Unidad 3 - Gestión de formularios y enlaces

Introducción

En HTML existen dos principales métodos para interactuar con los usuarios:

Enlaces (`<a>`)

Permiten la navegación entre páginas de una aplicación web.

```
<a href="url" target="destino">...</a>
```

- `url` → `url_clasica?nombre=valor[&nombre_N=valor_N...]`
- `destino` → Dónde abrir el documento vinculado (ej. `_blank` para una nueva ventana).

Formularios (`<form>`)

Permiten a los usuarios proporcionar información a la aplicación web.

```
<form action="url" method="metodo" target="destino">...</form>
```

- `metodo` → Puede ser `GET` o `POST`.

Recuperar los datos en la aplicación

Datos proporcionados a través de la URL

Se almacenan en la variable global `$_GET`, una matriz asociativa donde:

- Las claves corresponden al nombre de los parámetros.
- Los valores son los datos enviados.

Datos proporcionados mediante formularios

Se almacenan en las matrices globales `$_GET` o `$_POST`, dependiendo del método usado en el formulario:

- **Clave**: Nombre del campo (atributo `name`).
- **Valor**: Información ingresada por el usuario.

Elementos de un formulario

Un formulario puede incluir los siguientes elementos:

- `<input>` con distintos tipos (`type`):
  - `text`, `password`, `radio`, `checkbox`, `hidden`, `image`, `submit`, `reset`, `button`, `file`
- `<textarea>`
- `<select>`

Problemas y soluciones con los datos recuperados

Problemas comunes

- **Representación de los campos de texto**: Usar constructos como `value="<?php echo $dato;?>"` puede ser problemático.
- **Seguridad del sitio web**: Riesgo de inyección de código malicioso.

Soluciones

1. **Uso del método** `POST`: Evita exponer datos en la URL.
2. **Sanitización de datos**: Uso de funciones como `htmlspecialchars()` para convertir caracteres especiales en entidades HTML:
   - `$` → `&`
   - `"` → `&quot;`
   - `'` → `&#039;`
   - `<` → `<`
   - `>` → `>`
   - Aplicación de filtros.
3. **Validación de datos**: Mediante expresiones regulares y funciones específicas.

```
<form action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>" method
="post">
```

Utilización de filtros

PHP dispone de una extensión para filtrar y sanear datos de entrada.

Constante	Propósito
`FILTER_VALIDATE_BOOLEAN`	Valida un booleano
`FILTER_VALIDATE_EMAIL`	Valida una dirección de correo
`FILTER_VALIDATE_FLOAT`	Valida un número decimal
`FILTER_VALIDATE_INT`	Valida un entero
`FILTER_VALIDATE_IP`	Valida una dirección IP
`FILTER_VALIDATE_REGEXP`	Valida con una expresión regular
`FILTER_VALIDATE_URL`	Valida una URL
`FILTER_SANITIZE_EMAIL`	Elimina caracteres no válidos en email
`FILTER_SANITIZE_ENCODED`	Elimina caracteres especiales

```
filter_var($dato, constante_filtro, opciones);
```

- Devuelve los datos filtrados o `FALSE` si el filtro falla.

Control de datos recuperados

Al manejar datos ingresados por el usuario, se recomienda:

1. **Limpieza de datos**:
   - Uso de funciones como `trim()` para eliminar espacios no deseados.
2. **Validación**:
   - **Datos obligatorios**: `empty()`, `isset()`
   - **Longitud máxima**: `strlen()`
   - **Formato**: Expresiones regulares y `preg_match()`
   - **Tipo de dato**: Validadores específicos.
3. **Tratamiento del dato**:
   - Si los datos son correctos, se procesan.
   - Si no, se muestra un mensaje de error o se solicita corregirlos.

Expresiones regulares (RegEx)

Símbolo	Significado
`/`	Delimitador de un patrón (marca el inicio y el final del patrón)
`^`	Inicio del `string`
`$`	Final del `string`
`[]`	Rango de valores permitidos
`()`	Subpatrón obligatorio
`|`	Elección
`{}`	Repeticiones (e.g., `{3}`, `{1,5}`)
`*`	0 o más repeticiones
`+`	1 o más repeticiones
`?`	0 o 1 repetición
`'`	Cualquier carácter
`\d`	Un dígito
`\`	Escapar un metacaracter
`!`	Permite que el patrón ignore entre mayúsculas y minúsculas. Debe colocarse después del delimitador final.

Redirección a otra página

En el modelo cliente-servidor, la comunicación se basa en mensajes HTTP. Para redirigir al usuario, se pueden usar:

- **Etiquetas HTML**:
  - `<a href="...">...</a>` o formularios con `action`.
- **PHP con `header`**:

```
header("Location: ./otraweb.php");
```

Demostración:

`redirigir.php`:

```
<?php
    $nombre = "Esther";
    echo $nombre;
    header("location: ./otraweb.ph
p");
?>
```

`otraweb.php`:

```
<?php
    echo "Hola";
?>
```

En el navegador, se pueden inspeccionar los mensajes HTTP para verificar las cabeceras utilizadas. La redirección inicia un nuevo contexto en la aplicación.