cliente e enviados automaticamente com cada solicitação subsequente ao mesmo domínio.

Cookies são pequenos pedaços de dados (pares **chave-valor**) que permitem a persistência de informações entre requisições HTTP. Como o HTTP é um protocolo sem estado, os cookies são amplamente utilizados para gerenciar sessões, rastrear preferências do usuário, autenticação, e muito mais.

A diferença entre $\_COOKIE e $\_SESSION está em **como e onde os dados são armazenados e manipulados**.

* Texto

  Descrição gerada automaticamente**Exemplo:** Definindo um cookie:

Logotipo

Descrição gerada automaticamente

***Quando Usar Cada Um***

**Use $\_COOKIE quando:**

1. Os dados precisam ser **persistentes** entre visitas (ex.: preferências do usuário, idioma, carrinho de compras que permanece após sair do site).
2. Não há dados sensíveis sendo armazenados (dados de autenticação, por exemplo, não devem ir em cookies).

**Use $\_SESSION quando:**

1. Precisa armazenar **dados temporários** que só precisam existir enquanto o usuário estiver navegando no site.
2. Os dados são **sensíveis** (ex.: login do usuário, tokens de segurança).
3. O tamanho dos dados é maior do que o limite permitido pelos cookies.

## Por que usar $\_COOKIE quando já temos $\_SESSION?

1. **Persistência entre visitas**:
   * **Sessões são temporárias**: Elas geralmente duram enquanto o navegador está aberto ou até que o servidor expire a sessão (padrão de 24 minutos sem atividade). Se o usuário fechar o navegador, a sessão é perdida.
   * **Cookies podem ser persistentes**: Eles permitem armazenar informações que permanecem no navegador do cliente mesmo depois que a sessão acaba ou o navegador é fechado.
   * **Exemplo prático**:
     + Se você quiser "lembrar" o usuário na próxima visita ao site (como o recurso "Mantenha-me conectado"), você precisará usar cookies para salvar algo como o ID do usuário ou um token de autenticação.
2. **Redução de carga no servidor**:
   * Sessões armazenam dados no servidor, ocupando recursos como memória ou espaço em disco.
   * Cookies, por outro lado, são armazenados no navegador do cliente e **não consomem recursos do servidor** (além do processamento ao recebê-los em cada requisição).
   * **Quando usar**: Quando você tem muitas informações "não críticas" (como preferências do usuário) e quer evitar sobrecarregar o servidor.
3. **Acessibilidade entre diferentes sites/subdomínios**:
   * Sessões são restritas ao domínio e dependem do cookie PHPSESSID.
   * Cookies podem ser configurados para funcionar **entre subdomínios ou até diferentes sites**. Isso é útil em redes de sites que compartilham informações entre si.
   * **Exemplo prático**:
     + Em um site principal e um blog em subdomínios diferentes, cookies podem "sincronizar" preferências como o idioma do usuário.
4. **Controle e configuração no lado do cliente**:
   * Cookies podem ser acessados e manipulados diretamente pelo cliente (navegador). Isso permite que os dados sejam utilizados por scripts de front-end (como JavaScript), algo que sessões não permitem.
   * **Exemplo prático**:
     + Um cookie pode ser lido pelo JavaScript para ajustar o tema do site (claro ou escuro) antes mesmo de o PHP carregar a página.
5. **Não depende de sessões ativas**:
   * Sessões precisam ser iniciadas com session\_start(). Se a sessão não foi iniciada ou já expirou, seus dados são perdidos.
   * Cookies, por outro lado, são transmitidos com cada requisição HTTP, independentemente de o servidor ter iniciado ou não uma sessão.
   * **Exemplo prático**:
     + Mesmo que o usuário não esteja logado ou que a sessão tenha expirado, você ainda pode mostrar um aviso "Bem-vindo de volta, Moisés!" com base em um cookie persistente.

***Exemplo de Uso Combinado***

Você pode usar $\_SESSION e $\_COOKIE juntos para aproveitar as vantagens de ambos.

1. **Lembrar o usuário logado:**
   * Salve o ID do usuário em um cookie para persistência.
   * Texto

     Descrição gerada automaticamenteUse a sessão para armazenar informações mais detalhadas e sensíveis.
2. **Texto

   Descrição gerada automaticamenteResgatar dados do cookie e reativar sessão:**