jinyPHP - 파트2 09. 데이터삽입

**09. 데이터삽입**

====================

테이블은 열(Rows) 기준으로 다수의 데이터를 각각의 컬럼에 맞게 가지고 있습니다. 열(rows)은 다른 말로 레코드라고 부릅니다. 레코드는 데이터베이스의 기본 저장 단위 입니다.

데이터 레코드는 한줄로 처리하며, 한줄의 레코드는 여러 개의 컬럼 데이터로 구성됩니다. 그리고 테이블은 레코드의 집합입니다.

**09.1 전체삽입**

====================

테이블에 새로운 데이터를 삽입을 해봅니다. SQL 쿼리명령 INSERT INTO 은 테이블에 새로운 레코드를 삽입합니다. 기본적으로 한줄의 컬럼을 묽어 레코드를 삽입을 합니다.

**쿼리문법)**

|  |
| --- |
| INSERT INTO 테이블명 VALUES (value1, value2, value3, ...); |

위의 INSERT INTO 기본 문법은 VALUES () 괄호 안의 데이터를 모든 컬럼을 대상으로 순차적으로 값을 1:1로 대입하여 입력을 처리합니다. 각각의 데이터는 콤마(,)로 구분합니다.

**09.1.1 쿼리 실습**

====================

입력되는 데이터는 테이블의 컬럼의 갯수 와 데이터를 갯수가 일치하여야 합니다. 만일 컬럼의 개수와 데이터의 개수가 일치하지 않으면 삽입명령 쿼리는 오류를 발생합니다. 먼저 테이블의 컬럼의 정보를 확인하기 위해서 DESC 명령을 실행해 봅니다.

|  |
| --- |
| mysql> desc members;  +-----------+--------------+------+-----+---------+----------------+  | Field | Type | Null | Key | Default | Extra |  +-----------+--------------+------+-----+---------+----------------+  | Id | int(11) | NO | PRI | NULL | auto\_increment |  | LastName | varchar(255) | YES | | NULL | |  | FirstName | varchar(255) | YES | | NULL | |  | Address | varchar(255) | YES | | NULL | |  | City | varchar(255) | YES | | NULL | |  | Country | varchar(255) | YES | | NULL | |  +-----------+--------------+------+-----+---------+----------------+  6 rows in set (0.02 sec) |

컬럼 필드의 갯수에 맞게 insert into 삽입을 명령합니다.

|  |
| --- |
| insert into members values ('','hojin','lee','shinchon','seoul','korea'); |

|  |
| --- |
| mysql> insert into members values ('','hojin','lee','shinchon','seoul','korea');  Query OK, 1 row affected, 1 warning (0.01 sec) |

데이터 삽입이 정상적으로 처리가 되었습니다. 정상적으로 입력이 되었는지 select 를 통하여 입력된 데이터를 확인합니다.

|  |
| --- |
| mysql> select \* from members;  +----+----------+-----------+----------+-------+---------+  | Id | LastName | FirstName | Address | City | Country |  +----+----------+-----------+----------+-------+---------+  | 1 | hojin | lee | shinchon | seoul | korea |  +----+----------+-----------+----------+-------+---------+  1 row in set (0.00 sec) |

입력한 데이터가 잘 저장이 되었습니다.

**09.2 선택삽입**

====================

첫번째 데이터 삽입방법은 컬럼 갯수에 맞게 전체의 데이터를 입력하는 방식이었습니다. 데이터를 입력하다 보면 전체의 데이터를 모두 입력을 할 필요가 없는 경우도 많이 있습니다.

INSERT 쿼리명령시 테이블 이름 뒤에 괄호()를 이용하여 입력하고자 하는 컬럼명을 지정할 수 있습니다. 지정된 컬럼만 선택되어 데이터를 삽입을 하게 됩니다. 이때에도 지정한 컬럼의 개수와 데이터의 개수는 일치 하여야 합니다.

**쿼리문법)**

|  |
| --- |
| INSERT INTO 테이블명 (컬럼명1, 컬럼명2, 컬럼명3, ...) VALUES (value1, value2, value3, ...); |

**09.2.1 쿼리 실습**

====================

컬럼을 선택하여 입력을 할때는 테이블의 컬럼의 순서와 상관없습니다.

|  |
| --- |
| insert into members (LastName,Country) values ('jiny','korea'); |

선언한 순서대로 1:1로 매칭이 되어 데이터값이 입력됩니다.

|  |
| --- |
| mysql> **insert into members (LastName,Country) values ('jiny','korea');**  Query OK, 1 row affected (0.00 sec) |

정상적으로 데이터 입력 쿼리가 실행되었습니다. Select 명령을 통하여 데이터를 확인해 봅니다.

|  |
| --- |
| mysql> **select \* from members;**  +----+----------+-----------+----------+-------+---------+  | Id | LastName | FirstName | Address | City | Country |  +----+----------+-----------+----------+-------+---------+  | 1 | hojin | lee | shinchon | seoul | korea |  | 2 | jiny | NULL | NULL | NULL | korea |  +----+----------+-----------+----------+-------+---------+  2 rows in set (0.00 sec) |

우리는 테이블의 모든 컬럼에 데이터를 저장하지 않았습니다. 결과를 보면 입력한 값 이외의 필드에는 NULL 로 채워집니다.

**09.2.2 PHP 실습**

====================

PHP코드를 통하여 데이터를 삽입해 보도록 합니다. 관련 메소드를 생성하여 소스에 적용을 합니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| // array 배열기반 insert 쿼리를 생성합니다.  public function arrQueryInsert($tbname, $arr)  {  $arrKey = array\_keys($arr);  For ($i=0;$i<count($arr);$i++) {  // $key = $arr[$i];  echo $arrKey[$i]."<br>";  $key = $arrKey[$i];  $insert\_filed .= "`".$key."`,";  $insert\_value .= "'".$arr[$key]."',";  }  $queryString = "INSERT INTO ".$tbname." ($insert\_filed) VALUES ($insert\_value);";  $queryString = str\_replace(",)",")",$queryString);  return $queryString;  }  // 어레이데이터를 insert 쿼리로 삽입합니다.  public function arrInsert($tbname, $arr)  {  if ($tbname) {  $queryString = $this->arrQueryInsert($tbname, $arr);  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 데이터 삽입!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 테이블 이름이 없습니다.");  }  } |

**예제파일) insert-01.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 어레이 배열의 키/값을 통하여 데이터를 삽입합니다.  $tbname = "members";  $arr = array('FirstName' => "jiny", 'LastName' => "1234");  $db->arrInsert($tbname,$arr);  ?> |

**출력화면)**

mysql connected!

FirstName

LastName

새로운 데이터를 삽입합니다.

쿼리성공] INSERT INTO members (`FirstName`,`LastName`) VALUES ('jiny','1234')

**09.3 다중삽입**

====================

기본적으로 INSERT 명령은 하나의 데이터 셋을 저장합니다. 하지만, 컬럼의 개수가 일정한 다수의 데이터는 다음과 같이 콤마로 구분하여 복수의 데이터를 입력을 할 수 있습니다.

**쿼리문법)**

|  |
| --- |
| INSERT INTO 테이블명 VALUES (value1, value2, value3, ...),  (value1, value2, value3, ...), (value1, value2, value3, ...); |

위처럼 기존 INSERT INTO 문구에서 데이터 블럭을 콤마(,)로 구분하여 여러 데이터를 한번에 입력할 수 있는 문법을 만들 수 있습니다.

**09.3.1 쿼리 실습**

====================

두개의 데이터를 하나의 INSERT 쿼리를 생성하여 저장을 합니다.

|  |
| --- |
| insert into members values ('','james','kim','youngdun-po','seoul','korea')**,** ('','eric','jung','seocho-gu','seoul','korea'); |

두개의 데이터는 괄호 + 콤마 + 괄호 형태로 구분되어 집니다.

|  |
| --- |
| mysql> **insert into members values ('','james','kim','youngdun-po','seoul','korea'), ('','eric','jung','seocho-gu','seoul','korea');**  Query OK, 2 rows affected, 2 warnings (0.00 sec)  Records: 2 Duplicates: 0 Warnings: 2 |

쿼리가 정상적으로 수행이 되었습니다. 또한, 2개의 데이터가 저장이 되었다고 결과도 같이 출력을 합니다. SELECT 명령을 통하여 입력한 데이터를 확인해 봅니다.

|  |
| --- |
| mysql> **select \* from members;**  +----+----------+-----------+-------------+-------+---------+  | Id | LastName | FirstName | Address | City | Country |  +----+----------+-----------+-------------+-------+---------+  | 1 | hojin | lee | shinchon | seoul | korea |  | 2 | jiny | NULL | NULL | NULL | korea |  | 3 | james | kim | youngdun-po | seoul | korea |  | 4 | eric | jung | seocho-gu | seoul | korea |  +----+----------+-----------+-------------+-------+---------+  4 rows in set (0.00 sec) |

**09.4 최신 Id 확인**

====================

테이블에는 기본적으로 각각의 열을 구분하는 프라이머리 키 컬럼이 있습니다. 또한 이 키값은 자동증가 제약사항 속성이 설정되어 있습니다. 자동 증가되는 프라이머리 키는 새로운 데이터를 삽입을 할 때마다 +1씩 증가가 됩니다. 또한 새로운 데이터를 저장할 때 프라이머리 키 값은 자동 증가 되기 때문에 생략하고 저장을 합니다.

이렇게 생략된 프라이머리 키값을 새로운 데이터 저장후에 확인을 할 수 있습니다. 최신 증가된 프라이머리 키값을 확인하는 이유는 데이터 삽입과 동시에 삽입한 데이터를 수정, 다른 테이블에 연결하는 등의 작업이 필요한 경우 입니다.

**09.4.1 PHP 실습**

====================

새롭게 추가된 데이터의 최신 프라이머리 아이디는 PHP의 제공된 함수를 통하여 확인을 할 수 있습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| // 마지막 삽입된 Id 값 확인  public function insertId(){  return mysqli\_insert\_id($this->dbcon);  } |

**예제파일) insert-02.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  // 어레이 배열의 키/값을 통하여 데이터를 삽입합니다.  $tbname = "members";  $arr = array('FirstName' => "jiny", 'LastName' => "123400");  $db->arrInsert($tbname,$arr);  echo "마지막 Id = ".$db->insertID();  ?> |

출력화면)

mysql connected!

FirstName

LastName

쿼리성공] INSERT INTO members (`FirstName`,`LastName`) VALUES ('jiny','123400');

데이터 삽입!

마지막 Id = 7

**09.5 Prepared**

====================

웹과 같이 외부와 공개된 상태에서 데이터 값을 입력 받아 삽입 쿼리를 생성을 하는 것은 다소 위험할 수 있습니다. 악의적인 사용자는 데이터의 입력에 SQL쿼리 오류를 발생할 수 있는 코드를 삽입을 하여 해킹을 시도할 수 있습니다.

MYSQLi는 이러한 안전한 쿼리를 생성할 수 있도록 Prepared 라는 SQL 인젝션 공격을 대비할 수 있는 매소드를 제공합니다. prepared 메소드 기능을 사용하기 위해서는 쿼리의 문장을 생성하는prepared 와 테이터를 연결하는 Bound 매개변수로 구성됩니다.

또한 한번 설정한 prepared 기능은 유사한 SQL문을 반복적으로 재사용할 수 있습니다. SQL쿼리의 오류 및 코드를 효율적으로 작성할 수 있습니다.

**예제파일) prepared.php**

|  |
| --- |
| <?php  $servername = "localhost";  $username = "username";  $password = "password";  $dbname = "myDB";  // Create connection  $conn = new mysqli($servername, $username, $password, $dbname);  // Check connection  if ($conn->connect\_error) {  die("Connection failed: " . $conn->connect\_error);  }  // prepare and bind  $stmt = $conn->**prepare**("INSERT INTO members (firstname, lastname, email) VALUES **(?, ?, ?)**");  $stmt->bind\_param("sss", $firstname, $lastname, $email);  // set parameters and execute  // 데이터1  $firstname = "jiny";  $lastname = "lee";  $email = "jiny@jinyphp.com";  $stmt->execute();  // 데이터2  $firstname = "hojin";  $lastname = "lee";  $email = "hojin@jinyphp.com";  $stmt->execute();  $stmt->close();  $conn->close();  ?> |

위의 예제에서 데이터 삽입을 위한 기본 쿼리를 생성합니다. 쿼리는 다음과 같습니다.

**"INSERT INTO members (firstname, lastname, email) VALUES (?, ?, ?)"**

쿼리에는 이상한 물음표(?) 기호가 들어가 있습니다. 물음표(?) 기호는 인젝션 공격을 대비하고 실제적인 데이터가 들어가는 자리입니다. 물음표는 정수, 문자열 등 다양한 데이터로 대체될 수 있습니다.

prepare를 통하여 쿼리설정이 되었다면 두번째로 실제적인 데이터가 들어가는 바인딩 설정을 합니다. bind\_param() 함수는 다음과 같이 설정됩니다.

**$stmt->bind\_param("sss", $firstname, $lastname, $email);**

첫번째 인자로는 바인딩 데이터의 데이터 타입을 설정합니다. 그 뒤에 바인딩 되는 데이터들을 매개변수 인자로 전달합니다.

바인딩함수의 데이터 타입은 약자를 이용하여 사용합니다. 인수 “sss” 는 바인딩 되는 데이터의 타입을 지정하는 것입니다. 위의 3가지 데이터 모두 스트링 이기 때문에 sss 로 지정합니다. 바인딩 데이터 타입은 4가지가 있습니다.

* i - integer
* d - double
* s - string
* b - BLOB

**09.6 우선표현**

====================

컬럼 선택적인 INSERT 쿼리문을 작성하여 데이터를 삽입할 때, 컬럼명과 데이터는 1:1로 매칭이 되어 집니다. 이때 매칭된 컬럼값을 이용하여 데이터 입력으로 재사용 할 수 있습니다.

다음은 2개의 값을 저장하는 INSERT 문입니다.

|  |
| --- |
| mysql> insert into board (level,pos) values (1,level+1);  Query OK, 1 row affected (0.01 sec) |

위의 예에서는 level 값에 1을 저장하고, post 값은 앞전에 설정한 level 값에 +1 을 더해서 pos 값을 입력하라는 표현 입니다.

Select를 통하여 입력된 데이터를 확인해 봅니다.

|  |
| --- |
| mysql> select \* from board;  +----+---------+-------+-------+------+  | Id | regdate | title | level | pos |  +----+---------+-------+-------+------+  | 3 | NULL | NULL | 1 | 2 |  +----+---------+-------+-------+------+  1 row in set (0.00 sec) |

level에는 1 값이 저장되고, pos에는 2값이 저장이 되었습니다. 이렇게 입력한 값들의 서로 연관성이 있는 경우 기존 매칭된 값을 이용하여 새로운 값을 삽입을 할 수 있습니다.

**09.7 처리옵션**

====================

새로운 데이터를 삽입할 때 동작을 지정할 수 있는 몇가지의 옵션을 추가로 지원합니다.

**09.7.1 LOW\_PRIORITY**

====================

삽입 우선 순위를 변경하여 처리합니다. 만일 누군가가 데이터를 읽기 작업을 하고 있으면, 해당 작업들이 모두 끝날때 까지 삽입을 지연합니다.

콘솔 실습)

|  |
| --- |
| mysql> insert LOW\_PRIORITY board (level,pos) values (1,level+1);  Query OK, 1 row affected (0.00 sec) |

우선순위를 낮추어 작업을 하는 것은, 수많은 사용자가 동시에 다수의 INSERT 작업이 이루어 질때 INSERT 작업들을 모아서 처리하는 효과를 가지고 올 수 있습니다.

한꺼번에 모아서 처리를 하는 것은, 빠른 삽입속도와 성능 향상을 기대할 수 있습니다.

**09.7.2 GNORE**

====================

데이터 중복시 무시합니다. 이 옵션은 다수의 데이터를 테이블 삽입시 발생될 수 있는 오류들을 예방할 수 있습니다. 예로 unique 속성을 가지는 프라이머리 키는 중복 입력이 되지 않기 때문에 MYSQL은 삽입 실행도중에 에러를 출력하게 되고 중단됩니다.

하지만 IGNORE 옵션을 사용을 하게 되면 해당 데이터는 무시하고, 다음 데이터로 넘어가서 처리를 계속하게 됩니다.

콘솔 실습)

|  |
| --- |
| mysql> insert IGNORE board (level,pos) values (1,level+1);  Query OK, 1 row affected (0.01 sec) |

**09.8 REPLACE**

====================

INSERT 쿼리를 생성을 할 때 중복을 허용하지 않는 프라이머리 키나 인덱스는 실행되지 않습니다. 하지만 REPLACE 명령문은 중복된 인덱스 컬럼이 있을 경우 기존 데이터를 삭제하고 새로운 데이터로 교체를 하게 됩니다.

먼저 테이블의 데이터 내용을 확인해 봅니다.

|  |
| --- |
| mysql> select \* from board;  +----+---------+-------+-------+------+  | Id | regdate | title | level | pos |  +----+---------+-------+-------+------+  | 2 | NULL | NULL | 2 | 5 |  | 3 | NULL | NULL | 1 | 2 |  +----+---------+-------+-------+------+  2 rows in set (0.00 sec) |

Id가 2로 중복된 데이터를 REPLACE로 삽입을 해봅니다.

|  |
| --- |
| mysql> REPLACE INTO board (Id, level,pos) values (2,21,53);  Query OK, 2 rows affected (0.01 sec) |

오류가 나지 않고 정상적으로 수행이 되었습니다. Selelct 명령을 통하여 다시한번 내용을 확인해 봅니다.

|  |
| --- |
| mysql> select \* from board;  +----+---------+-------+-------+------+  | Id | regdate | title | level | pos |  +----+---------+-------+-------+------+  | 2 | NULL | NULL | 21 | 53 |  | 3 | NULL | NULL | 1 | 2 |  +----+---------+-------+-------+------+  2 rows in set (0.00 sec) |

위의 실습에서는 인덱스가 2인 동인한 컬럼을 REPLACE 를 통하여 기존에 데이터가 삭제되고 새로운 데이터로 변경되는 것을 확인 할 수 있습니다.