Progetto CSP: Innovation & Creativity for School

Istituto Tecnico Industriale Statale
"Giulio Cesare Faccio" – Vercelli
Gruppo 5°ELETTRONICI

Alessandro Alessio, Marco Chiavieri, Kabir Ferro
Professore Massimo Brusa.

QR PARKING

PIATTAFORMA PHP

Struttura e funzionamento delle classi e delle funzioni.

myPDO

myPDO è un estensione della libreria PDO.

PDO (PHP Data Object) è una classe implementata dalla versione 5 di PHP la quale permette di interagire con un database, indifferentemente dal tipo, con un'unica interfaccia.
PDO permette quindi una buona portabilità della piattaforma.
Come svantaggio si hanno però delle prestazioni ridotte rispetto l'utilizzo delle librerie mysql o mysqli.

myPDO permette una connessione automatica al database senza dover inserire le credenziali di accesso ad ogni inizializzazione dell'oggetto.
Semplifica alcune procedure rispetto all'utilizzo standard della classe PDO, e prende traccia del numero di query utilizzate.

Struttura

```
class myPDO extends PDO {

        private $dsn
        private $username
        private $password
        public $px

        public function __construct()

        protected static $queries_count
        public function get_queries_count()
        public function queryinc()

        public function query($query)
        public function element($query)
        public function assoc($query)
        public function result($query)

        public $users
        public $user_profiles
        public $user_status
        public $user_permits

}
```

Metodi della libreria myPDO:

Costruttore

Descrizione:

myPDO:: __construct ()

Instaura la connessione inviando i dati alla classe PDO

Parametri:

private ***myPDO->dsn***

Tipologia del database, Host e Nome del database

private ***myPDO-> username***

Username per la connessione al database

private ***myPDO->password***

Password per la connessione al database

Assoc

Descrizione:

Array **myPDO::assoc**(stringa *$query*)

Permette di recuperare un array associativa dal database richiesta tramite ***query***

Parametri:

query

Richiesta da effettuare al database

Valore di ritorno:

Array associativa richiesta al database *($result[0]*)

Esempio:

Tabella '**users'**:

id	**user**
1	Giovanni
2	Silvio
3	Riccardo

```
$result = myPDO::assoc(" SELECT * FROM users WHERE id = '1' ");

// Restituisce
$result['id'] =  '1';
```

```
$result['user'] = 'Giovanni';
```

result

Descrizione:

Oggetto **myPDO::result**(stringa *$query*)

Permette di recuperare un oggetto PDOStatement creato da **PDO::prepare** ed eseguito da **PDO::execute** richiesto tramite ***query***

Parametri:

query

Richiesta da effettuare al database

Valore di ritorno:

Oggetto PDOStatement

Esempi:

Tabella '**users'**:

id	**user**
1	Giovanni
2	Silvio
3	Riccardo

```
$result = myPDO::result(" INSERT INTO user (id, user) VALUES('4','Luigi')  ");
```

id	**user**
1	Giovanni
2	Silvio
3	Riccardo
4	Luigi

Tabella '**users'**:

id	**user**
1	Giovanni

2	Silvio
3	Riccardo

```
$result = myPDO::result(" SELECT * FROM user LIMIT 0,2");

while($row = $result->fetch()):
        echo $row['user'] . '<br>';
endwhile;

// Restituisce
Giovanni
Silvio
```

get_queries_count

Descrizione:

Stringa **myPDO:: get_queries_count** ()

Permette di recuperare il numero di query eseguite

Esempio:

```
$result = myPDO::element(" SELECT user FROM users WHERE id = '1' ");

echo myPDO:: get_queries_count ()

// Visualizza '1'
```

Proprietà della libreria myPDO

public ***myPDO->px***

Prefisso delle tabelle del database

public ***myPDO->users***

Array associativa contenente il nome della tabella utente ed il nome delle sue colonne

```
array(
        'tablename'             => 'qp_users'
        'id'                    => 'id',
        'status'                => 'us',
        'permits'               => 'up',
        'activation_code'       => 'uac',
        'new_password'          => 'unp',
        'registration_date'     => 'urd',
        'email'                 => 'email',
        'username'              => 'username',
        'password'              => 'password',
);
```

public ***myPDO->user_profiles***

Array associativa contenente il nome della Profili utente ed il nome delle sue colonne

```
array(
        'tablename'     => 'qp_user_profiles',
        'id'            => 'id',
        'user_id'       => 'uid',
        'plate'         => 'plate'
);
```

public ***myPDO->user_status***

Array associativa contenente i valori numerici degli stati utente

```
array(
        'inactive'      => '0',
        'active'        => '1',
        'in_activation' => '2'
);
```

public ***myPDO->user_permits***

Array associativa contenente i valori numerici dei permessi utente

```
array(
        'administrator' => '0',
        'moderator'     => '1',
        'user'          => '2'
);
```

Le proprietà ***myPDO->users***, ***myPDO->user_profiles***, ***myPDO->user_status***, ***myPDO->user_permits*** ***s***ono utili per evitare la riprogrammazione della piattaforma in caso di eventuali modifiche al database.

qpUsers_basic

qpUsers_basic permette la gestione degli utenti della piattaforma.
qpUsers_basic è una classe astratta in cui vengono definite alcune proprietà e alcuni metodi da utilizzare nelle sue estensioni.

Struttura

```
abstract class qpUsers_basic  {

        protected static $pdo
        protected function setPDO()

        public function __construct()

        public $default_status
        public $default_permit
        public $default_login

        public function set_default_status($status)
        public function set_default_permit($permit)
        public function set_default_login($type)

        public function passwordEncryption($pwd)

}
```

Metodi della libreria qpUsers_basic:

Costruttore

Descrizione:

qpUsers_basic:: __construct ()

Inizializza un oggetto myPDO, permettendo alla classe l'interazione con la classe myPDO e di conseguenza con un database.

set_default_status

Descrizione:

Booleana **qpUsers_basic:: set_default_status** (stringa *$status*)

Permette di impostare il parametro qpUsers_basic-> default_status.

Parametri:

status

Dev'essere uno dei valori chiave dell'array ***myPDO->user_status***

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempio:

```
qpUsers_basic:: set_default_status ("active");
```

set_default_permit

Descrizione:

Booleana **qpUsers_basic:: set_default_permit** (stringa *$permit*)

Permette di impostare il parametro qpUsers_basic-> default_permit.

Parametri:

permit

Dev'essere uno dei valori chiave dell'array ***myPDO->user_permits***

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempio:

```
qpUsers_basic:: set_default_permit ("moderator");
```

set_default_login

Descrizione:

Booleana **qpUsers_basic:: set_default_login** (stringa *$type*)

Permette di impostare il parametro qpUsers_basic-> default_login.

Parametri:

type

Può assumere due valori: ‘username’ oppure ‘email’

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempio:

```
qpUsers_basic:: set_default_login(“email”);
```

passwordEncryption

Descrizione:

Stringa **qpUsers_basic:: passwordEncryption** (stringa *$password*)

Restituisce una stringa contenente una crittografia md5 di ***password***.

Parametri:

password

Stringa da sottoporre alla crittografia

Valore di ritorno:

Stringa

Esempio:

```
echo qpUsers_basic:: passwordEncryption (“password”);

// Visualizza bca0fe503ca30ced99bcee750648640e
```

qpUsers

qpUsers è un estensione della classe qpUsers_basic

Struttura

```
class qpUsers extends qpUsers_basic {

        public function prepare($type, $data)

        public function add($data)
        public function edit($data)
        public function get($id)

        public function activation ($user, $code)
        public function change_password ($id, $old_password, $new_password)
        public function new_password ($id)
        public function verify_new_password ($id, $code)
        public function set_new_password($id, $password)

        public function login($user, $password)
        public function logout()
        public function logged()
        public function log_info($type = 'id')
        public function log_refresh()

        public function ulist($s,$q)

        public function new_profile($id, $plate)

}
```

Metodi della libreria qpUsers:

prepare

Descrizione:

Booleana **qpUsers:: prepare** (stringa *$type,* array *$data*)

Permette di preparare **data** ritornando con un oggetto **qpUser_statement**.

In pratica prepara i dati utente (Verificando la correttezza dei dati) per utilizzarli con l'oggetto **qpUser_statement** potendo poi inserirli nel database o modificare dei dati già esistenti nel database.

Parametri:

type

Può assumere due valori: 'add oppure 'edit'

data

```
array(
        'id'                    => ‘ ’, // Obbligatorio con type ‘edit’
        'status'                => ‘ ’,
        'permit'                => ‘ ’,
        'activation_code'       => ‘ ’, //Utilizzabile solo con type ‘edit’
        'new_password'          => ‘ ’, //Utilizzabile solo con type ‘edit’
        'email'                 =>‘ ’, // Obbligatorio con type ‘add’
        'username'              => ‘ ’, // Obbligatorio con type ‘add’
        'password'              => ‘ ’, // Obbligatorio con type ‘add’
);
```

L’indice ‘status’ dev’essere uno dei valori chiave dell’array ***myPDO->user_status***, se assente prende il valore di **qpUsers->default_status**.

L’indice ‘permit’ dev’essere uno dei valori chiave dell’array ***myPDO->user_permits***, se assente prende il valore di **qpUsers->default_permit**.

Con **type** ‘edit’ dev’essere presente almeno uno degli indici 'status' , 'permit', 'activation_code' , 'new_password', 'email', 'username' , 'password'.

Valore di ritorno:

oggetto **qpUser_statement**

Approfondimento:

Se utilizzato con **type** ‘add’ prepara data per l’aggiunta di di un utente, mentre se utilizzato con **type** ‘edit’ prepara data per la modifica di eventuali dati di un utente.

‘activation_code’ viene generato automaticamente ed è un numero compreso tra **100000** e **999999**.
‘email’ viene controllato tramite **qpFormCheck->email()**
‘username’ viene controllato tramite **qpFormCheck->username()**
‘password’ viene controllato tramite **qpFormCheck->password()**

Esempi:

```
$data = array(
        'id'                    => ‘2’,
        'password'              => ‘ nuovapassword’,
);

$prepared = qpUsers_basic:: prepare (‘edit’, $data)
```

```
$data = array(
        'email'                 =>’name@hostname.com’,
        'username'              => ‘username ’,
        'password'              => ‘password’,
);
```

```
$prepared = qpUsers:: prepare (‘add, $data)
```

add

Descrizione:

Mix **qpUsers:: add** (stringa *$data*)

Utilizza **qpUsers::prepare()** e con l’oggetto di ritorno effettua **qpUser_statement::insert()**

In pratica effettua il ciclo completo per l’aggiunta di un utente al database

Parametri:

data

Vedi **qpUsers::prepare()**

Valore di ritorno:

In caso di successo un oggetto **qpUser_statement**, vedi **qpUser_statement::insert().**
In caso di insuccesso Booleana(False).

edit

Descrizione:

Mix **qpUsers:: edit** (stringa *$data*)

Utilizza **qpUsers::prepare()** e con l’oggetto di ritorno effettua **qpUser_statement::update()**

In pratica effettua il ciclo completo per la modifica di eventuali dati di un utente.

Parametri:

data

Vedi **qpUsers::prepare()**

Valore di ritorno:

In caso di successo un oggetto **qpUser_statement**, vedi **qpUser_statement::update().**
In caso di insuccesso Booleana(False).

get

Descrizione:

Mix **qpUsers:: get** (numero intero *$id*)

Recupera i dati di un utente.

Parametri:

id

Numero intero (Id utente)

Valore di ritorno:

oggetto **qpUser_statement**

Esempi:

id	user
1	Giovanni
2	Silvio
3	Riccardo

```
$getuser = qpUsers:: get ('2');

echo $getuser->username;
// Visualizza Giovanni
```

activation

Descrizione:

Booleana **qpUsers:: activation** (numero intero *$id,* numero intero *$code*)

Confronta il valore **code** con il codice di attivazione dell'utente avente codice identificativo **id**.
Se i codici combaciano esegue l'attivazione dell'utente.

Parametri:

id

Numero intero (Id utente)

code

Numero intero compreso tra 100000 e 999999 (Codice di attivazione dell'utente)

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempi:

id	user	Activation_code
1	Giovanni	101010
2	Silvio	202020
3	Riccardo	205060

```
$a = qpUsers:: activation ('1', '205080');
// $a = FALSE

$b = qpUsers:: activation ('2', '202020);
// $b = TRUE
```

change_password

Descrizione:

Booleana **qpUsers:: change_password** (numero intero *$id,* stringa *$old_password,* stringa *$new_password*)

Confronta il valore **old_password** con la passwrod dell'utente avente codice identificativo **id**.
Se le password combaciano verifica **new_password** tramite **qpFormCheck::password()** ed esegue la modifica della password inserendo **new_password** in caso di esito positivo.

Parametri:

id

Numero intero (Id utente)

old_password

Stringa (Password dell'utente)

new_password

Stringa (Nuova password dell'utente)

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempi:

id	user	password
1	Giovanni	bicicletta
2	Silvio	armadio
3	Riccardo	margherita

```
$a = qpUsers:: change_password ('1', 'bicicletta', 'nuovabicicletta');
// $a = TRUE

$b = qpUsers:: change_password ('1', 'armadio, 'nuovabicicletta');
// $b = FALSE
```

new_password

Descrizione:

Mix **qpUsers:: new_password** (numero intero *$id*)

Imposta un codice (casuale tra 10 per ottenere una nuova password all'utente con codice identificativo **id**

Parametri:

id

Numero intero (Id utente)

Valore di ritorno:

In caso di successo restituisce il codice per la creazione di una nuova password.
In caso di insuccesso valore Booleano(False)

Esempi:

id	user	password	new_password
1	Giovanni	bicicletta	
2	Silvio	armadio	
3	Riccardo	margherita	

```
$a = qpUsers:: new_password ('1');
// $a = 202020
```

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	
3	Riccardo	margherita	

verify_new_password

Descrizione:

Booleano **qpUsers:: verify_new_password** (numero intero *$id,* numero intero *$code*)

Verifica il codice inserito **code** con il codice salvato nel database per la creazione di una nuova password

Parametri:

id

Numero intero (Id utente)

code

Numero intero compreso tra 100000 e 999999 (Codice di attivazione dell'utente)

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempi:

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	
3	Riccardo	margherita	

```
$a = qpUsers:: verify_new_password (‘1’, 202020);
// $a = TRUE

$b = qpUsers:: verify_new_password (‘1’, 282828);
// $b = FALSE
```

set_new_password

Descrizione:

Booleano **qpUsers:: set_new_password** (numero intero *$id,* stringa *$password*)

Modifica la password di un utente resettando lo stato di possibilità per la modifica della password.

Parametri:

id

Numero intero (Id utente)

password

Stringa contenente la nuova password.

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempi:

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	303030
3	Riccardo	margherita	808000

```
$a = qpUsers:: set_new_password (‘1’, ‘nuovabicicletta);
// $a = TRUE
```

id	user	password	new_password
1	Giovanni	nuovabicicletta	0
2	Silvio	armadio	303030
3	Riccardo	margherita	808000

login

Descrizione:

Booleano **qpUsers:: login** (numero intero *$user,* stringa *$password*)

Verifica i valori **user** e **password** con le credenziali inserite nel database. Se combaciano inizializza una sessione utente.

NECESSITA DI UNA SESSIONE GIA' INIZIALIZZATA CON **session_start()**

Parametri:

user

Con ***qpUsers_basic->default_login = 'username'*** **qpUsers:: login** effettua una ricerca per username.
Con ***qpUsers_basic->default_login = 'email'*** **qpUsers:: login** effettua una ricerca per email.

password

Stringa contenente la password.

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Approfondimento:

qpUsers:: login crea le seguenti valiabili $_SESSION[…]:

```
$_SESSION['qp_user_id'] = ID utente
$_SESSION['qp_user_status'] = Stato utente
$_SESSION['qp_user_permit'] = Permessi utente
$_SESSION['qp_user_email'] = Email utente
$_SESSION['qp_user_username'] = Username utente
```

Esempi:

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	303030
3	Riccardo	margherita	808000

```
$a = qpUsers:: login ('Giovanni', 'bicicletta');
// $a = TRUE
```

logout

Descrizione:

qpUsers:: login ()

Elimina l'ultima sessione inizializzata;

Parametri:

Non accetta parametri

Valore di ritorno:

Nessuno

Esempi:

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	303030
3	Riccardo	margherita	808000

```
$a = qpUsers:: login ('Giovanni', 'bicicletta');
// $a = TRUE

qpUsers:: logout
```

logout

Descrizione:

Booleano **qpUsers:: logged** ()

Permette di rilevare una sessione utente

Parametri:

Non accetta parametri

Valore di ritorno:

Valore Booleano (True in caso di sessione utente instaurata – False in caso di sessione utente nulla)

Esempi:

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	303030
3	Riccardo	margherita	808000

```
$a = qpUsers:: login (‘Giovanni’, ‘bicicletta’);
// $a = TRUE

$b = qpUsers:: logged()
// $b = TRUE

qpUsers:: logout ()

$b = qpUsers:: logged()
// $a = FALSE
```

log_info

Descrizione:

Mix **qpUsers:: log_info** (stringa *$type* = 'id')

Ritorna con uno dei valori sessione a seconda della richiesta **type**

Parametri:

type

Puo’ assumere i seguenti valori: id, status, permit, email, username, all,

Valore di ritorno:

Stringa con **type** id, status, permit, email, username, o nullo.
Array con **type** all

```
array(
        'id'       => Id utente,
        'status' =>  Stato utente,
        'permit' => Permessi utente,
        'email'  => email utente,
        'username'=> username utente
);
```

Esempio:

id	user	password	new_password
1	Giovanni	bicicletta	202020
2	Silvio	armadio	303030
3	Riccardo	margherita	808000

```
$a = qpUsers:: login (‘Giovanni’, ‘bicicletta’);
// $a = TRUE

$b = qpUsers:: log_info(‘username’)
// $b = Giovanni
```

new_profile

Descrizione:

Booleano **qpUsers:: new_profile** (numero intero *$id*, stringa *$plate*)

Verifica **plate** tramite **qpFormCheck::plate()** e inserisce nel profilo utente con numero identificativo **id** la targa **plate**.

Parametri:

id

Numero intero (Id utente)

plate

Stringa contenente la targa.

Valore di ritorno:

Valore Booleano (True in caso di successo – False in caso di insuccesso)

Esempio:

```
$a = qpUsers:: login (‘1, ‘AA000AA);
// $a = TRUE
```

qpUsers

qpUsers è un estensione della classe qpUsers_basic

Struttura

```
class qpUser_statement extends qpUsers_basic {

        public $prepared
        public $successful

        public $id
        public $status
        public $permit
        public $activation_code
        public $new_password
        public $email
        public $username
        public $password

        public $plate

        public $havepost
        protected static $sql

        public function __construct($type, $data)

        public function insert()
        public function update()

        public function ulist_fetch()
        public function uprofiles_fetch()
}
```

Metodi della libreria qpUser_statement:

Costruttore

Descrizione:

qpUsers_basic:: __construct ($type, $data)

Inizializza qpUser_statement, attribuendo alle proprietà i valori necessari da utilizzare con i suoi metodi.

insert

Descrizione:

	qpUsers:: insert () Utilizza le proprietà della classe per inserire i dati preparati tramite qpUsers::prepare() nel database.
	## update **Descrizione:** **qpUsers:: update** () Utilizza le proprietà della classe per editare i dati preparati tramite qpUsers::prepare().

Funzioni sulla base della classe qpUsers

Di seguito alcune funzioni per facilitare la programmazione procedurale basate sulla classe qpUsers

ULIST

Funzioni per visualizzare una lista utenti

function **load_UsersList ($s = 0, $q = 10)**
function **loop_UsersList()**
function **id()**
function **status()**
function **permit()**
function **activation_code()**
function **email()**
function **username()**
function **password()**

load_UsersList

Descrizione:

load_UsersList (numero intero *$s* = 0, numero intero *$q* =10,)

Effettua la funzione qpUsers::ulist per caricare un oggetto qpUsers_statement per realizzazione di una lista utenti

Parametri:

s

Valore di partenza per la ricerca nel database

q

Numero di righe da caricare dal database

SQL: **LIMIT $s, $q**

Valore di ritorno:

Nessuno

loop_UsersList

Descrizione:

Booleano **loop_UsersList** ()

Da utilizzare in un loop.
Carica la variabile globale $qpUser_ulist_data contenente i valori caricati dal database

Parametri:

Non accetta parametri

Valore di ritorno:

Nessuno

id

Descrizione:

Echo **id** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza l'ID dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

status

Descrizione:

Echo **status** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza lo stato dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

permit

Descrizione:

Echo **permit** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza i permessi dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

Activation_code

Descrizione:

Echo **activation_code** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza il codice di attivazione dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

email

Descrizione:

Echo **email** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza l'email dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

username

Descrizione:

Echo **username** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza l'username dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

password

Descrizione:

Echo **password** ()

Da utilizzare in un loop caricato tramite **loop_UsersList**.
Visualizza la password dell'utente

Parametri:

Non accetta parametri

Valore di ritorno:

Echo

ESEMPIO DI UTILIZZO ULIST

id	user	email
1	Giovanni	Giovanni@host.it
2	Silvio	Silvio@host.it
3	Riccardo	Riccardo@host.it

```
load_UsersList(0,5);
while(loop_UsersList()) :
        id(); echo ' - ';
        email(); echo ' - ';
        username(); echo ' <br> ';
endwhile;
a = qpUsers:: login ('1, 'AA000AA);

// RISULTATO

1 - Giovanni - Giovanni@host.it
2 - Silvio - Silvio@host.it
3 - Riccardo - Riccardo@host.it
```