En **PHP**, GET y POST son **métodos HTTP** que se usan para enviar datos desde un cliente (navegador) a un servidor.
Ambos se acceden en PHP mediante **superglobales**: $_GET y $_POST.

---

1. Diferencias principales

Característica	GET	POST
Dónde viajan los datos	En la URL (?clave=valor)	En el cuerpo de la petición HTTP
Visibilidad	Visible en la barra de direcciones	No visible en la URL
Tamaño de datos	Limitado (depende del navegador/servidor, ~2000 caracteres)	Mucho mayor (depende de configuración php.ini)
Seguridad	Menos seguro (datos expuestos en URL y logs)	Más seguro para datos sensibles (aunque se recomienda HTTPS)
Uso típico	Consultas, filtros, navegación, enlaces compartibles	Formularios con datos sensibles, envíos grandes, operaciones que modifican datos
Cacheable	Sí (puede ser almacenado en caché y en historial)	No (por defecto)

---

2. Ejemplo en PHP

HTML con GET

Html

```
<form action="procesar.php" method="get">
   <label>Nombre:</label>
   <input type="text" name="nombre">
   <input type="submit" value="Enviar por GET">
</form>
```

HTML con POST

Html

```
<form action="procesar.php" method="post">
   <label>Nombre:</label>
   <input type="text" name="nombre">
```

```
    <input type="submit" value="Enviar por POST">
</form>
```

---

procesar.php

Php

```
<?php
// Validar si se recibieron datos por GET
if ($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['nombre'])) {
    $nombre = htmlspecialchars($_GET['nombre']); // Evitar XSS
    echo "Recibido por GET: " . $nombre;
}

// Validar si se recibieron datos por POST
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['nombre'])) {
    $nombre = htmlspecialchars($_POST['nombre']); // Evitar XSS
    echo "Recibido por POST: " . $nombre;
}
?>
```

---

3. Buenas prácticas

- **Usar GET** para acciones **idempotentes** (que no cambian datos), como búsquedas o filtros.
- **Usar POST** para acciones que **modifican datos** o envían información sensible.
- Siempre **validar y sanitizar** la entrada (htmlspecialchars, filter_input, etc.).
- Usar **HTTPS** para proteger datos, incluso con POST.
- Evitar enviar contraseñas o datos críticos por GET.
