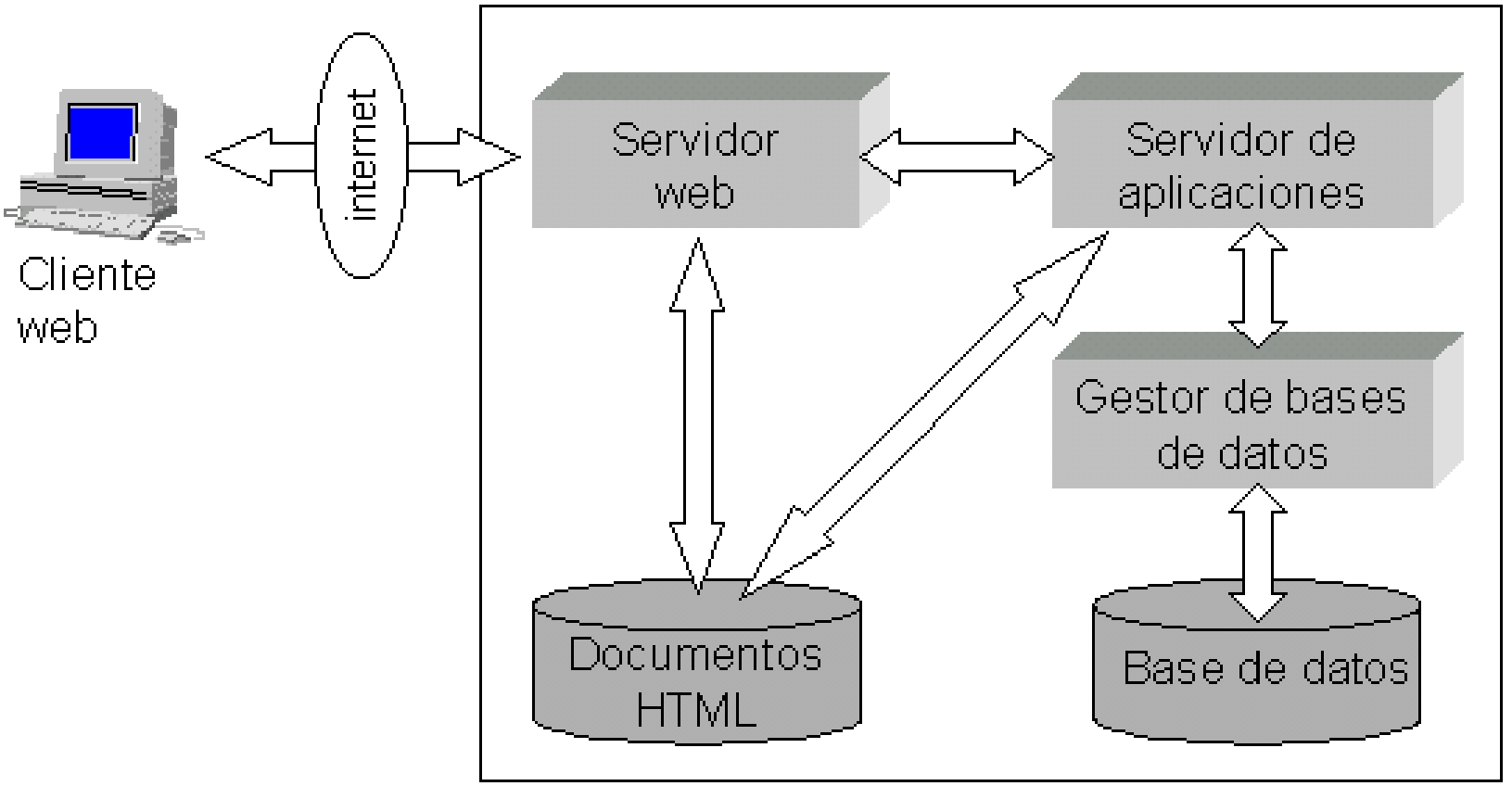
3.5 Acceso a bases de datos

Bases de datos: Introducción

- Las bases de datos permiten almacenar de una forma estructurada y eficiente toda la información de un sitio web.
- Ventajas :
  - Proporcionar información actualizada
  - Facilitar la realización de búsquedas
  - Disminuir los costes de mantenimiento
  - Implementar sistemas de control de acceso
  - Almacenar preferencias de los usuarios

Aplicación web apoyada en una Base de Datos : Diseño



Acceso a bases de datos en PHP

- PHP ofrece una gran variedad de librerías para el acceso a bases de datos:
  - MySQL
  - PostgreSQL
  - Oracle
  - MS SQL Server
  - Sybase
  - …
  - SQLite
  - ODBC
- Cada fabricante tiene su propia API:
  - Al elegir la base de datos no podremos cambiar de fabricante fácilmente.
- Existe el proyecto PEAR:DB para ofrecer un interfaz común y otros paquetes de abstracción como ADOdb, dbx, etc. a cambio de ¿una pérdida de rendimiento? http://phplens.com/lens/adodb/

MySQL

- **Características de MySQL**
  - Modelo relacional, multiusuario
- **Tipos de datos**
  - Numéricos
    - tinyint, smallint, mediumint, int, integer, bigint
    - decimal, float, numeric
  - Fecha y hora
    - date, time, datetime, year, timestamp
  - Cadena
    - char, varchar
    - tinytext, text, mediumtext, longtext
    - tinyblob, blob, mediumblob, longblob
    - enum, set
  - Debe elegirse adecuadamente el tipo y el tamaño de cada campo
- Referencia recomendada: http://dev.mysql.com/doc/refman/5.0/es/tutorial.html

Administración de MySQL: PHPMyAdmin

- **phpMyAdmin** es una herramienta para la administración del servidor de bases de datos MySQL.
- Dispone de una interfaz gráfica y es de libre distribución.
- Permite realizar todo tipo de operaciones sobre bases de datos:
  - crear, borrar y modificar tablas
  - consultar, insertar, modificar y eliminar datos
  - definir usuarios y asignar permisos
  - realizar copias de seguridad
  - etc.
- Está escrita en PHP y se ejecuta desde el navegador.
- Si está instalada en la carpeta phpmyadmin, se ejecuta escribiendo en la barra de direcciones del navegador la url

  http://localhost/phpmyadmin/
- Puede administrar bases de datos locales y remotas.

Procedimiento para el acceso a una BD en una página PHP

- Los pasos para acceder desde PHP a una base de datos son los siguientes:
  1. Conectar con el servidor de bases de datos.
  2. Seleccionar una base de datos.
  3. Enviar la instrucción SQL a la base de datos.
  4. Obtener y procesar los resultados.
  5. Cerrar la conexión con el servidor de bases de datos.

Funciones de MySQL en PHP

- Las funciones concretas de MySQL que realizan estas operaciones son:
  - Conectar con el servidor de bases de datos:
    - **mysql_connect()**
  - Seleccionar una base de datos:
    - **mysql_select_db()**
  - Enviar la instrucción SQL a la base de datos:
    - **mysql_query()**
  - Obtener y procesar los resultados:
    - **mysql_num_rows(), mysql_fetch_array() y mysql_affected_rows()**
  - Cerrar la conexión con el servidor de bases de datos:
    - **mysql_close()**
- Adicionalmente se pueden usar:
  - Obtención del mensaje de error:
    - **mysql_error()**

Conexión/Desconexión a una base de datos

- **mysql_connect(servername, username, password)**:
  - Devuelve el manejador de la conexión o falso en caso de error.
  - servername: Opcional. Indica el servidor al que nos contectamos. Por defecto es "localhost:3306“
  - username: Opcional. Indica el usuario con el que vamos a iniciar la sesión en el servidor. Por defecto se emplea el nombre de usuario establecido en la propiedad mysql.default_user.
  - password: Optional. Indica la contraseña. Por defecto es la cadena vacía.
- **mysql_close(connection)**:
  - Devuelve verdadero si to es correcto. Falso en otro caso.
  - Connection: Opcional. Indica la conexión cerrar. Si no se especifica se usa por defecto la última conexión creada por mysql_connect().

- Siempre se ha de cerrar una conexión abierta:
  - Si no se hace se puede llegar al límite de conexiones (lo que puede significar una sobrecarga) y provocar fallo.

Ejemplo: Conexión/Desconexión a una base de datos

```
<?php
   $con = mysql_connect("localhost","peter","abc123");
   if (!$con) {
        die(' No puedo conectar: ' . mysql_error());
   }
   // Continua el procesamiento
   …
   mysql_close($con);
?>
```

Seleccionar una base de datos

- **mysql_select_db(database,connection)**:
  - Devuelve verdadero si todo ha ido bien. Falso en caso contrario.
  - Database: Indica el nombre de la base de datos que queremos solucionar.
  - Connection: Opcional. Indica la conexión sobre la que trabajaremos. Si no se especifica se emplea la última conexión abierta.

Ejemplo: Seleccionar una base de datos

```
<?php
    $con = mysql_connect("localhost","peter","abc123");
    if (!$con) {
          die(' No puedo conectar: ' . mysql_error());
    }
    $db_selected = mysql_select_db("test_db", $con);
    if (!$db_selected) {
          mysql_close($con);
          die (“No puedo usar la base de datos: " . mysql_error());
    }
    // Continua el procesamiento
    …
    mysql_close($con);
?>
```

Envío de instrucciones SQL a la base de datos

- Independientemente del tipo de instrucción.

- **mysql_query(query,connection)**:
  - Devuelve el conjunto de resultado si todo ha ido bien (Select) o verdadero (resto de SQL). Falso en caso contrario.
  - query: Consulta a enviar (no debe terminar con punto y coma).
  - connection: Opcional. Indica la conexión sobre la que trabajaremos. Si no se especifica se emplea la última conexión abierta.

Ejemplo: Envío de instrucciones SQL a la base de datos

```
<?php
   $con = mysql_connect("localhost","peter","abc123");
   … Comprobar conexión …
   $db_selected = mysql_select_db("test_db", $con);
   … Comprobar  selección de bd …
   $result = mysql_query("SELECT * FROM person“,$con);
   if (!$result) {
         die (“Error al ejecutar la consulta: " . mysql_error());
         }
   … Procesar los resultados …
   mysql_close($con);
?>
```

Procesamiento de resultados de instrucciones Select

- Sentencia de selección:
  - **mysql_num_rows(data)**:
    - Devuelve el número de filas en un conjunto de resultados.
    - data: Conjunto de resultados devuelto por *mysql_query*.
  - **mysql_fetch_array(data)**:
    - Devuelve (sucesivamente) como vector asociativo y numérico una fila de un conjunto de resultados y falso cuando ya no haya más filas.
    - data: Conjunto de resultados devuelto por *mysql_query*.
  - **mysql_free_result(data)**:
    - Libera los recursos usados para mantener un conjunto de resultados cuando ya no lo necesitamos. Su uso no es necesario pero sí recomendable para grandes resultados (ya que los recursos de los resultados se liberan automáticamente cuando termina la ejecución de la página PHP).
    - data: Conjunto de resultados devuelto por *mysql_query*.

Ejemplo: Procesamiento de resultados de Instrucciones Select

```
<?php
    … Conexión y selección de bd y sus comprobaciones …
    $result = mysql_query("SELECT * FROM person“,$con);
     if (!$result) {
            die (“Error al ejecutar la consulta: " . mysql_error());
    }
    echo "<table border='1'> <tr> <th>Firstname</th> <th>Lastname</th>
    </tr>";
    while($row = mysql_fetch_array($result)) {
            echo "<tr>";
            echo "<td>" . $row['FirstName'] . "</td>";
            echo "<td>" . $row['LastName'] . "</td>";
            echo "</tr>";
            }
    echo "</table>";
    mysql_free_result($result);
    mysql_close($con);
?>
```

Procesamiento de resultados de sentencias modificadoras

- INSERT, UPDATE, DELETE, CREATE, …:
  - **mysql_affected_rows(connection)**:
    - Devuelve el número de filas que se vieron afectadas por una setencia modificadora o -1 en caso de error.
    - connection: Opcional. Indica la conexión sobre la que trabajaremos. Si no se especifica se emplea la última conexión abierta.

Ejemplo: Procesamiento de resultados de SQL Modificador

```
<?php
   … Conexión y selección de bd y sus comprobaciones …
   $result=mysql_query("DELETE FROM Person WHERE
   LastName like ‘%e%‘“,$con);
   if (!$result) {
        die (“Error al ejecutar la consulta: " . mysql_error());
   }
   $rc = mysql_affected_rows($result);
   echo “Filas borradas: $rc ";
   mysql_close($con);
?>
```

Interacción usando formularios

```
<?php
   … Conexión y selección de bd y sus comprobaciones …
   $sql="INSERT INTO person (FirstName, LastName, Age)
   VALUES
   ('$_POST[firstname]','$_POST[lastname]','$_POST[age]')";
   if (!mysql_query($sql,$con)) {
         die('Error: ' . mysql_error());
         }
   echo "1 fila añadida";
   mysql_close($con);
?>
```

Interacción usando formularios: Seguridad

- Alterando las entradas al script PHP se podrían manipular las consultas que este ejecuta sobre la base de datos.
- Las entradas desde fuentes externas deberán ser filtradas:
  - Campos de formularios:
    - GET
    - POST
  - Valores de cookies.
- Fundamentalmente habrá que filtrar los caracteres extraños.
- Muchos de los ataques se evitan configurando el servidor con el valor “magic_quotes_gpc” a “on”. Pero no podemos confiar siempre en los administradores

Ejemplo: Seguridad en formularios

EJ46

```
…Conexiones, selección y comprobación …
$sql = "SELECT * FROM usuarios WHERE usuario='$_POST[user]'
    AND contrasenya='$_POST[pass]'";
echo "<h2>$sql</h2>";
echo get_magic_quotes_gpc();

$result = mysql_query($sql,$con);
    if (!$result) {
        die ("Error al ejecutar la consulta: " . mysql_error());
    }

if(mysql_num_rows($result)!=0)
    echo '<h1>Acceso permitido</h1>';
else
    echo '<h1>Acceso denegado</h1>';
… Resto de la lógica…
```

Experimento:

- Pruebe a introducir el valor ‘ OR 1=1 # como usuario, dejando la contraseña en blanco.

```
<h1>Entrada en el sistema</h1>
<form method="post" enctype="multipart/form-data">
   User: <input name="user" type="text"/> <br />
   Password: <input name="pass" type="text"/> <br />
   <input type="submit" name="envio" value="Enviar" />
   <input type="reset" name="rest" value="Restaurar" />

</form>
```

Cómo evitamos la inyección de SQL

- Filtrando las entradas:
  - valor “magic_quotes_gpc” a “on”. No recomendado.
  - Uso de la función *addslashes* sobre todas las entradas.
  - Uso de filtros FILTER_SANITIZE_STRING

Ejemplo: Evitar inyección SQL

EJ47

```
…
$filtros = Array (
    'user'=>FILTER_SANITIZE_STRING,
    'pass'=> FILTER_SANITIZE_STRING
    );

$entradas = filter_input_array(INPUT_POST, $filtros);
echo $entradas['user'];
echo '<br/>';
echo $entradas['user'] == "' OR 1=1 #";
$sql = "SELECT * FROM usuarios WHERE
    usuario='$entradas[user]' AND contrasenya='$entradas[pass]'";
$result = mysql_query($sql,$con);
```

3.6 Cookies y sesiones

3.6.1 Cookies

- ¿Qué es una cookie?
- Manejo de cookies en PHP
- Cookies como vectores
- Experimento: Comprobar si el navegador soporta cookies

Cookies

- Pequeños trozos de información que dejamos en el navegador de quien nos visita. Cada vez que el navegador pide una página del sitio le envía la información que éste dejó en anteriormente.
- ¿Para qué?
  - Recordar información temporal: Ej. Recordar el usuario, las selecciones que hizo, carritos de la compra, etc…
  - No recomendable para información sensible:
    - Se almacena en el ordenador del usuario.
    - Puede ser fácilmente modificada para romper la seguridad.
- ¿Por qué?
  - HTTP es un protocolo sin estado. Esto significa que en una aplicación con varias páginas, al pasar de una a otra, no podremos recordar datos de lo que el usuario hizo anteriormente.
    - Soluciones:
      - Pasar datos como campos ocultos de formularios. Complica el diseño de las páginas (hay que modificar el HTML) y la forma de pasar de una página a otra. Dejar en el navegador del cliente los datos, y leerlos cuando los necesitemos. Es una solución mucho más cómoda.

Manejo de cookies en PHP

- Crear una cookie:
  - setcookie(nombre,valor[,caducidad]):
    - nombre: Obligatorio. Cadena que indica el nombre de la cookie.
    - valor: Obligatorio. Cadena (u otro valor convertible en cadena) que indicará qué valor deberá tener la cookie.
    - caducidad: Opcional. Entero que indica en segundos cuando deberá de caducar la cookie. Ej. time()+3600*24*30 hará que la cookie caduque pasados 30 días. Si no se indica el parámetro, la cookie caducará al cerrar el navegador.
  - **Siempre se han de crear antes de que se le envíe cualquier salida al navegador.**
- Leer una cookie:
  - Al recibir una petición de una página PHP carga automáticamente el valor de todas las cookies del sitio web en la variable superglobal $_COOKIE.
  - $_COOKIE es un vector asociativo. Para acceder al valor de una cookie se usa como clave su nombre. Ej. $_COOKIE['user'].
  - Para comprobar si una cookie ha sido establecida se emplea isset(). Ej. isset($_COOKIE["user"]).
- Borrar una cookie: La volvemos a crear con un periodo de caducidad que ya haya pasado. P. ej. time() – 3600.

Ejemplo: Crear una cookie

```
<?php
   // Este bloque de PHP está antes de que se
   // envíe cualquier salida al navegador
   setcookie("user", "Alex Porter", time()+3600);
?>
<html> <!-- Se está enviando una salida -->
   <body>
         …
   </body>
</html>
```

Ejemplo: Leer una cookie

```
<?php
   // Imprimir la cookie ‘user’
   echo $_COOKIE[‘user’];
   // Imprimir todas las cookies recibidas
   print_r($_COOKIE);
   ?>
```

Ejemplo: Comprobar si está establecida una cookie

```
<html>
   <body>
   <?php
        if (isset($_COOKIE[‘user’]))
                echo “Hola " . $_COOKIE["user"] . "!<br />";
        else
                echo “¿Te conozco?<br />";
   ?>
   </body>
</html>
```

Ejemplo: Borrar una cookie

```
<?php
        // Caducó hace una hora
        setcookie("user", "", time()-3600);
?>
```

Cookies como vectores

- Establecimiento:

```
<?php
setcookie ("name[first]", "Dennis", time() + (60*60*24));
setcookie ("name[last]", "Pallett", time() + (60*60*24));
?>
```

- Lectura:

```
<?php
        echo "First Name: " . $_COOKIE['name']['first'];
        echo "<br />Last Name: " . $_COOKIE['name']['last'];
?>
```

Experimento: Comprobar si el navegador soporta cookies

EJ48

```
<?php
    // Comprobamos si se ha pasado antes por la página
    // (Se estableción la cookie)
    if ($_GET['establecida'] != 'si') {
            // Establecemos la cookie
            setcookie ('prueba', 'prueba', time() + 60);
            // Obligamos al navegador a recargar la página
            header ('Location: ej48.php?establecida=si');
            }
    else {
            // Comprobamos si la cookie está establecida
            if (isset($_COOKIE['prueba']) && !empty($_COOKIE['prueba'])){
                    setcookie ('prueba', 'prueba', time() - 3600);
                    echo 'Su navegador soporta cookies';
                    }
            else
                    echo 'Su navegador <b>NO</b> soporta cookies '.$_COOKIE['prueba'];

            }
?>
```

3.6.2 Sesiones

- ¿Qué es una sesión?
- Manejo de sesiones en PHP
- Experimento: Sesiones sin cookies

Sesiones

- Pequeños trozos de información **almacenados temporalmente en el servidor**.
  - Esta información está asociada a una sesión de navegación (usuario).
  - Se elimina cuando el usuario no accede al sitio durante un tiempo determinado.
- ¿Para qué?
  - Recordar información temporal: Ej. Recordar que el usuario ya está autentificado.
  - Recomendada para información sensible:
    - **NO** se almacena en el ordenador del usuario por lo que no puede ser alterada por éste.
- ¿Por qué?
  - Misma razón que las cookies. En este caso la solución es más elaborada, pero se sigue basando en las cookies o en paso de parámetros entre páginas.

Manejo de sesiones en PHP

- Comenzar una sesión:
  - session_start()
  - Siempre antes de mandar cualquier salida al usuario.
  - Crea (si no existe) una sesión para el usuario en el servidor y le asigna un identificador único.
- Establecer/Leer una variable de sesión:
  - Acceso por la variable superglobal $_SESSION.
  - Vector asociativo: Ejs.
    - $_SESSION['views']=1;
    - echo "Pageviews=". $_SESSION['views'];
  - Se puede comprobar la existencia con isset()
- Eliminar una vairable de sesión:
  - Usar unset(). Ej. unset($_SESSION['views']);
- Destruir la sesión:
  - session_destroy();

Ejemplo: Contador de páginas vistas por un usuario

EJ49

```
<?php
  session_start();
  if(isset($_SESSION['views']))
     $_SESSION['views']++;
  else
     $_SESSION['views']=1;
  echo "Views=". $_SESSION['views'];
  ?>
```

Experimento: Sesiones sin cookies

- Pruebe a deshabilitar las cookies
  - ¿Puede el servidor mantener los datos del usuario?
  - ¿Por qué?
  - ¿Hay otras alternativas?
    - http://www.desarrolloweb.com/articulos/propagar-id-sesion-php.html
    - Ej. `<a href="leer_sesion.php?<?=SID?>">Paso variable de sesión por URL</a>`
- Pruebe a abrir otro navegador (de otro usuario u otro fabircante) ¿Se mantiene la información de sesión de uno a otro?

Ejercicio 6

- Partiendo del código del ejercicio 4 (muro de facebook) modifíquelo que:
  - Use cookies para recordar (durante 24) el último nombre de usuario, de forma que este campo aparezca pre-relleno en su caso.
  - Use una variable de sesión para recordar la contraseña que escribió el usuario y exija que esta sea la misma que se ha introducido después de la primera vez que se mandan los datos.

Ejercicio 7

- (0.5 ptos.) Lea el material suministrado y plantee en el foro abierto a tal efecto una cuestión sobre el funcionamiento o implementación del sistema descrito.
  - Obviamente no se aceptarán preguntas con escaso interés o utilidad (poca o mucha dificultad) para “salir del paso”.
  - No se aceptarán repeticiones o variaciones de una misma pregunta.
- (0.5 ptos.) : En la clase siguiente tendrá que responder a las preguntas de los compañeros (una por persona; al azar). Una persona puede responder más de una (Puntos extra).