UF3: Tècniques d'accés a dades

Desenvolupament web en entorn servidor

3.4

EXTENSIÓ PDO

PHP DATA OBJECTS

JESUÏTES El Clot
Escola del Clot
Sergi Grau

Introducció

Sergi Grau
sergi.grau@fje.edu

L'EXTENSIÓ OBJECTES DE DADES DE PHP (PDO PER LES SEVES SIGLES EN ANGLÈS) DEFINEIX UNA INTERFÍCIE LLEUGERA PER PODER ACCEDIR A BASES DE DADES EN PHP. CADA CONTROLADOR DE BASES DE DADES QUE IMPLEMENTI LA INTERFÍCIE PDO POT EXPOSAR CARACTERÍSTIQUES ESPECÍFIQUES DE LA BASE DE DADES, COM LES FUNCIONS HABITUALS DE L'EXTENSIÓ. OBSERVEU QUE NO ES POT REALITZAR CAP DE LES FUNCIONS DE LA BASES DE DADES UTILITZANT L'EXTENSIÓ PDO PER SI MATEIXA; S'HA D'UTILITZAR UN CONTROLADOR DE PDO ESPECÍFIC DE LA BASE DE DADES PER TENIR ACCÉS A UN SERVIDOR DE BASE DE DADES.

Característiques

Sergi Grau
sergi.grau@fje.edu

PDO proporciona una capa d'abstracció d'accés a dades, el que significa que, independentment de la base de dades que s'estigui utilitzant, s'usen les mateixes funcions per fer consultes i obtenir dades. PDO no proporciona una abstracció de bases de dades, no reescriu SQL ni emula característiques absents. S'hauria utilitzar una capa d'abstracció totalment desenvolupada si fos necessària aquesta capacitat.

PDO ve amb PHP 5.1, i està disponible com una extensió PECL per PHP 5.0; PDO requereix les característiques noves de OO del nucli de PHP 5, de manera que no s'executarà amb versions anteriors de PHP.

Instal·lació en sistemes UNIX

Sergi Grau
sergi.grau@fje.edu

PDO i el controlador PDO_SQLITE estan activats per defecte a partir de PHP 5.1.0. Podria ser necessari activar el controlador PDO per a la base de dades de la seva elecció; consulteu la documentació sobre els Controladors de PDO per a bases de dades específiques per obtenir més informació.

Quan s'instal·la PDO com un mòdul compartit, serà necessari actualitzar el fitxer php.ini, així, l'extensió PDO es carregarà automàticament quan s'executi PHP. També serà necessari activar qualsevol controlador de bases de dades específic en aquest fitxer, assegureu-vos que estiguin declarats després de la línia pdo.so, doncs PDO ha inicilizarse abans de carregar les extensions específiques per a bases de dades. Si es construeixen estàticament PDO i les extensions específiques per a bases de dades, es pot ometre aquest pas.

```
extensió = pdo.so
```

Instal·lació en sistemes Windows

Sergi Grau
sergi.grau@fje.edu

PDO i la majoria de controladors vénen en PHP com extensions compartides i, per activar, simplement s'ha d'editar el fitxer php.ini. Aquest pas no és necessari per a PHP 5.3 o superior, ja que ja no es requereix la DLL per utilitzar PDO.

extensió = php_pdo.dll

Després, escolliu els altres fitxers DLL de bases de dades específiques, i utilitzeu dl () per carregar en temps d'execució, o activar-los a php.ini per sota de la línia php_pdo.dll. Per exemple:

```
extensió = php_pdo.dll
extensió = php_pdo_firebird.dll
extensió = php_pdo_informix.dll
extensió = php_pdo_mssql.dll
extensió = php_pdo_mysql.dll
```

Constants de classe

Sergi Grau
sergi.grau@fje.edu

Aquestes constants estan definides per aquesta extensió i estaran disponibles només quan l'extensió hagi estat compilada amb PHP, o bé sigui carregada dinàmicament en execució.

```
PDO::PARAM_BOOL ( integer )
Representa un tipus de dada booleà.
PDO::PARAM_NULL ( integer )
Representa el tipus de dada NULL de SQL.
PDO::PARAM_INT ( integer )
Representa el tipus de dada INTEGER de SQL.
PDO::PARAM_STR ( integer )
Representa el tipus de dada CHAR, VARCHAR de SQL, o un altre tipus de dades de string.
PDO::PARAM_LOB ( integer )
Representa el tipus de dada LOB (objecte gran) de SQL.
PDO::PARAM_STMT ( integer )
Representa un tipus de conjunt de registres. No està admès actualment per cap controlador.
PDO::PARAM_INPUT_OUTPUT ( integer )
Especifica que el paràmetre és d'entrada / sortida (INOUT) per un procediment emmagatzemat. S'ha de
fer un OR a nivell de bits amb un tipus de dades PDO :: PARAM_ * explícit.
PDO::FETCH_LAZY ( integer )
Especifica que el mètode d'obtenció de tornar cada fila com un objecte amb els noms de les
variables que es corresponen als noms de les columnes retornats en el conjunt de resultats.
PDO::FETCH_LAZY crea els noms de les variables de l'objecte a mesura que es accedeixen. No és
vàlida dins de PDOStatement :: fetchAll () .
PDO::FETCH_ASSOC ( integer )
Especifica que el mètode d'obtenció de tornar cada fila com una matriu indexat pels noms de les
columnes retornats en el corresponent conjunt de resultats. Si aquest conté múltiples columnes amb
el mateix nom, PDO::FETCH_ASSOC retorna un únic valor per nom de columna.
```

. . .

Connexions i la seva administració

Sergi Grau
sergi.grau@fje.edu

Les connexions s'estableixen creant instàncies de la classe base PDO. No importa el controlador que s'utilitzi, sempre s'utilitzarà el nom de la classe PDO. El constructor accepta paràmetres per especificar l'origen de dades (conegut com DSN) i, opcionalment, el nom d'usuari i la contrasenya (si n'hi ha).

Si haguessin errors de connexió, es llançarà un objecte PDOException. Es pot capturar l'excepció si cal manejar la condició de l'error, o es podria optar per deixar-la en mans d'un gestor d'excepcions global d'una aplicació configurat mitjançant set_exception_handler ()

```
<?php
try {
$gbd = new PDO ( 'mysql:host=localhost;dbname=test' , $usuari ,
$contrasenya );
foreach( $gbd -> query ( 'SELECT * from FOO' ) as $fila ) {
print_r ( $fila );
}
$gbd = null ;
} catch ( PDOException $e ) {
print "¡Error!: " . $e -> getMessage () . "<br/>" ;
die();
}
?>
```

Connexions i la seva administració

Sergi Grau
sergi.grau@fje.edu

Un cop realitzada amb èxit una connexió a la base de dades, serà retornada una instància de la classe PDO l'script. La connexió romandrà activa durant el temps de vida de l'objecte PDO. Per tancar la connexió, cal destruir l'objecte assegurant-se que totes les referències a ell existents siguin eliminades (això es pot fer assignant NULL a la variable que conté l'objecte). Si no es realitza explícitament, PHP tancarà automàticament la connexió quan l'script finalitzi.

```
<?php
$gbd = new PDO ( 'mysql:host=localhost;dbname=test' , $usuari ,
$contrasenya );
...


// terminar i tancar
$gbd = null ;
?>
```

Connexions i la seva administració

Sergi Grau
sergi.grau@fje.edu

Moltes aplicacions web es beneficiaran de l'ús de connexions persistents a servidors de bases de dades. Les connexions persistents no són tancades al final del script, sinó que són en memòria cau i reutilitzades quan un altre script sol · liciti una connexió que usi les mateixes credencials. La memòria cau de connexions persistents permet evitar la càrrega addicional d'establir una nova connexió cada vegada que un script necessiti comunicar-se amb la base de dades, donant com a resultat una aplicació web més ràpida.

```
<?php
$gbd = new PDO ( 'mysql:host=localhost;dbname=test' , $usuari ,
$contrasenya , array( PDO :: ATTR_PERSISTENT => true ));
?>
```

Exemple de transaccions

Sergi Grau
sergi.grau@fje.edu

```
<?php
try {
$gbd = new PDO ( 'odbc:SAMPLE' , 'db2inst1' , 'ibmdb2' ,
array( PDO :: ATTR_PERSISTENT => true ));
echo "Conectado\n" ;
} catch ( Exception $e ) {
die( "problema de connexio: " . $e -> getMessage ());
}

try {
$gbd -> setAttribute ( PDO :: ATTR_ERRMODE , PDO :: ERRMODE_EXCEPTION );

$gbd -> beginTransaction ();
$gbd -> exec ( "insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')" );
$gbd -> exec ( "insert into salarychange (id, amount, changedate)
values (23, 50000, NOW())" );
$gbd -> commit ();

} catch ( Exception $e ) {
$gbd -> rollBack ();
echo "Error: " . $e -> getMessage ();
}
?>
```

Sentències preparades i procediments emmagatzemats

Sergi Grau
sergi.grau@fje.edu

Moltes de les bases de dades més madures admeten el concepte de sentències preparades. Aquestes poden definir-se com un tipus de plantilles compilades per SQL que les aplicacions volen executar, que poden ser personalitzades usant paràmetres de variables. Les sentències preparades ofereixen dos grans beneficis:

La consulta només necessita ser analitzada (o preparada) un cop, però pot ser executada múltiples vegades amb els mateixos o diferents paràmetres. Quan la consulta és preparada, la base de dades analitzarà, compilarà i optimitzarà el seu pla per executar-la. Per a consultes complexes, aquest procés pot ser suficient temps com perquè alenteixi notablement una aplicació si cal repetir la mateixa consulta moltes vegades amb els mateixos paràmetres. Usant una sentència preparada, l'aplicació evita repetir el cicle d'anàlisi / compilació / optimització. Això significa que les sentències preparades usen menys recursos i s'executen més ràpidament.

Sentències preparades i procediments emmagatzemats

Sergi Grau
sergi.grau@fje.edu

Els paràmetres per les sentències preparades no necessiten estar entre cometes, el controlador automàticament s'encarrega d'això. Si una aplicació utilitza exclusivament sentències preparades, el desenvolupador pot estar segur que no passaran injeccions SQL (però, si altres parts de la consulta es construeixen amb dades d'entrada sense escapar, les injeccions SQL possibles).

Les sentències preparades són tan útils que són l'única característica que PDO emularà per als controladors que no les admetin. Això assegura que una aplicació sigui capaç d'usar el mateix paradigma d'accés a dades independentment de les capacitats de la base de dades.

Exemple

Sergi Grau
sergi.grau@fje.edu

```
<?php
$sentencia = $gbd -> prepare ( "INSERT INTO REGISTRY (name, value) VALUES (:name,
:value)" );
$sentencia -> bindParam ( ':name' , $nombre );
$sentencia -> bindParam ( ':value' , $valor );

$sentencia = $gbd -> prepare ( "INSERT INTO REGISTRY (name, value) VALUES (?, ?)" );
$sentencia -> bindParam ( 1 , $nombre );
$sentencia -> bindParam ( 2 , $valor );


$nombre = 'sergi' ;
$valor = 1 ;
$sentencia -> execute ();


$nombre = 'joan' ;
$valor = 2 ;
$stmt -> execute ();
?>
```

Exemple

Sergi Grau
sergi.grau@fje.edu

```php
<?php
$sentencia = $gbd -> prepare ( "SELECT * FROM REGISTRY where name = ?" );
if ( $sentencia -> execute (array( $_GET [ 'name' ]))) {
while ( $fila = $sentencia -> fetch ()) {
print_r ( $fila );
}
}
?>

<?php
$sentencia = $gbd -> prepare ( "CALL sp_returns_string(?)" );
$sentencia -> bindParam ( 1 , $valor_retorn , PDO :: PARAM_STR , 4000 );

$sentencia -> execute ();

print "el procediment retorna $valor_retorn \n" ;

?>
```

La classe PDO

Sergi Grau
sergi.grau@fje.edu

Representa una connexió entre PHP i un servidor de base de dades.

```
PDO {
__construct (string $dsn [, string $username [, string $password [, array
$driver_options ]]])
bool beginTransaction (void)
bool commit (void)
mixed errorCode (void)
array errorInfo (void)
int exec (string $statement )
mixed getAttribute (int $attribute )
static array getAvailableDrivers (void)
bool inTransaction (void)
string lastInsertId ([string $name = NULL ])
PDOStatement prepari (string $statement [, array $driver_options = array ()])
PDOStatement query (string $statement )
string quote (string $string [, int $parameter_type = PDO :: PARAM_STR])
bool Rollback (void)
bool setAttribute (int $attribute , mixed $value )
}
```

La classe PDO

Sergi Grau
sergi.grau@fje.edu

Representa una connexió entre PHP i un servidor de base de dades.

PDO :: beginTransaction - Comença una transacció
PDO :: commit - Consigna una transacció
PDO :: __construct - Crea una instància de PDO que representa una connexió a una base de dades
PDO :: errorCode - Obté un SQLSTATE associat amb l'última operació en el gestor de la base de dades
PDO :: errorInfo - Obté informació estesa de l'error associat amb l'última operació del controlador de la base de dades
PDO :: exec - Executa una sentència SQL i retorna el nombre de files afectades
PDO :: getAttribute - Retorna un atribut de la connexió a la base de dades
PDO :: getAvailableDrivers - Retorna un array amb els controladors de PDO disponibles
PDO :: inTransaction - Comprova si una transacció està activa
PDO :: lastInsertId - Retorna l'ID de l'última fila o seqüència inserida
PDO :: prepari - Prepara una sentència perquè s'execute i retorna un objecte sentència
PDO :: query - Executa una sentència SQL, retornant un conjunt de resultats com un objecte PDOStatement
PDO :: quote - entre cometes una cadena de caràcters per usar-la en una consulta
PDO :: Rollback - Reverteix una transacció
PDO :: setAttribute - Estableix un atribut

Exemple

Sergi Grau
sergi.grau@fje.edu

```
<?php
$dbh = new PDO ( 'odbc:sample' , 'db2inst1' , 'ibmdb2' );

$count = $dbh -> exec ( "DELETE FROM fruita WHERE color = 'vermell'" );

print( "files esborrades $count rows.\n" );
?>

<?php
function getFruita ( $conn ) {
$sql = 'SELECT nom, color, calories FROM fruita ORDER BY name' ;
foreach ( $conn -> query ( $sql ) as $fila ) {
print $fila [ 'nom' ] . "\t" ;

print $fila [ 'color' ] . "\t" ;

print $fila [ 'calories' ] . "\n" ;

}
}
?>
```

La classe PDOStatement

Sergi Grau
sergi.grau@fje.edu

Representa una sentència preparada i, després de l'execució de la instrucció, un conjunts de resultats associat.

```
PDOStatement implements Traversable {
/ * Propietats * /
readonly string $ queryString ;
/ * Mètodes * /
bool bindColumn ( mixed $column , mixed &$param [, int $type [, int $maxlen [, mixed
$driverdata ]]])
bool bindParam ( mixed $parameter , mixed &$variable [, int $data_type = PDO :: PARAM_STR
[, int $length [, mixed $driver_options ]]])
bool bindValue ( mixed $parameter , mixed $value [, int $data_type = PDO :: PARAM_STR])
bool closeCursor (void)
int columnCount (void)
bool debugDumpParams (void)
string errorCode (void)
array errorInfo (void)
bool execute ([array $input_parameters ])
mixed fetch ([int $fetch_style [, int $cursor_orientation = PDO :: FETCH_ORI_NEXT [, int
$cursor_offset = 0]]])
array fetchAll ([int $fetch_style [, mixed $fetch_argument [, array $ctor_args = array
()]]])
string fetchColumn ([int $column_number = 0])
mixed fetchObject ([string $class_name = "stdClass" [, array $ctor_args ]])
mixed getAttribute (int $attribute )
array getColumnMeta (int $column )
bool nextRowset (void)
int rowCount (void)
bool setAttribute (int $attribute , mixed $value )
bool setFetchMode (int $mode )
```

La classe PDOStatement

Sergi Grau
sergi.grau@fje.edu

PDOStatement :: bindColumn - Vincula una columna a una variable de PHP
PDOStatement :: bindParam - Vincula un paràmetre al nom de variable especificat
PDOStatement :: bindValue - Vincula un valor a un paràmetre
PDOStatement :: closeCursor - Tanca un cursor, habilitant a la sentència perquè sigui executada de nou
PDOStatement :: columnCount - Retorna el nombre de columnes d'un conjunt de resultats
PDOStatement :: debugDumpParams - Bolca una ordre preparat de SQL
PDOStatement :: errorCode - Obté el SQLSTATE associat amb l'última operació del gestor de sentència
PDOStatement :: errorInfo - Obté informació ampliada de l'error associat amb l'última operació del gestor de sentència
PDOStatement :: execute - Executa una sentència preparada
PDOStatement :: fetch - Obté la fila d'un conjunt de resultats
PDOStatement :: fetchAll - Retorna una matriu que conté totes les files del conjunt de resultats
PDOStatement :: fetchColumn - Retorna una única columna de la fila d'un conjunt de resultats
PDOStatement :: fetchObject - Obté la següent fila i la retorna com un objecte
PDOStatement :: getAttribute - Recupera un atribut de sentència
PDOStatement :: getColumnMeta - Retorna metadades d'una columna d'un conjunt de resultats
PDOStatement :: nextRowset - Avança fins al següent conjunt de files d'un gestor de sentència multiconjunt de files
PDOStatement :: rowCount - Retorna el nombre de files afectades per l'última sentència SQL
PDOStatement :: setAttribute - Estableix un atribut de sentència
PDOStatement :: setFetchMode - Estableix la manera d'obtenció per aquesta sentència

Exemple

Sergi Grau
sergi.grau@fje.edu

```php
<?php
$gsent = $gbd -> prepare ( "SELECT nom, color FROM fruita" );
$gsent -> execute ();

$resultat = $gsent -> fetchAll ();
print_r ( $resultat );



$gsent = $gbd -> prepare ( "SELECT nom, color FROM fruita" );
$gsent -> execute ();


print( "PDO::FETCH_ASSOC: " );
print( "Tornar la següent fila com una matriu indexat per nom de colunmna\n" );
$result = $gsent -> fetch ( PDO :: FETCH_ASSOC );
print_r ( $result );
print( "\n" );
```

Exemple

Sergi Grau
sergi.grau@fje.edu

```
print( "PDO::FETCH_BOTH: " );
print( "Tornar la següent fila com una matriu indexat per nom i número de
columna\n" );
$result = $gsent -> fetch ( PDO :: FETCH_BOTH );
print_r ( $result );
print( "\n" );

print( "PDO::FETCH_LAZY: " );
print( "Tornar la següent fila com un objecte anònim amb noms de columna com
propietats\n" );
$result = $gsent -> fetch ( PDO :: FETCH_LAZY );
print_r ( $result );
print( "\n" );

print( "PDO::FETCH_OBJ: " );
print( "Tornar la següent fila com un objecte anònim amb noms de columna com
propietats\n" );
$result = $gsent -> fetch ( PDO :: FETCH_OBJ );
print $result -> NAME ;
print( "\n" );
?>
```

Controladors de PDO

Sergi Grau
sergi.grau@fje.edu

Nom del controlador	Bases de dades admeses
PDO_CUBRID	Cobriu
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird / Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure
PDO_4D	4D

Funcions de MySQL (PDO_MYSQL)

Sergi Grau
sergi.grau@fje.edu

PDO_MYSQL és un controlador que implementa la interfície d'Objectes de Dades de PHP (PDO) per permetre l'accés de PHP a bases de dades de MySQL 3.x, 4.xi 5.x.

PDO_MYSQL aprofitarà el suport natiu de sentències preparades present en MySQL 4.1 i superior. Si s'utilitza una versió anterior de les biblioteques client de MySQL, PDO les emularà.

Utilitzeu - with-PDO-mysql [= DIR] per instal · lar l'extensió PDO MySQL, on l'opció [= DIR] és el directori base de la instal · lació de MySQL. Si a [= DIR] se li passa mysqlnd , llavors serà utilitzat el controlador natiu de MySQL.

Opcionalment, - with-mysql-sock [= DIR] estableix la ubicació del punter del socket de unix de MySQL per a totes les extensions de MySQL, incloent PDO_MYSQL. Si no s'especifica, es buscarà en les ubicacions predeterminades.

Opcionalment, - with-zlib-dir [= DIR] s'utilitza per establir la ruta per al prefix d'instal · lació de libz.

```
$. / Configure - with-PDO-mysql - with-mysql-sock = / var / mysql / mysql.sock
```

Funcions de MySQL (PDO_MYSQL)

Sergi Grau
sergi.grau@fje.edu

PDO_MYSQL DSN - Connectar-se a bases de dades de MySQL

```
<?php
$dsn = 'mysql:host=localhost;dbname=testdb' ;
$nombre_usuari = 'nom_usuari' ;
$contrasenya = 'contrasenya' ;
$opcions = array(
PDO :: MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8' ,
);

$gbd = new PDO ( $dsn , $nom_usuari , $contrasenya , $opciones );
?>
```