# Ajax

Envío de datos

GET y POST

# GET

- Hasta ahora hemos estado recuperando información del servidor , ahora vamos a ver como podemos mandar información al servidor.
- Depende si utilizamos GET o POST, tendremos que hacer cosas distintas.

# Enviar mediante GET

- Vamos a utilizar en los ejemplos PHP.
- Hay varias formas de pasar datos al servidor. Una de ellas es pasar las variables por URL, Esta es la forma que se emplea con GET.
- Aquí al escribir la ruta en el método open, escribimos también las variables que vamos a pasar

# Sintaxis 1

open("GET","archivo.php?nombre=valor",true);

- **archivo.php** : ruta del archivo PHP.
- **?** : Signo que indica que enviamos algo después del archivo.
- **nombre=valor** : "nombre de la variable y su valor, separados por el signo igual.

En este ejemplo el nombre de la variable y su valor tienen valores fijos.

# Sintaxis 2

- url="archivo.php?enviaEmail="+varemail;
- open ("GET",url,true);

- En este caso el valor tiene un valor variable y por lo tanto le sacamos de las comillas.

# Sintaxis 3

```
ruta="archivo.php";
envio1="enviarEmail="+varemail;
envio2="enviarTelefono="+vartelefono;
envio3="enviarContras="+varcontras;
url=ruta+"?"+envio1+"&"+envio2+"&"+envio3;
open ("GET",url,true);
```

Si queremos pasar mas de una variable con GET.

# Práctica

- Tenemos un archivo php llamado php1, que nos valida los datos de un formulario que vamos a crear ,como respuesta nos devuelve un texto.
- Creamos un formulario para introducir un email y la contraseña dos veces. Pondremos un botón de envío y después recogeremos los datos y con Ajax los mandamos al servidor y recogemos y visualizamos la respuesta.

# GET vs POST

- Con GET al pasar los datos con la URL éstos pueden mostrarse en la barra del navegador al ser transferidos y quedan reflejados en el historial del navegador, por lo que pueden ser vistos por cualquiera que abra la página.
- Además el método GET es más lento.
- Esto en archivos de pequeño tamaño, como los que estamos usando, no se nota, pero si hay que transferir una gran cantidad de datos deberemos usar el método POST.
- Si queremos hacer un trabajo profesional, donde se guarde la privacidad de los datos, y la carga de archivos se haga de forma rápida deberemos elegir el método POST.

| GET | | POST |
|---|---|---|
| SI | Los datos son visibles en la url | NO |
| SI | Los datos pueden permanecer en el historial del navegador | NO |
| SI | Una url puede ser guardada conteniendo parámetros de un envío de datos | NO |
| SI | Existen restricciones en la longitud de los datos enviados | NO |
| No (los datos además de ser visibles pueden quedar almacenados en logs) | Se considera preferible para envío de datos sensibles (datos privados como contraseñas, números de tarjeta bancaria, etc.) | Sí (sin que esto signifique que por usar post haya seguridad asegurada) |

| | | |
|---|---|---|
| Sí (sólo admite caracteres ASC-II) | Restricciones de tipos de datos | No (admite tanto texto como datos binarios p.ej. archivos) |
| Sí | Riesgo de cacheado de datos recuperados en los navegadores | No |
| Si | Posibles ataques e intentos de hackeo | No |

# USAR POST

## 1-Recoger los datos en una cadena :

- Al igual que con GET recogemos los datos en una cadena. Cada dato consta de un nombre y un valor separados por el signo igual. Cada dato va separado del anterior por el signo ampersand (&).
- La **diferencia con GET** es que la cadena con los datos no la pasamos por la URL sino que la enviaremos como parámetro del método send().

# USAR POST

**2-Cambiar las cabeceras :** Para que el método POST sea admitido debemos cambiar las cabeceras, enviando nuevas cabeceras mediante el método **setRequestHeader().**

El código para cambiar las cabeceras es siempre el mismo, lo pondremos después del método open() y antes del método send() y enviamos tres cabeceras:

- 1ª.- objetoAjax.setRequestHeader("Content-type", "application/x-www-form-urlencoded");

  Esta cabecera indica el tipo de contenido: los datos enviados deben tratarse como los extraídos de un formulario.
- 2ª.- objetoAjax.setRequestHeader("Content-length", cadenadatos.length);

  Indica el número de datos que vamos a transmitir.

  De ahí en el segundo parámetro cadenadatos debe ser la variable donde se guardan los datos para enviar.
- 3ª.- objetoAjax.setRequestHeader("Connection", "close");

  Cierra la conexión del envío de cabeceras.

# USAR POST

- **Forma de envio** :

  los datos se envian como el primer parámetro del metodo "send": **send(cadenadatos).**

  En el método open() indicaremos como primer parámetro "POST" y como segundo la ruta del archivo sin añadirle nada más.

# FormData

- var datos=new FormData();
- datos.append('nombre','Juan');
- datos.append('apellido','Perez');

- solicitud.open("POST", url, true);
- solicitud.send(datos);

# Práctica

- Con el mismo ejemplo anterior vamos a modificar el programa anterior para enviar con POST.
- Retocar el php , cambiando la recepción de datos GET por POST.

# Acentos

- Archivos xml

<?xml version="1.0" encode="ISO-8859-1"?>

- Archivos de texto

  Guardar como "UTF-8".

- Archivos php

```
header("Content-Type: text/plain; charset=iso-8859-1");
```