PWD-Programación Web Dinámica

PHP – MySQL
PDO

Universidad Nacional del Comahue

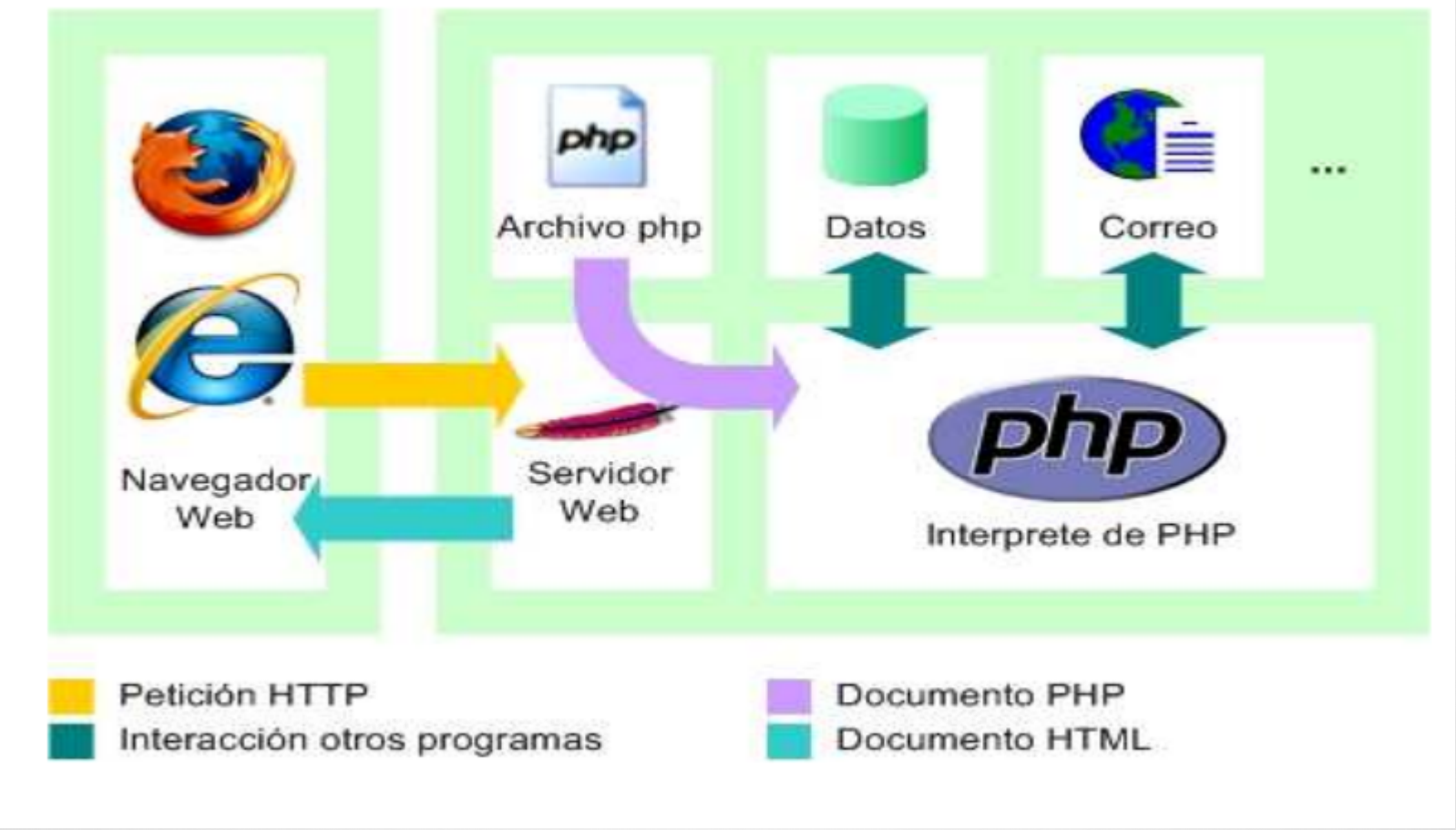
Facultad de Informática

Departamento de Programación

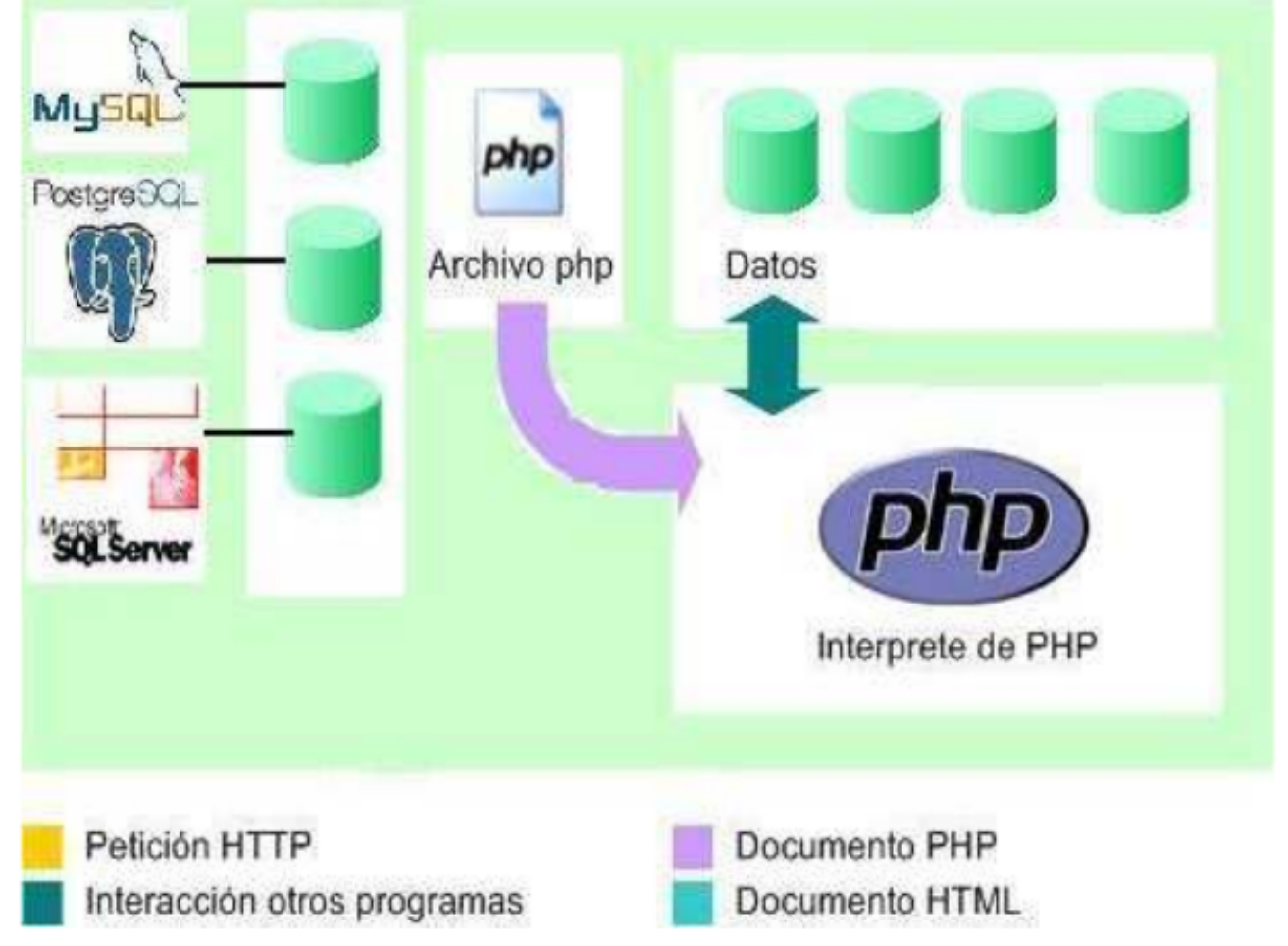
PHP - MYSQL

- Para comenzar analicemos unas preguntas:
  - ¿Necesito utilizar una Base de Datos?
    - ¿En qué casos la necesito?
      - ¿En que casos no?

App Web : PHP–HTML-Mysql



App Web : PHP–HTML-Mysql



PDO - Objetos de Datos de PHP

- Es una extensión que permite al desarrollador abstraerse de la base de datos de una aplicación.
- Hacer el código portable a otras plataformas y motores de bases de datos
- Disponible a partir de la versión 5.1
- PDO tiene implementaciones para muchos motores de bases de datos, entre ellos: MySQL -PostgreSQL- Firebird- SQLite- DB2- Oracle- SQL Server

PDO - Objetos de Datos de PHP

- Es una extensión que permite al desarrollador abstraerse de la base de datos de una aplicación.
- Hacer el código portable a otras plataformas y motores de bases de datos
- Disponible a partir de la versión 5.1
- PDO tiene implementaciones para muchos motores de bases de datos, entre ellos: MySQL -PostgreSQL- Firebird- SQLite- DB2- Oracle- SQL Server

PHP–HTML: Subir Archivos

- La conexión a una base de datos se realiza creando una instancia de la clase base PDO.
- El constructor acepta parámetros para especificar la fuente de datos y opcionalmente el nombre de usuario y el password.

PDO - Objetos de Datos de PHP

- **Pasos para trabajar con una base de datos:**
  - **Conectar con el servidor de bases de datos**
  - **Seleccionar una base de datos**
  - **Enviar la instrucción SQL a la base de datos**
  - **Obtener y procesar los resultados**
  - **Cerrar la conexión con el servidor de bases de datos**

PDO - Conexión

- La conexión a una base de datos se realiza creando una instancia de la clase base PDO

```
<?php
//Nos conectamos al motor indicando drivers, servidor, base de datos, usuario y password
$dbh = new PDO('mysql:host=localhost;dbname=test',
$user, $pass);
?>
```

- La conexión continuará activa mientras siga activo el objeto PDO

PDO - Consultas

- Para realizar consultas a la base de datos, tenemos los siguientes métodos:
- query(): Ejecuta y devuelve el resultado de la consulta.
- Si es una consulta que no devuelve registros, el método devuelve TRUE o FALSE
- Si la consulta devuelve registros, el método devuelve los registros correspondientes o FALSE.
- En caso de tratarse de un select nos devuelve un objeto de la clase PDOStatement

PDO - Consultas

- **Funciones de la Clase PDO:**

PDOStatement::fetch permite obtener la siguiente fila de un conjunto de resultados de una consulta Esta instrucción tiene varios estilos de recuperación, entre ellos:

  - PDO::FETCH_NUM: Retorna la siguiente fila como un arreglo indexado por posición.

  - PDO::FETCH_ASSOC: Retorna la siguiente fila como un arreglo indexado por el nombre de la columna.

- Si se produce un error, la instrucción fetch retornará FALSE.

PDO - Consultas

- ***PDOStatement::fetchAll*** retornará un arreglo conteniendo todas las filas de un conjunto de resultados.
- ***PDOStatement::columnCount*** Devuelve el número de columnas de un conjunto de resultados.
- ***PDOStatement::rowCount*** devuelve el número de filas afectadas por la última sentencia SQL

PDO - Uso

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test',$user,
$pass);
$consulta = "INSERT INTO $dbTabla (nombre, apellidos) VALUES
('.$nombre.', '.$apellidos.')";
if ($dbh->query($consulta)){
        echo "<p>Registro creado correctamente.</p>";
}
else { echo "<p>Error al crear el registro.<p>"; }
$dbh = NULL;
?>
```

PDO - Uso

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test',$user,
$pass);
$consulta = "SELECT * FROM $dbTabla";
$result = $dbh->query($consulta);
if (!$result) {
        echo "<p>Error en la consulta.</p>";
}else {
        $datos = $result->fetchAll(PDO::FETCH_ASSOC)
         foreach ($datos as $valor) {
                echo "<p>$valor[“nombre”] $valor[“apellidos”]</p>";
         }
}
?>
```

PDO - Uso

- ***exec():*** Ejecuta la consulta pasada como parámetro y retorna el número de registros afectados.

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test',
$user, $pass);
$sql = "UPDATE releases WHERE id > 2";
$affectedRows = $dbh->exec($sql);
echo "Registros modificados: $affectedRows";
?>
```

PDO - Uso

- ***errorInfo():*** devuelve un array con la información del error sobre la última operación realizada por el manejador de la base de datos.
- El array contiene los siguientes campos:
  - Pos 0: Código de error
  - Pos 1: Código de error específico del driver.
  - Pos 2: Mensaje del error específico del driver.

PDO - Transacciones

- ***$dbh->beginTransaction()***: iniciar una transacción.
- ***$dbh->commit()***: aceptar la transacción.
- ***$dbh->rollback()***: deshacer la transacción.

Extendiendo PDO

- Vamos a implementar la clase BaseDatos a partir de PDO:
  - La nueva clase hereda de PDO.
  - Esta clase nos permitirá agregar toda la funcionalidad que consideremos necesaria.
  - Atributos:

```
$engine; /* Motor de base de datos*/
$host; /* Servidor de base de datos*/
$database; /* Nombre de la base de datos*/
$user; /* Usuario con el nos conectaremos*/
$pass; /* clave del usuario */
$debug; /* Valor booleano que indicara si queremos que nos
muestre las consultas o no * /
```

Extendiendo PDO

- Método Constructor:

```
public function __construct(){
        $this->engine = 'mysql';
        $this->host = 'tecnicatura-server';
        $this->database = ‘Prueba';
        $this->user = 'root';
        $this->pass = ‘123456';
        $this->debug = true;
        $dns = $this->engine.':dbname='.$this->
        database.";
        host=".$this->host;
        parent::__construct( $dns, $this->user, $this-
        >pass )
}
```

- Ver implementación clase BaseDatos.php