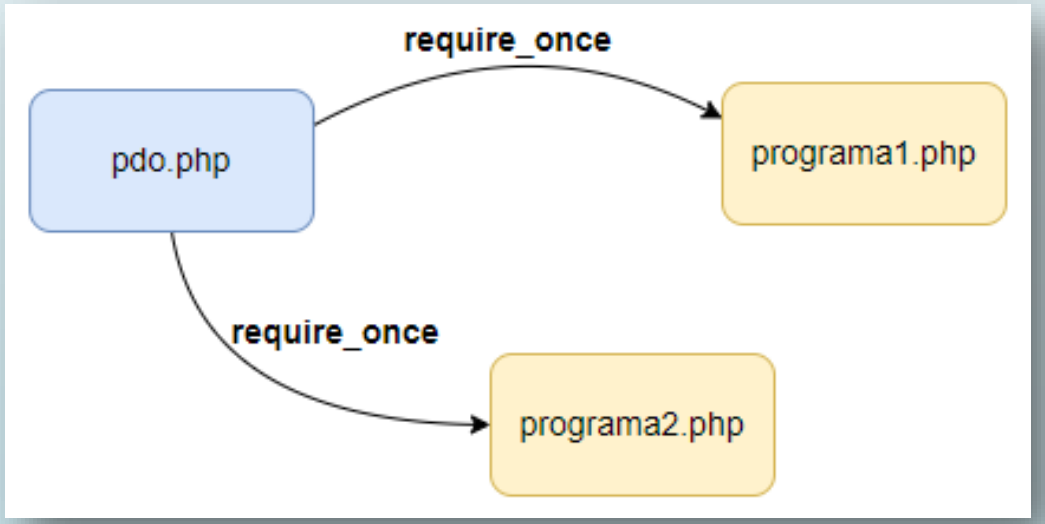
TEMA 4.3

Insertando y borrando datos

Patrón de conexión a BD

- Siempre que hagamos una aplicación con PHP y MySQL, tendremos que iniciar una conexión a la BD
- PHP pierde las conexiones cuando termina de ejecutarse un programa php, por tanto, en cada programa php de mi aplicación necesitaré volver a conectarme antes de acceder a la BD
- Patrón: Hacer un programa php con las sentencias de la conexión a BD, y luego incluirlo en todos los programas php que necesiten la conexión.
- Así, tendré un fichero pdo.php, donde incluiré todo lo necesario para establecer mi conexión



Código de la conexión

- El código para la conexión es el siguiente:
- Creamos una conexión, un objeto de tipo PDO, indicando dominio del servidor, puerto, nombre de la base de datos, usuario y contraseña
- Como hay que gestionar posibles errores de conexión, estableceremos la forma de hacerlo, ya que PDO ofrece varias.
- Utilizamos el método **setAttribute(parámetro, valor)**, para establecer el valor del parámetro "ATTR_ERRMODE"
- Indicaremos que se quiere la opción "ERRMODE_EXCEPTION", que implica que un error generará una excepción a capturar (con try...catch, ya lo veremos)

pdo.php

```php
<?php
// Conexión a la BD martabd
$pdo=new PDO("mysql:host=localhost;port=3306;dbname=martabd",
    'marta','marta');
    // Control de errores
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
?>
```

Inserción de datos en la base de datos

- Para dar de alta información en una base de datos, tendremos típicamente un formulario para que el usuario escriba los datos.
- El usuario utilizará el formulario y finalmente confirmará el alta mediante el botón de submit.
- Nuestro programa tiene que hacer lo siguiente:
  1. **Validar la información** de entrada: Aunque el navegador pueda validar algo, tenemos que contemplar una segunda validación en servidor.
  2. **Escribir una sentencia SQL** de inserción de información: Crearemos un string que contendrá el **INSERT**.
  3. **Preparar la sentencia** antes de ejecutarla. PDO prepara las sentencias de forma que:
     1. Previene la inyección de SQL
     2. Optimiza el rendimiento de sentencias que se ejecutan múltiples veces con distintos parámetros
  4. **Ejecutar la sentencia** SQL mediante la función **execute** de PDO

Código: Insertando registros (1)

MODELO - CONTROLADOR

Transp-43-1-insert.php

```php
<?php
    session_start();
    require_once "pdo.php";
    if (isset($_POST["nombre"]) && isset($_POST["email"]) && isset($_POST["password"]))
    {
        // Preparo el string con el INSERT: Utilizo placeholders
        $sql = "INSERT INTO usuarios (nombre, email, password)
                VALUES (:nombre, :email, :password)";
        // Para mostrar luego
        $_SESSION["mensaje"] = $sql;
        // Le digo a PDO que prepare la sentencia
        $resultado = $pdo->prepare($sql);
        // Le digo a PDO que ejecute la sentencia, sustituyendo los placeholders
        $resultado->execute( [
            ':nombre' => $_POST['nombre'],
            ':email' => $_POST['email'],
            ':password' => $_POST['password']
        ]);
        header("Location: Transp-43-1-insert.php");
        return;
    }
?>
```

- Si entra la request con un POST, realizamos la inserción
- **$sql**: Contiene un string con la sentencia sql.
- **Placeholders**: No se concatenan directamente los valores (NUNCA). En cambio, se indica un parámetro, poniendo delante ":". Este parámetro es posicional, lo gestiona PDO posteriormente.
- **$resultado**: Es un objeto tipo PDOStatement (sentencia PDO).
- **prepare**: Método de PDO que prepara la sentencia para optimizar su ejecución y para prevenir inyección de SQL
- **execute**: Método de PDO que ejecuta una sentencia. Como parámetro recibe un array asociativo donde los placeholders son la clave y el valor los valores que queremos que estén en nuestro SQL. Aquellos que NUNCA concatenamos en $sql.
- Por supuesto, utilizamos el patróns **POST-redirect-GET** para que no se inserte el registro varias veces con un refresco de la página!!

Código: Insertando registros (2)

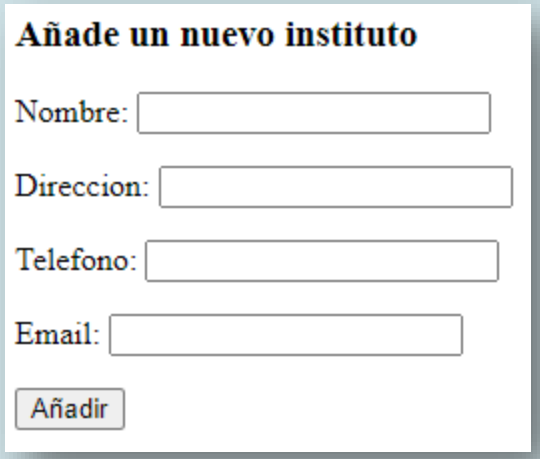
VISTA

```php
<?php
  $mensaje = isset($_SESSION["mensaje"]) ? $_SESSION["mensaje"] : "";
  unset($_SESSION["mensaje"]);
?>
<!DOCTYPE html>
<html>
<head>
  <title>Insertando</title>
</head>
<body>
  <pre><?=$mensaje?></pre>
  <p>Añade un nuevo usuario</p>
  <form method="post">
    <p>
      <label for="nombre">Nombre: </label>
      <input type="text" name="nombre" id="nombre"/>
    </p>
    <p>
      <label for="email">Email: </label>
      <input type="text" name="email" id="email"/>
    </p>
    <p>
      <label for="password">Password: </label>
      <input type="text" name="password" id="password"/>
    </p>
    <p>
      <input type="submit" value="Añadir"/>
    </p>
  </form>
```

- Siguiendo el patrón **POST-redirect-GET**, la vista sólo se ejecuta la primera vez que se entra, o mediante redirect.
- **$mensaje** contendrá el valor de los mensajes que existan en la sesión, grabados por el proceso que gestionó el POST e hizo la redirección.
- El resto es un **formulario POST** que permitirá al usuario final añadir usuarios a nuestra tabla **usuarios** de la BD.

Ejercicio 1:

```sql
CREATE TABLE institutos (
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(128),
    direccion VARCHAR(128),
    telefono VARCHAR(20),
    email VARCHAR(128)
) ENGINE=InnoDB CHARSET=utf8;
```



1. Crea una tabla institutos en tu base de datos, con los campos: id, nombre, dirección, teléfono, email.
2. La clave primaria es id, valor numérico con autoincremento.
3. Crea un formulario PHP que permita al usuario añadir institutos a la base de datos. Todos los campos serán de tipo "text"
4. Antes de añadir la información, deberás validar los campos:
    1. Todos los datos estarán informados
    2. El email debe llevar una "@"
    3. El teléfono tendrá al menos 9 dígitos.
5. Si no se cumple alguna validación, no se insertará el registro y se mostrará un mensaje en color rojo, indicando la causa.
6. Emplea el patrón POST-redirect-GET-Flash

Mostrando la información que voy almacenando

```php
<body>
  <!-- Muestro los registros existentes en una tabla HTML-->
  <h1>Usuarios</h1>
  <?php
    echo "<table border='1'>";
    $query = "SELECT nombre, email, password from usuarios";
    $resultado = $pdo->query($query);
    while ( $fila = $resultado->fetch(PDO::FETCH_ASSOC) )
    {
      echo "<tr>";
      echo "<td>" . $fila["nombre"] . "</td>";
      echo "<td>" . $fila["email"] . "</td>";
      echo "<td>" . $fila["password"] . "</td>";
      echo "</tr>";
    }
    echo "</table>";
  ?>
  <!-- Muestro el formulario de nuevo usuario -->
  <pre><?=$mensaje?></pre>
  <p>Añade un nuevo usuario</p>
  <form method="post">
```

- Podemos, en el mismo programa donde inserto, mostrar una lista con los registros almacenados.
- Para ello, mantenemos el código anterior, y añadimos en la parte de vista una query a la tabla para recuperar los registros y mostrarlos.

Usuarios

Marta Olmedilla	martaO@europa.es	1a52e17fa899cf40fb04cfc42e6352f1
alumno	alumno@europa.es	alumno
admin	admin@insti.org	admin
pepe	pepe@insti.org	pepe123
Carmen	mamen@eu.es	carmenchu

Añade un nuevo usuario

Nombre:

Email:

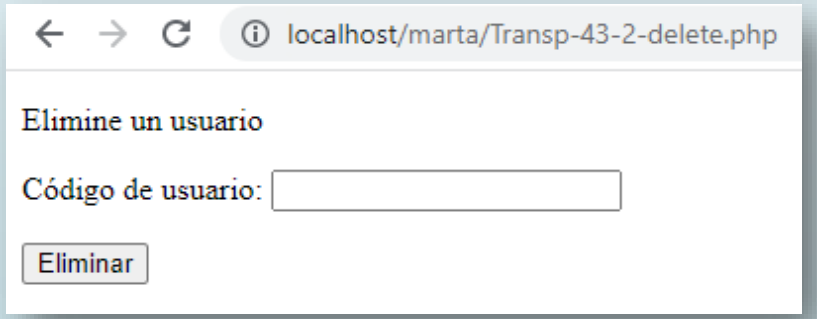
Password:

Añadir

Eliminar un registro de la base de datos

- Podemos eliminar un registro de la base de datos:
  1. Mediante un formulario que permite indicar al usuario un código y eliminarlo.
  2. Integrando un botón de "eliminar" en el propio formulario de consulta e inserción de información
- La primera opción es muy sencilla, pero poco amigable al usuario:

```php
<?php
  session_start();
  require_once "pdo.php";
  if (isset($_POST["usuario_id"]))
  {
    if (empty($_POST["usuario_id"])) {
      $_SESSION["mensaje"] = "Debe indicar un código de usuario";
    } else {
      $sql = "DELETE FROM usuarios WHERE usuario_id = :usuario_id";
      $resultado = $pdo->prepare($sql);
      $resultado->execute(array( ':usuario_id' => $_POST['usuario_id'] ));
    }
    header("Location: Transp-43-2-delete.php");
    return;
  }
?>
```



Añadir y eliminar

Podemos modificar el código de añadir usuario, de forma que en el mismo programa:

- Visualice todos los usuarios
- Pueda eliminar uno de ellos
- Pueda añadir nuevos usuarios

Para hacer el botón borrar, hemos introducido un <td> al final, con un formulario tipo post con los campos:

- Un input oculto con el identificativo del usuario
- Un botón submit, con el nombre "borrar" y el valor "Borrar"

Código del programa insertar-eliminar: Formulario

Añadiremos un formulario en la tabla de visualización de usuarios, en la última celda, con un campo oculto cuyo valor es el identificativo del usuario

```php
<?php
  echo "<table border='1'>";
  $query = "SELECT nombre, email, password, usuario_id from usuarios";
  $resultado = $pdo->query($query);
  while ( $fila = $resultado->fetch(PDO::FETCH_ASSOC) )
  {
    echo "<tr>";
    echo "<td>" . $fila["nombre"] . "</td>";
    echo "<td>" . $fila["email"] . "</td>";
    echo "<td>" . $fila["password"] . "</td>";
    echo "<td>";
    echo '<form method="post">' .
              '<input type="hidden" name="usuario_id" value="'. $fila["usuario_id"].'"/>' .
              '<input type="submit" value="Borrar" name="borrar"/>'.
            '</form>';
    echo "</td>";
    echo "</tr>";
  }
  echo "</table>";
?>
```

- Cada fila de la tabla tiene ahora un formulario en la última celda
- El formulario contendrá dos input:
  1. **usuario_id**: Campo oculto cuyo valor es el usuario_id de esa fila
  2. **Borrar:** Botón visible tipo submit
- Si el usuario hace click en un botón, el navegador realiza un request de tipo POST con los parámetros:
  - usuario_id="NNN"
  - borrar="Borrar"

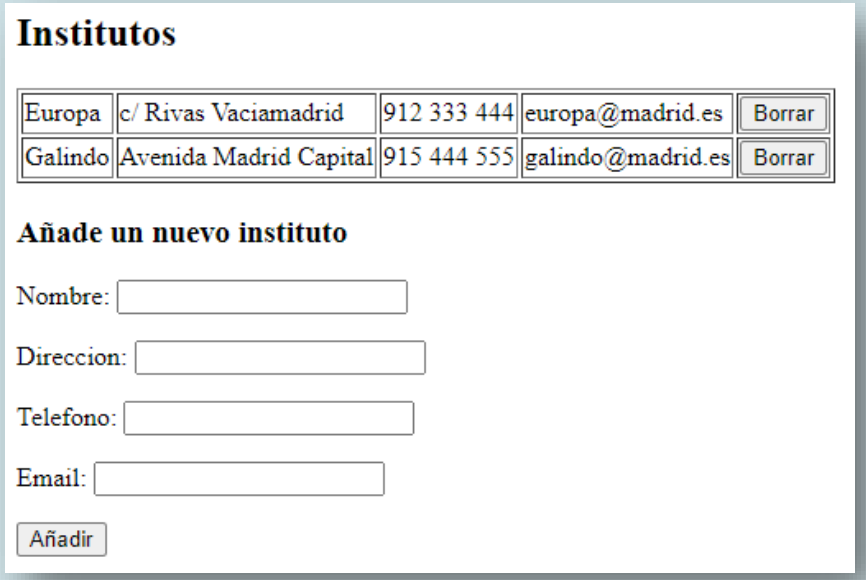
Código del programa insertar-eliminar: Eliminación

Añadiremos, después de la gestión del POST de inserción, código adicional para gestión del POST de borrado.

```php
// Gestión del formulario de eliminación de datos
if (isset($_POST["borrar"]) && isset($_POST["usuario_id"])) {
  $sql = "DELETE FROM usuarios WHERE usuario_id = :usuario_id";
  $resultado = $pdo->prepare($sql);
  $resultado->execute(array( ":usuario_id" => $_POST["usuario_id"]));
  $_SESSION["mensaje"] = "Usuario " .  $_POST["usuario_id"] . " eliminado";
  header("Location: Transp-43-3-ins-del.php");
  return;
}
```

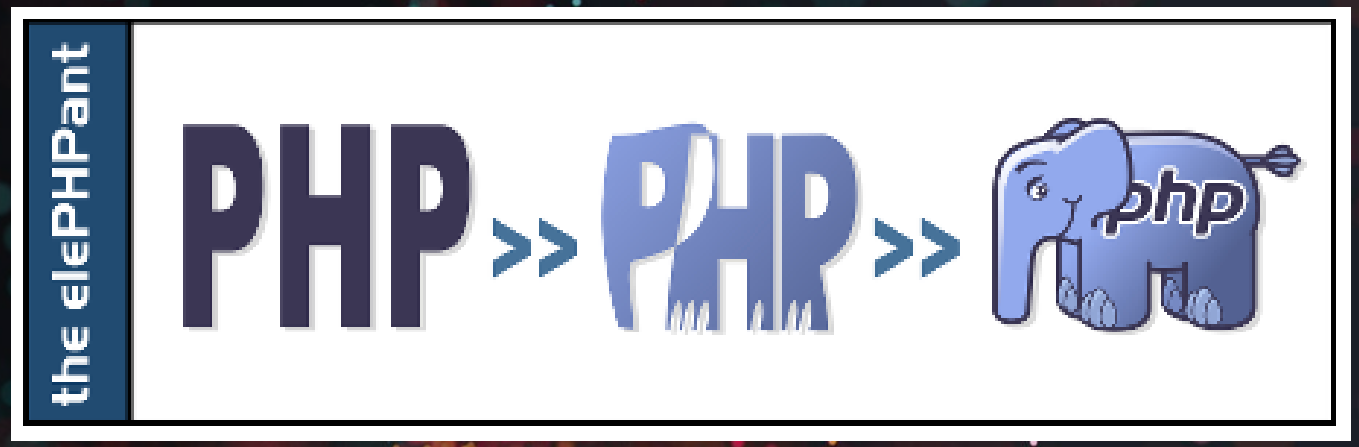
- Este código se ejecutará si se recibió un REQUEST de tipo POST, con los parámetros "borrar" y "usuario_id", es decir, si el usuario hizo click en alguno de los botones "Borrar".
- Dentro preparamos una sentencia SQL de eliminación de un registro de usuario: Aquel cuyo usuario_id tenga como valor el enviado con el POST
- Por supuesto, mantenemos el patrón POST-redirect-GET

Ejercicio 2:



Modifica el programa que hiciste en el Ejercicio 1 de estas transparencias, para que además de insertar registros:

1. Muestre una tabla con los registros existentes
2. Añada a la derecha de cada registro existente un botón de eliminar
3. Para este ejercicio debes seguir utilizando el patrón POST-redirect-GET-flash en todos los formularios



https://www.php.net/docs.php

https://www.php.net/manual/es/book.pdo.php