Chapter 12 PHP와 MySQL 연동

PHP와 MySQL 사이를 연동하기 위한 API는 MySQLi 확장 그리고 PDO(Php Data Object) 방법이 있습니다. 본 도서에서는 MySQLi API를 이용하여 연동하는 과정을 살펴봅니다.
또한 MySQLi API에는 절차지향과 객체지향의 2가지 방식이 있습니다. 이 2가지 방식의 차이점을 소개하고 객체지향 방식으로 PHP와 MySQL의 연동에 대해 설명하겠습니다.

1. MySQLi 확장

MySQLi에서 i의 의미는 improved 즉 향상되었다는 의미이며, 기존의 MySQL 보다 보안, 속도면에서 향상되었다고 합니다.
MySQLi 확장 API는 MySQL 4.1 이후 버전, PHP 5.0 이후 버전부터 사용 가능합니다.
MySQLi 확장은 절차지향적, 객체지향적 API를 함께 제공하여 목적에 따라 적절한 방식을 선택하여 프로그램할 수 있습니다.

1.1 절차지향 vs 객체지향

절차지향(Procedual Oriented)은 프로그램을 절차적인 즉, 코드의 흐름 또는 순서를 중심으로 작성해 나가는 방식을 말합니다.
객체지향(Object Oriented)은 프로그램을 객체 위주로 즉, 객체가 주어(또는 중심)가 되어 작성해 나가는 방식을 말합니다. 그래서 저자는 객체지향이라는 단어보다는 객체주어 또는 객체중심 이라는 단어를 사용합니다.

1.2 MySQLi API 코딩 스타일

PHP 언어로 MySQL에 접속하여 질의문을 처리하는 코드입니다. 절차지향 방식과 객체주어(중심) 방식으로 작성되었으며 동일한 역할을 수행합니다.
(1) 절차지향 방식

```
1  <?php
2  $host = "localhost";
3  $user = "root";
4  $link = mysqli_connect($host, $user);
5  mysqli_select_db($link, "friend");
6  $result = mysqli_query($link, "SELECT name FROM friend");
7  $number = mysqli_num_rows($result);
8  mysqli_close($link);
9  ?>
```

- 4번 라인 : mysqli_connect 함수를 사용하여 접속 관련 시스템 자원을 $link 변수에 저장합니다.
- 5번 라인 : mysqli_select_db 함수는 $link의 접속정보를 이용하여 friend 데이터베이스 선택합니다.
- 6번 라인 : mysqli_query 함수는 $link 접속정보를 이용하여 SELECT 질의문을 수행하고 그 결과를 $result 변수에 저장합니다.
- 7번 라인 : mysqli_num_rows 함수는 $result 변수에 저장된 결과에서 행의 개수를 계산하

여 $number 변수에 저장합니다.
- 8번 라인 : mysqli_close 함수는 $link 접속을 종료합니다.
코드 설명에서 보듯이 절차지향 방식은 함수가 주어가 되어 $link, $result, $number 변수를 목적어처럼 사용하는 방식으로 수행됩니다.

(2) 객체주어(중심) 방식

```
1  <?php
2  $host = "localhost";
3  $user = "root";
4  $link = new mysqli($host, $user);
5  $link->select_db("friend");
6  $result = $link->query("SELECT name FROM friend");
7  $number = $result->num_rows;
8  $link->close();
9  ?>
```

- 4번 라인 : mysqli 클래스를 사용하여 접속 관련 시스템 자원을 $link 객체 인스턴스에 저장합니다.
- 5번 라인 : $link 객체 인스턴스는 select_db 메서드를 이용하여 friend 데이터베이스를 선택합니다.
- 6번 라인 : $link 객체 인스턴스는 query 메서드를 이용하여 SELECT 질의문을 수행하고 그 결과를 $result 객체 인스턴스에 저장합니다.
- 7번 라인 : $result 객체 인스턴스는 num_rows 프로퍼티를 이용하여 행의 개수를 $number 변수에 저장합니다.
- 8번 라인 : $link 객체 인스턴스는 close 메서드를 이용하여 접속을 종료합니다.
코드 설명에서 보듯이 객체중심 방식은 객체가 주어가 되어 메서드와 프로퍼티를 동사 또는 목적어처럼 사용하는 방식으로 수행됩니다.

(3) 혼합형
언제든지 스타일 간에 전환할 수 있지만 코드 명확성과 코딩 스타일상의 이유로 두 스타일을 혼합하는 것은 권장하지 않지만, 다음과 같이 두 스타일을 혼합해도 정상적으로 처리는 됩니다.

```
   <?php
1  $host = "localhost"; $user = "root";
2  $link = new mysqli($host, $user);
3  $link->select_db("friend");
4  $result = $mysqli_query($link, "SELECT name FROM friend");
5  $number = $result->num_rows;
6  $link->close();
   ?>
```

두 방식은 큰 성능 차이가 없습니다. 사용자는 개인 취향에 따라 선택할 수 있습니다.
여러분들은 어떤 방식은 선호하나요? 사실 어떤 방식을 사용해도 상관은 없습니다. 두가지 방식들의 개념만 확실히 알고 있다면 서로의 방식으로 전환하는 것은 어렵지 않다고 생각합

니다.
MySQL에서 절차지향 방식으로 만든 기존의 프로그램을 재활용하려면 MySQLi 절차지향 방식을 사용해야 하겠고, 아직도 절차지향 방식이 많이 사용되기도 합니다.
객체중심 방식은 새롭게 MySQLi를 접하는 학습자에게 적합하다고 할 수 있지만, 이 역시 꼭 맞다고 할 수도 없습니다. 참여하는 프로젝트에서 어떤 방식으로 진행하느냐에 따라 다를 수도 있으니까요.
어쨌든 다양하고 복잡한 이유로 최종 선택을 하게 되는데 본 도서에서는 객체중심 방식으로 진행하겠습니다.
이유는 딱 한가지... 모든 함수명에 mysqli를 접두사로 사용하기 때문에 코드를 읽어나가는데 좀 불편합니다. 개인적인 생각입니다.

2. 사용자 계정의 비밀번호 설정

지금까지 명령 프롬프트 콘솔을 통해 MySQL DBMS에 접속할 때 비밀번호 없이 접속했습니다. 앞으로의 에제를 위해 꼭 필요한 건 아니지만 사용자 계정에 비밀번호를 설정하는 것은 매우 중요합니다. root 계정에 대해 비밀번호를 설정하겠습니다.
사용자 계정에 대한 비밀번호 설정은 mysql 데이터베이스의 user 테이블에서 진행합니다. user 테이블의 구조도 살펴봅니다.

```
C:\Users\Jaeho-L>mysql -hlocalhost -uroot
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
+--------------------+
MariaDB [(none)]> use mysql;
Database changed
MariaDB [mysql]> show tables;
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| column_stats              |
        :
| user                      |
+---------------------------+
MariaDB [mysql]> desc user;
+------------------------+---------------------+------+-----+----------+-------+
| Field                  | Type                | Null | Key | Default  | Extra |
+------------------------+---------------------+------+-----+----------+-------+
| Host                   | char(60)            | NO   |     |          |       |
| User                   | char(80)            | NO   |     |          |       |
| Password               | longtext            | YES  |     | NULL     |       |
   :
```

2.1 user 테이블의 password 필드

user 테이블에서 주의깊게 살펴볼 필드는 Host, User, Password입니다. 각 필드에 기본적으로 어떤 데이터 값이 있는지 확인해 봅니다. Host 필드에는 localhost, 127.0.0.1 값이 있고, User에는 root, Password에는 어떤 값도 없네요. 즉 localhost에 접속할 root 계정의 비밀번호는 설정되지 않은 상태입니다(user 테이블에 user 필드가 있는 점 주의하세요).

```
MariaDB [mysql]> select host, user, password from user;
+-----------+------+----------+
| Host      | User | Password |
+-----------+------+----------+
| localhost | root |          |
| 127.0.0.1 | root |          |
| ::1       | root |          |
| localhost | pma  |          |
+-----------+------+----------+
```

2.2 root 계정의 비밀번호 설정

root 계정의 비밀번호를 1234로 설정해 보겠습니다. 실제로는 이런 비밀번호를 설정하면 절대로 안됩니다.

비밀번호 설정 방법은 SET 구문과 password 함수를 이용하여 user 테이블의 root에 대해 password 값을 설정하면 됩니다. 그리고 이런 변경은 환경설정의 변경에 해당되므로 flush privileges 명령은 실행하여 새로운 설정을 적용시켜야 합니다.

환경설정 변경은 계정 및 패스워드가 추가, 변경되는 경우이며, 데이터의 추가, 삭제, 수정 등에는 해당되지 않습니다.

제대로 변경되었는지 select 구문을 통해 확인해 봅니다.

```
MariaDB [mysql]> SET PASSWORD=PASSWORD('1234');
MariaDB [mysql]> flush privileges;
MariaDB [mysql]> select host, user, password from user;
+-----------+------+-------------------------------------------+
| Host      | User | Password                                  |
+-----------+------+-------------------------------------------+
| localhost | root | *A4B6157319038724E3560894F7F932C8886EBFCF |
| 127.0.0.1 | root |                                           |
| ::1       | root |                                           |
```

2.3 비밀번호 설정후 mysql 서버 접속

비밀번호가 변경되면 앞으로 MySQL에 접속하려면 비밀번호를 입력해야 합니다.

```
MariaDB [(none)]> exit
C:\Users\Jaeho-L>mysql -hlocalhost -uroot
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
NO)
C:\Users\Jaeho-L>mysql - hlocalhost -uroot -p1234
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.4.24-MariaDB mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]>
```

3. MySQLi 확장 함수와 상수

3.1 MySQLi 미리 정의된 상수

MySQLi 확장에서는 미리 정의된 상수를 선언해 놓고 있습니다. 가장 많이 사용되는 상수를 다음 표에 제시합니다. 모든 MySQLi 상수들은 php.net 사이트에서 확인 바랍니다.

상수	설명
MYSQLI_ASSOC	배열 데이터의 인덱스를 필드이름으로 반환
MYSQLI_NUM	배열 데이터의 인덱스를 숫자로 반환
MYSQLI_BOTH	배열 데이터의 인덱스를 필드이름, 숫자로 반환

3.2 MySQLi 확장 함수

MySQLi 확장 함수들의 일부만 다음 표에 제시합니다. 모든 MySQLi 확장 함수들은 php.net 사이트에서 확인 바랍니다.

객체중심 인터페이스		절차지향 인터페이스
프로퍼티 (30개)	$mysqli::affected_rows	mysqli_affected_rows()
	$mysqli::client_info	mysqli_get_client_info()
	: : :	: : :
	$mysqli_result::lengths	mysqli_fetch_lengths()
	$mysqli_result::num_rows	mysqli_num_rows()
메서드 (70개)	mysqli::autocommit()	mysqli_autocommit()
	mysqli::query()	mysqli_query()
	: : :	: : :
	mysqli_result::fetch_array()	mysqli_fetch_array()
	mysqli_result::fetch_field()	mysqli_fetch_field()

위 표에서 보듯이 절차지향의 함수명들에는 『mysqli_』라는 접두어가 공통적으로 표시되어 있습니다. 객체 중심에서는 프로퍼티와 메서드로 표현되며, 같은 명칭의 절차지향 함수명과 동일한 역할을 수행합니다.

객체 중심의 프로퍼티와 메서드의 명칭 앞에는 『$mysqli::』 또는 『#mysqli_result::』가 표기되어 있는데, 이것은 객체 인스턴스의 이름이며 꼭 이 표기를 사용하지 않아도 됩니다.

$mysqli 명칭은 데이터베이스 접속정보를 저장하고 있는 객체이고, $mysqli_result 명칭은 SELECT 구문의 실행결과를 저장하고 있는 객체를 의미합니다.

위의 표에 있는 표기는 다음의 표기와 같은 의미입니다.

```
$mysqli::query()              ➡ $mysqli->query()
$mysqli_result::affected_rows ➡ $mysqli_result->affected_rows
```

일반적으로 $mysqli 명칭을 사용하지만, 개발자가 원하는 명칭을 사용해도 됩니다.

```
1  $mysqli = new mysqli($host, $user);
2  $mysqli->select_db("friend");
3  $mysqli_result = $mysqli->query("SELECT name FROM friend");
4  $number = $mysqli_result->fetch_array(MYSQLI_BOTH);
5  $mysqli->close();
```

다음 코드 일반적으로 사용하는 $mysqli 대신 $link를 사용하고 있고, $mysqli_result 대신 $result를 사용하고 있습니다.

```
1  $link = new mysqli($host, $user);
2  $link->select_db("friend");
3  $result = $link->query("SELECT name FROM friend");
4  $number = $result->fetch_array(MYSQLI_BOTH);
5  $link->close();
```

4. PHP와 MySQL의 연동

PHP와 MySQL의 연동은 기본적으로 다음과 같은 순서로 진행됩니다.

① MySQL 접속 ➡ ② 데이터베이스 (생성) 선택 ➡ ③ 쿼리문 실행 ➡ ④ MySQL 종료

사용되는 PHP 예제 파일들은 『H:\xampp\htdocs\source\12』에 있습니다. PHP 파일들의 실행은 반드시 Apache 웹서버가 동작되는 환경이어야 합니다. 즉 브라우저의 주소창에 PHP 파일들을 실행시켜야 하며 경로는 『localhost/source/12/file.php』 형태로 접속해야 합니다.

4.1 MySQL 접속 : new mysqli(), mysqli_connect()

mysqli()는 접속정보 객체 인스턴스의 생성을 위한 클래스입니다. MySQL DBMS에 접속하기 위해서 가장 먼저 수행해야 할 작업은 mysqli() 클래스의 인스턴스 객체를 생성하는 것인데, 이 객체는 MySQL 접속과 관련된 정보를 저장하고 있는 시스템 자원입니다. 이 객체가 각종 프로퍼티와 메서드들을 사용하여 다양한 쿼리문을 수행하게 됩니다.

객체중심	`$link = new mysqli('localhost', 'user', 'password', 'dbname');`
절차지향	`$link = mysqli_connect('localhost', 'user', 'password', 'dbname');`

2절에서 비밀번호를 생성했으므로 mysql 서버에 접속할 때 비밀번호를 지정합니다.

db_connect_1.php

```
1  <?php
2  //$link = new mysqli("localhost", "root", "1234");
3  $host = "localhost";
4  $user = "root";
5  $password = "1234";
6  $link = new mysqli($host, $user, $password);
7  //$link = mysqli_connect($host, $user, $password);
8  ?>
```

- 1번 라인 : 호스트명(localhost), 계정(root), 비밀번호(1234)를 문자열로 직접 입력
- 2~4번 라인 : 접속을 위한 기본 정보를 $host, $user, $password 변수에 저장
- 5번 라인 : mysqli() 클래스의 인수들을 호스트, 사용자, 비밀번호 순서로 new 키워드를 이용하여 새로운 접속정보 객체인 $link를 생성. 『C:\XAMPP>mysql -hlocalhost -uroot -p1234』 명령과 같은 의미
- 6번 라인 : 절차지향 방식

db_connect_2.php

```
1  <?php
2  $host = "localhost"; $user = "root"; $password = "1234";
3  $link = new mysqli($host, $user, $password);
4  if (!$link) {
5      die("DB 접속 실패" . $link->connect_error);
6  } else echo "DB 접속 성공<br>";
7  ?>
```

$link 객체가 정상적으로 생성되면 이 객체가 중심이 되어 MySQLi 확장 API가 제공하는 다양한 질의어를 처리하게 됩니다.
접속을 위한 정보가 잘못 기재되면 다음과 같은 에러가 발생합니다.

```
Fatal error: Uncaught mysqli_sql_exception: Access denied for user
'root'@'localhost' (using password: YES) ..... phpmysql\db_connect.php on line 6
```

4.2 기존 데이터베이스 선택

4.2.1 접속과 동시에 데이터베이스 선택

이미 만들어진 데이터베이스가 있다면 접속과 동시에 사용할 수 있도록 지정할 수 있습니다.

dbname_connect.php

```
<?php
$host = "localhost"; $user = "root"; $password = "1234";
$dbname = "mysql";
$link = new mysqli($host, $user, $password, $dbname);
if (!$link) {
    die("DB 접속 실패" . $link->connect_error);
} else echo "DB 접속 성공<br>";
?>
```

- 2번 라인 : 시스템 데이터베이스인 mysql을 $dbname 변수에 저장
- 3번 라인 : mysqli() 클래스의 4번째 인수로 $name을 지정하여 $link라는 접속정보 객체 인스턴스 생성.

4.2.2 접속후 데이터베이스 선택 : select_db()

MySQL에 접속한 후에 특정 데이터베이스 선택려면 select_db() 메서드를 사용합니다.
『MariaDB [mysql]>use mysql;』 구문과 같은 의미입니다.

객체중심	$link->select_db("DB명")	bool : 0 또는 1
절차지향	mysqli_select_db($link, "DB명")	

select_db.php

```
<?php
$host = "localhost"; $user = "root"; $password = "1234";
$link = new mysqli($host, $user, $password);
$mydb = $link->select_db("mysql");
//mysqli_select_db($link, "mysql");
?>
```

- 3번 라인 : 접속정보 객체인 $link에서 select_db() 메서드를 이용하여 mysql 데이터베이스를 선택

4.3 데이터베이스 생성

create_db_1.php

```
<?php
1 $host = "localhost"; $user = "root"; $password = "1234";
2 $link = new mysqli($host, $user, $password);
3 $link->query("create database testdb");
  //$link = mysqli_connect($host, $user, $password);
  //mysqli_query($link, "create database testdb");
  ?>
```

- 3번 라인 : $link 객체는 query() 메서드로 create 구문을 실행하여 데이터베이스를 생성.

query() 메서드는 다음에 살펴봅니다.

testdb 데이터베이스가 생성되면 명령프롬프트를 통해 확인해 보세요.

4.4 쿼리문 실행 : query()

query()는 쿼리문을 실행하여 그 결과값을 반환하는 메서드입니다.

객체중심	$link->query($sql)	쿼리문 실행결과
절차지향	mysqli_query($link, $sql)	

query() 메서드의 인수는 쿼리문 문자열을 지정해도 되지만, 변수에 문자열을 먼저 저장하고 그 변수를 인수로 지정해도 됩니다. 변수에 저장되는 쿼리문은 하나의 문장만 가능합니다.

```
1 $host = "localhost"; $user = "root"; $password = "1234";
2 $link = new mysqli($host, $user, $password);
3 $link->query("create database testdb");
4 $sql = "SELECT * FROM testdb";
5 $result = $link->query($sql);
  //$sql 대신 위 "SELECT ~" 문자열 지정 가능
  //$result = $link->query(SELECT name FROM friend");
```

- 4번 라인 : SELECT 구문은 $sql 변수에 저장

- 6번 라인 : 접속정보 객체인 $link에서 query() 메서드를 이용하여 $sql 변수에 저장된 쿼리문을 실행하고 그 결과를 $result 객체 변수에 저장

4.5 DBMS 종료 : close()

close()는 접속중인 DBMS를 종료하는 메서드이며 MySQL 콘솔에서 exit 명령에 해당됩니다.

객체중심	$link->close()	bool : 0 또는 1
절차지향	mysqli_close($link)	

db_close.php

```
1 $link = new mysqli($host, $user, $password);
2 $link->close();
3 //mysqli_close($link);
```