Conception de sites web dynamique

HTML – CSS – JAVASCRIPT – PHP
BASE DE DONNÉES

IF04U050

David Bouchet

david.bouchet.paris5@gmail.com

Conception de sites web dynamique

PHP
5e partie

PostgreSQL – Mettre à jour une table

Instruction *SQL* : UPDATE

```
$query[] = "UPDATE customer SET age = 16 WHERE id = 4";
$query[] = "UPDATE customer SET email = 'lea.s@yh.com' WHERE id = 6";

foreach ($query as $q)
    pg_query($q);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
4	roger	15	roger.b@ht.fr
5	jade	32	jade15@ht.fr
6	léa	24	lea@gm.com
7	jade	27	jade.s@gm.com

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr
6	léa	24	lea.s@yh.com

PostgreSQL – Supprimer une ligne

Instruction *SQL* : DELETE FROM

```
$query = "DELETE FROM customer WHERE id = 6";
pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr
6	léa	24	lea.s@yh.com

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

PostgreSQL – Extraction de données

Une requête d'extraction $result = pg_query($query) avec un SELECT renvoie une **référence sur une table de résultats**

Comment y accéder ?

$nb = **pg_num_rows**($result) → nombre de lignes de la table résultat

$nb = **pg_num_fields**($result) → nombre de colonnes

$row = **pg_fetch_row**($result) → ligne courante sous forme de **tableau** indicé et incrémente le pointeur de ligne

$row = **pg_fetch_assoc**($result) → ligne courante sous forme de **tableau** associatif ; incrémente le pointeur de ligne

$row = **pg_fetch_object**($result) → ligne courante sous forme d'**objet** et incrémente le pointeur de ligne

PostgreSQL – Extraction de données

Récupérer toutes les entrées de la table *customer*.

```
$query = "SELECT * FROM customer";
$result = pg_query($query) or exit;
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result contient la table ci-contre.

```
echo pg_num_rows($result)."<br />";      // Affiche : 6
echo pg_num_fields($result)."<br />";    // Affiche : 4
```

PostgreSQL – pg_fetch_row() (1)

PHP

```
while ($ligne = pg_fetch_row($result))
{
    print_r($ligne);
    echo "<br />";
}
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

Affichage

```
Array ( [0] => 1 [1] => roger [2] => 24 [3] => roger@gm.com )
Array ( [0] => 2 [1] => david [2] => 35 [3] => david@yh.com )
Array ( [0] => 3 [1] => max [2] => 24 [3] => max@gm.com )
Array ( [0] => 5 [1] => jade [2] => 32 [3] => jade15@ht.fr )
Array ( [0] => 7 [1] => jade [2] => 27 [3] => jade.s@gm.com )
Array ( [0] => 4 [1] => roger [2] => 16 [3] => roger.b@ht.fr )
```

PostgreSQL – pg_fetch_row() (2)

PHP

```php
while ($ligne = pg_fetch_row($result))
{
    echo "id = ".$ligne[0]."; ";
    echo "name = ".$ligne[1]."; ";
    echo "age = ".$ligne[2]."; ";
    echo "email = ".$ligne[3];
    echo "<br />";
}
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

Affichage

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 2; name = david; age = 35; email = david@yh.com
id = 3; name = max; age = 24; email = max@gm.com
id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 7; name = jade; age = 27; email = jade.s@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr
```

PostgreSQL – pg_fetch_assoc() (1)

PHP

```
while ($ligne = pg_fetch_assoc($result))
{
    print_r($ligne);
    echo "<br />";
}
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

Affichage

```
Array ( [id] => 1 [name] => roger [age] => 24 [email] => roger@gm.com )
Array ( [id] => 2 [name] => david [age] => 35 [email] => david@yh.com )
Array ( [id] => 3 [name] => max [age] => 24 [email] => max@gm.com )
Array ( [id] => 5 [name] => jade [age] => 32 [email] => jade15@ht.fr )
Array ( [id] => 7 [name] => jade [age] => 27 [email] => jade.s@gm.com )
Array ( [id] => 4 [name] => roger [age] => 16 [email] => roger.b@ht.fr )
```

PostgreSQL – pg_fetch_assoc() (2)

PHP

```
while ($ligne = pg_fetch_assoc($result))
{
    foreach ($ligne as $k => $v)
        echo $k." = ".$v."; ";
    echo "<br />";
}
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

Affichage

```
id = 1; name = roger; age = 24; email = roger@gm.com;
id = 2; name = david; age = 35; email = david@yh.com;
id = 3; name = max; age = 24; email = max@gm.com;
id = 5; name = jade; age = 32; email = jade15@ht.fr;
id = 7; name = jade; age = 27; email = jade.s@gm.com;
id = 4; name = roger; age = 16; email = roger.b@ht.fr;
```

PostgreSQL – pg_fetch_object() (1)

Permet de renvoyer une ligne dans un objet.

```
object pg_fetch_object($result [,$row [,$class_name]]);
```

$row = Numéro de la ligne à récupérer. Les lignes sont numérotées en commençant à 0. Si l'argument est omis ou s'il vaut *NULL*, la ligne suivante est récupérée.

$class_name = Le nom de la classe à instancier. Les attributs de la classe doivent correspondre aux champs de la table (*).

(*) Les attributs seront modifiés en fonction de la valeur des champs (même s'ils sont déclarés privés). Si les attributs n'existent pas, ils seront crées automatiquement et auront un accès public.

Du point de vue vitesse, la fonction est identique à *pg_fetch_array()* et est presque aussi rapide que *pg_fetch_row()* (la différence est insignifiante).

PostgreSQL – pg_fetch_object() (2)

PHP

```
while ($ligne = pg_fetch_object($result))
{
    echo "id = ".$ligne->id."; ";
    echo "name = ".$ligne->name."; ";
    echo "age = ".$ligne->age."; ";
    echo "email = ".$ligne->email;
    echo "<br />";
}
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

Affichage

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 2; name = david; age = 35; email = david@yh.com
id = 3; name = max; age = 24; email = max@gm.com
id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 7; name = jade; age = 27; email = jade.s@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr
```

PostgreSQL – pg_fetch_object() (3)

PHP

```php
class Customer
{
    private $id;
    private $name;
    private $age;
    private $email;

    function Display()
    {
        echo "id = ".$this->id."; ";
        echo "name = ".$this->name."; ";
        echo "age = ".$this->age."; ";
        echo "email = ".$this->email;
        echo "<br />";
    }
}

while ($ligne = pg_fetch_object($result, NULL, "Customer"))
    $ligne->Display();
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

Affichage

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 2; name = david; age = 35; email = david@yh.com
id = 3; name = max; age = 24; email = max@gm.com
id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 7; name = jade; age = 27; email = jade.s@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr
```

PostgreSQL – Libération des ressources

bool **pg_free_result**($result)

Libère la mémoire et les données associées avec le jeu de résultats passé en paramètre.

pg_free_result() n'est vraiment utile que si vous risquez d'utiliser trop de mémoire durant votre script. La mémoire occupée par les résultats est automatiquement libérée à la fin du script.

Valeurs de retour : Renvoie *TRUE* en cas de succès ou *FALSE* si une erreur survient.

PostgreSQL – Critères de sélection (1)

Sélectionner les clients dont le nom est *roger*.

```
$query = "SELECT * FROM customer WHERE name='roger'";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

id	name	age	email
1	roger	24	roger@gm.com
4	roger	16	roger.b@ht.fr

} *$result*

PostgreSQL – Critères de sélection (2)

Sélectionner les clients dont l'âge est inférieur à 30 ans.

```
$query = "SELECT * FROM customer WHERE age < 30";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

id	name	age	email
1	roger	24	roger@gm.com
3	max	24	max@gm.com
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

$result

PostgreSQL – Critères de sélection (3)

Sélectionner les clients dont l'e-mail appartient à *gm.com*.

```
$query = "SELECT * FROM customer WHERE email LIKE '%gm.com'";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

id	name	age	email
1	roger	24	roger@gm.com
3	max	24	max@gm.com
7	jade	27	jade.s@gm.com

$result

PostgreSQL – Trier une table résultat (1)

Récupérer uniquement le nom et l'age des clients du plus jeune au plus vieux.

```
$query = "SELECT name, age FROM customer ORDER BY age";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

name	age
roger	16
roger	24
max	24
jade	27
jade	32
david	35

$result

PostgreSQL – Trier une table résultat (2)

Récupérer uniquement le nom et l'age des clients du plus vieux au plus jeune.

```
$query = "SELECT name, age FROM customer ORDER BY age DESC";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

name	age
david	35
jade	32
jade	27
roger	24
max	24
roger	16

$result

PostgreSQL – Limiter le nombre de ligne (1)

Récupérer uniquement le nom et l'age des clients du plus vieux au plus jeune en limitant le nombre de lignes à 2.

```
$query = "SELECT name, age FROM customer ORDER BY age DESC LIMIT 2";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

name	age
david	35
jade	32
jade	27
roger	24
max	24
roger	16

name	age
david	35
jade	32

$result

PostgreSQL – Limiter le nombre de ligne (2)

Récupérer uniquement le nom et l'age des clients du plus vieux au plus jeune en limitant le nombre de lignes à 2 en partant de la ligne 3.

```
$query = "SELECT name, age FROM customer ORDER BY age DESC LIMIT 2 OFFSET 3";
$result = pg_query($query);
```

id	name	age	email
1	roger	24	roger@gm.com
2	david	35	david@yh.com
3	max	24	max@gm.com
5	jade	32	jade15@ht.fr
7	jade	27	jade.s@gm.com
4	roger	16	roger.b@ht.fr

name	age
david	35
jade	32
jade	27
roger	24
max	24
roger	16

name	age
roger	24
max	24

$result

PostgreSQL – Requête préparée

Les requêtes préparées sont utiles quand on souhaite exécuter plusieurs fois une requêtes en changeant uniquement la valeur de certains paramètres.

Les fonctions relatives aux requêtes préparées sont les suivantes :

pg_query_params() : Envoie une commande au serveur et attend le résultat, avec les capacités de passer des paramètres séparément de la commande texte *SQL*.

pg_prepare() : Envoie une requête pour créer une requête préparée avec les paramètres donnés et attend l'exécution.

pg_execute() : Exécute une requête préparée.

PostgreSQL – pg_query_params() (1)

```
resource pg_query_params([$connection], $query, $params));
```

$query = La requête SQL avec ses paramètres. Si des paramètres sont utilisés, ils sont référés à *$1*, *$2*, etc.

$params = **Un tableau de valeurs** de paramètres pour substituer les variables *$1*, *$2*, etc. dans la requête préparée originale. Le nombre d'éléments présents dans le tableau doit concorder avec le nombre de variables à remplacer.

Valeurs de retour : Une ressource de résultats en cas de succès ou *FALSE* si une erreur survient.

PostgreSQL – pg_query_params() (2)

```php
$query = "SELECT * FROM customer WHERE age < $1";

$result = pg_query_params($query, array(30));
while ($ligne = pg_fetch_object($result, NULL, "Customer"))
        $ligne->Display();

echo "<br />";

$result = pg_query_params($query, array(25));
while ($ligne = pg_fetch_object($result, NULL, "Customer"))
        $ligne->Display();
```

PHP

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 3; name = max; age = 24; email = max@gm.com
id = 7; name = jade; age = 27; email = jade.s@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr

id = 1; name = roger; age = 24; email = roger@gm.com
id = 3; name = max; age = 24; email = max@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr
```

Affichage

PostgreSQL – pg_query_params() (3)

```php
$query = "SELECT * FROM customer WHERE name = $1 AND age < $2";

$result = pg_query_params($query, array("roger", 30));
while ($ligne = pg_fetch_object($result, NULL, "Customer"))
        $ligne->Display();

echo "<br />";

$result = pg_query_params($query, array("jade", 35));
while ($ligne = pg_fetch_object($result, NULL, "Customer"))
        $ligne->Display();
```

PHP

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr

id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 7; name = jade; age = 27; email = jade.s@gm.com
```

Affichage

PostgreSQL – pg_prepare() et *pg_execute()*

```php
$query = "SELECT * FROM customer WHERE name = $1 AND age < $2";
pg_prepare("MyQuery", $query);

$result = pg_execute("MyQuery", array("roger", 30));
while ($ligne = pg_fetch_object($result, NULL, "Customer"))
        $ligne->Display();

echo "<br />";

$result = pg_execute("MyQuery", array("jade", 35));
while ($ligne = pg_fetch_object($result, NULL, "Customer"))
        $ligne->Display();
```

PHP

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 4; name = roger; age = 16; email = roger.b@ht.fr

id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 7; name = jade; age = 27; email = jade.s@gm.com
```

Affichage

Intérêt des requêtes préparées (1)

Quelles différences entre les instructions suivantes ?

Concaténation de chaînes :

```
$query = "SELECT * FROM customer WHERE name =".$_GET['name1']." AND age < ".$_GET['age1'];
$result1 = pg_query($query);

$query = "SELECT * FROM customer WHERE name =".$_GET['name2']." AND age < ".$_GET['age2'];
$result2 = pg_query($query);
```

***pg_query_params()* :**

```
$query = "SELECT * FROM customer WHERE name = $1 AND age < $2";
$result1 = pg_query_params($query, array($_GET['name1'], $_GET['age2']));
$result2 = pg_query_params($query, array($_GET['name2'], $_GET['age2']));
```

***pg_prepare()* et *pg_execute()* :**

```
$query = "SELECT * FROM customer WHERE name = $1 AND age < $2";
pg_prepare("MyQuery", $query);

$result1 = pg_execute("MyQuery", array($_GET['name1'], $_GET['age2']));
$result2 = pg_execute("MyQuery", array($_GET['name2'], $_GET['age2']));
```

Intérêt des requêtes préparées (2)

pg_query_params(), *pg_prepare()*, *pg_execute()*

- Échappement automatique de caractères spéciaux. Les caractères spéciaux, comme les apostrophes ou les guillemets, peuvent être source d'erreur. Ceci permet d'éviter l'utilisation de fonctions spécifiques comme *pg_escape_string()* qui protège une chaîne de caractères pour une requête SQL.
- Protège contre les injections SQL.

pg_prepare(), *pg_execute()*

- Accélère le traitement d'une requête lorsqu'elle est exécutée plusieurs fois.

Injection SQL

Exemple :

```
$query = "SELECT * FROM customer WHERE age < ".$_GET['age1'];
$result = pg_query($query);

// Si $_GET['age1'] = "50; SELECT * FROM admin";
// La requête devient :
// SELECT * FROM customer WHERE age < 50; SELECT * FROM admin";
```

- En terminant la requête et en ajoutant un « ; », il devient possible d'ajouter une ou plusieurs requêtes à la suite de la première. Il est alors possible, dans certains cas, de récupérer des informations sur la base de données, d'ajouter des tables, des entrées ou encore d'ajouter un nouvel administrateur de la base.

- Ces injections SQL ne sont pas possible avec les fonctions *pg_query_params()*, *pg_prepare()* et *pg_execute()*. Les arguments sont traités directement par l'analyseur *SQL*, ce qui empêche ce type d'injection.

MySQL – Deux jeux de fonctions

Il existe deux jeux de fonctions différents pour s'interfacer avec une base de données MySQL :

- Les fonctions préfixées par *mysql* : mysql_<nom_fonction>.
- Les fonctions préfixées par *mysqli* : mysqli_<nom_fonction>.

L'extension *mysqli* (*MySQL Improved*) a été développée pour tirer parti des nouvelles fonctionnalités des systèmes *MySQL* version 4.1.3 et plus récent.

L'extension *mysqli* est incluse dans PHP depuis les versions 5.

Il est recommandé d'utiliser *mysqli* plutôt que *mysql*.

MySQL – Double Interface

L'extension *mysqli* fournit 2 interfaces.

Elle supporte
la programmation procédurale
mais aussi,
la programmation orientée objet.

Il n'y a pas de différence significative d'un point de vue performance entre les deux interfaces.

Les utilisateurs peuvent faire leur choix d'un point de vue personnel.

MySQL – Connexion (1)

Vous devez vous munir des informations suivantes :

- Nom de l'hôte.
- Nom de la base de données.
- Nom d'utilisateur.
- Mot de passe.

Base de données My*SQL* sur le serveur de Paris 5 :

- Nom de l'hôte : opale.ens.math-info.univ-paris5.fr
- Nom de la base de données : Dans le fichier "~/.my.cnf" (ex. gdupont)
- Nom d'utilisateur : <votre login> (ex. gdupont)
- Mot de passe : Dans le fichier "~/.my.cnf" (ex. uiot5434)

MySQL – Connexion (2)

Style procédural

```php
$link = mysqli_connect('opale.ens.math-info.univ-paris5.fr',
'gdupont', 'uiot5434', 'gdupont');

if ($link === false)
    exit(mysqli_connect_errno()." : ".mysqli_connect_error());

echo "La connexion a réussi";
```

Style orienté objet

```php
$mysqli = new mysqli('opale.ens.math-info.univ-paris5.fr',
'gdupont', 'uiot5434', 'gdupont');

if ($mysqli->connect_errno)
    exit($mysqli->connect_errno." : ".$mysqli->connect_error);

echo "La connexion a réussi";
```

MySQL – Déconnexion

Style procédural

```php
$link = mysqli_connect('opale.ens.math-info.univ-paris5.fr',
'gdupont', 'uiot5434', 'gdupont');

// Traitements sur la base de données...

// Déconnexion (facultative en fin de script).
mysqli_close($link)
```

Style orienté objet

```php
$mysqli = new mysqli('opale.ens.math-info.univ-paris5.fr',
'gdupont', 'uiot5434', 'gdupont');

// Traitements sur la base de données...

// Déconnexion (facultative en fin de script).
$mysqli->close()
```

MySQL – Exécuter une requête

Les fonctions d'exécution de requête (procédurale et orientée objet) :

- Renvoie *FALSE* en cas d'erreur.
- Renvoie une **table de résultat** s'il s'agit d'une requête d'extraction de données (ex. SELECT).
- Renvoie *TRUE* s'il ne s'agit pas d'une requête d'extraction de données (ex. CREATE TABLE).

Les fonctions *mysqli_errno()* et *mysqli_error()* (style procédural) ainsi que les attributs *errno* et *error* (style orienté objet) renvoient différentes informations sur la dernière erreur survenue

MySQL – Créer une table (1)

Style procédural

```
$query = "CREATE TABLE customer
          (id INT AUTO_INCREMENT PRIMARY KEY,
           name VARCHAR(50),
           age SMALLINT,
           email VARCHAR(64) UNIQUE)";

$result = mysqli_query($link, $query);

if ($result === true)
    echo "La table a été créée";
else
    echo mysqli_errno($link)." : ".mysqli_error($link);
```

MySQL – Créer une table (2)

Style orienté objet

```
$query = "CREATE TABLE customer
          (id INT AUTO_INCREMENT PRIMARY KEY,
           name VARCHAR(50),
           age SMALLINT,
           email VARCHAR(64) UNIQUE)";

$result = $mysqli->query($query);

if ($result === true)
    echo "La table a été créée";
else
    echo $mysqli->errno." : ".$mysqli->error;
```

MySQL – Remplir une table

```
$query[] = "INSERT INTO customer VALUES('', 'roger', 24, 'roger@gm.com')";
$query[] = "INSERT INTO customer VALUES('', 'david', 35, 'david@yh.com')";
$query[] = "INSERT INTO customer VALUES('', 'max', 24, 'max@gm.com')";
$query[] = "INSERT INTO customer VALUES('', 'roger', 15, 'roger.b@ht.fr')";
$query[] = "INSERT INTO customer VALUES('', 'jade', 32, 'jade15@ht.fr')";
$query[] = "INSERT INTO customer VALUES('', 'léa', 24, 'lea@gm.com')";
$query[] = "INSERT INTO customer VALUES('', 'jade', 27, 'jade.s@gm.com')";
```

Style procédural

```
foreach ($query as $q)
    if (mysqli_query($link, $q) === false)
        echo mysqli_errno($link)." : ".mysqli_error($link);
```

Style orienté objet

```
foreach ($query as $q)
    if ($mysqli->query($q) === false)
        echo $mysqli->errno." : ".$mysqli->error;
```

MySQL – Extraction de données (1)

Procédural	Orienté objet	Description
mysqli_num_rows()	*num_rows*	Le nombre de lignes dans un résultat (similaire à *pg_num_rows()*).
mysqli_num_fields()	*num_fields*	Le nombre de champs dans un résultat (similaire à *pg_num_fields()*).
mysqli_fetch_row()	*fetch_row()*	Récupère une ligne de résultat sous forme de tableau indexé (similaire à *pg_fetch_row()*).
mysqli_fetch_assoc()	*fetch_assoc()*	Récupère une ligne de résultat sous forme de tableau associatif (similaire à *pg_fetch_assoc()*).
mysqli_fetch_object()	*fetch_object()*	Retourne la ligne courante d'un jeu de résultat sous forme d'objet (similaire à *pg_fetch_object()*).
mysqli_free_result()	*close()*	Libère la mémoire.

MySQL – Extraction de données (2)

```php
$result = mysqli_query($link, "SELECT * FROM customer");

while ($row = mysqli_fetch_assoc($result))
{
    echo "id = ".$row['id']."; ";
    echo "name = ".$row['name']."; ";
    echo "age = ".$row['age']."; ";
    echo "email = ".$row['email'];
    echo "<br />";
}

echo "Nombre de lignes : ".mysqli_num_rows($result);
mysqli_free_result($result);
```

Style procédural

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 2; name = david; age = 35; email = david@yh.com
id = 3; name = max; age = 24; email = max@gm.com
id = 4; name = roger; age = 15; email = roger.b@ht.fr
id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 6; name = léa; age = 24; email = lea@gm.com
id = 7; name = jade; age = 27; email = jade.s@gm.com
Nombre de lignes : 7
```

Affichage

MySQL – Extraction de données (3)

```php
$result = $mysqli->query("SELECT * FROM customer");

while ($row = $result->fetch_assoc())
{
    echo "id = ".$row['id']."; ";
    echo "name = ".$row['name']."; ";
    echo "age = ".$row['age']."; ";
    echo "email = ".$row['email'];
    echo "<br />";
}

echo "Nombre de lignes : ".$result->num_rows;
$result->close();
```

Style orienté objet

```
id = 1; name = roger; age = 24; email = roger@gm.com
id = 2; name = david; age = 35; email = david@yh.com
id = 3; name = max; age = 24; email = max@gm.com
id = 4; name = roger; age = 15; email = roger.b@ht.fr
id = 5; name = jade; age = 32; email = jade15@ht.fr
id = 6; name = léa; age = 24; email = lea@gm.com
id = 7; name = jade; age = 27; email = jade.s@gm.com
Nombre de lignes : 7
```

Affichage