jinyPHP 2권 - 022. 프로시저

**22. 프로시저**

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

프로시저를 간략하게 말하면 SQL 명령의 묽음 입니다. 지금까지의 SQL 쿼리 명령은 한 개의 문장 단위로 실행을 하였습니다. 프로시저(Procedure)의 의미는 “절차”라는 뜻으로 여러 개의 작업 SQL 쿼리를 묽어서 하나의 명령처럼 처리 할 수 있는 기능입니다.

프로시저를 사용하는 목적은 여러가지의 SQL 명령 작업들을 묽어서 처리 할 수 있다는 점 입니다. SQL 명령들의 작업들이 서로 연관성이 있는 경우에 많이 사용을 합니다. 명령들의 묽음을 실행할 때 중간의 어느 하나 쿼리에 문제가 발생되면 해당 묽음의 쿼리는 의미가 없습니다.

또한, 묽음 쿼리를 진행을 하면서 처음부터 중간 오류가 발생된 시점에는 이미 데이터의 변경처리가 되었기 때문에 묽음을 다시 실행하기 위해서는 이전의 작업들을 다시 되돌리기 처리를 해야 합니다. 이런경우, 프로시저는 매우 편리하게 작업을 할 수 있습니다.

프로시저로 묶어진 질의는 한번의 쿼리로 해석 처리되기 때문에 데이터 처리 속도가 향상되는 효과를 얻을 수도 있습니다. 프로시저는 MYSQL 5.0 버전이상에서 지원하는 기능입니다.

**22.1 프로시저 생성**

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

프로시저 생성은 CREATE PROCEDURE 명령 키워드를 사용하여 생성을 합니다. 묶여지는 쿼리는 BEGIN ~ END 사이에 정의를 하면 됩니다.

**쿼리문법)**

|  |
| --- |
| CREATE PROCEDURE 프로시저이름()  BEGIN  SQL 쿼리1;  SQL 쿼리1;  SQL 쿼리3;  END |

프로시저를 한번 생성을 하게 되면 데이터베이스 프로시저 목록에 저장이 됩니다. 만일, 동일한 프로시저명을 생성 하면 중복이름 오류가 발생됩니다.

**22.2 DELIMITER**

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

SQL의 명령문은 마지막에 세미콜론(;)으로 끝납니다. 세미콜론은 하나의 쿼리명령을 완성하는 구분기호 입니다. 세미콜론으로 완성된 쿼리는 바로 실행이 됩니다.

세미콜론 구문기호는 프로시저를 작성할 때 잘못된 구분의 종료로 인하여 오류가 발생이 됩니다. 프로시저 안에 포함되는 SQL 쿼리문장이 세미콜론을 포함하고 있기 때문에 정확한 프로시저의 문장을 생성할 수 없습니다.

이런 경우 임시로 정의되어 있는 세미콜론(;) 구분기호를 다른 기호로 변경해 두면 구문기호 충돌로 인한 오류 동작을 예방할 수 있습니다.

DELIMITER 명령을 사용하면 현자의 사용 구분자인 세미콜론(;)을 임시로 다른 값으로 변경해 놓을 수 있습니다.

**쿼리문법)**

|  |
| --- |
| DELIMITER // |

위의 쿼리 문법은 현재의 구문기호 세미콜론(;)을 // 기호로 임시 변경을 합니다. 명령을 입력할 때는 DELIMITER 와 // 사이에 공백이 하나 있어야 합니다.

다시 원래의 세미콜론으로 복원을 한때는 다음과 같이 작성을 하면 됩니다.

|  |
| --- |
| DELIMITER ; |

임시로 변경해 놓은 구분기호 // 를 다시 세미콜론으로 변경합니다. 명령을 입력할 때는 DELIMITER 와 세미콜론(;) 사이에 공백이 하나 있어야 합니다.

**22.2.1 쿼리 실습**

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

구분기호 변경과 간단한 프로시저 문장을 작성해 보도록 하겠습니다.

delimiter //

create procedure mem1()

begin

select \* from members;

end

//

delimiter ;

먼저 delimiter // 를 통하여 구분기호 세미콜론을 //로 변경을 합니다. 프로시저 구분을 작성합니다. 프로시저 안에서 select \* from members; 와 같이 세미콜론을 포함하여도 오류가 발생되지 않습니다.

|  |
| --- |
| mysql> delimiter //  mysql> create procedure mem1()  -> begin  -> select \* from members;  -> end  -> //  Query OK, 0 rows affected (0.00 sec)  mysql> delimiter ; |

마지막에 프로시저의 마감 구분기호 // 를 작성합니다. 원래는 구분기호 세미콜론을 작성을 해야 하지만, 현재 구분기호를 //로 변경을 해주었기 때문입니다. 다시 구분기호를 세미콜론으로 복원해 놓습니다.

**22.2.2 PHP 실습**

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

PHP 코드를 통하여 구분기호를 변경하는 명령 메서드를 생성해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function delimiter($code)  {  if ($code) {  $queryString = "DELIMITER $code";  // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" DELIMITER 변경!");  // 객체 반환, 매서드체인  return $this;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] delimiter 키명을 적어주세요.");  }  } |

**22.3 프로시저 호출**

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

프로시저를 작성하게 되면 데이터베이스의 프로시저 목록에 등록이 됩니다. 등록된 프로시저는 호출하여 실행을 할 수 있습니다. CALL 명령은 등록된 프로시저를 실행을 합니다.

**쿼리문법)**

|  |
| --- |
| CALL 프로시저이름; |

**22.3.1 쿼리 실습**

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

앞에서 실습으로 생성해 놓은 프로시저를 다시 호출하여 실행해 보도록 하겠습니다. 프로시저 호출 명령 call 다음에 프로시저 이름을 입력하면 됩니다.

**call mem1;**

|  |
| --- |
| mysql> call mem1;  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  | Id | LastName | FirstName | Address | City | Country | manager | email |  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  | 1 | hojin | lee | shinchon | seoul | korea | infohojin | hojin@jinyphp.com |  | 2 | jiny | NULL | NULL | NULL | korea | infohojin | jiny@jinyphp.com |  | 3 | james | kim | youngdun-po | seoul | korea | info | james@jinyphp.com |  | 4 | eric | jung | seocho-gu | busan | korea | info | eric@jinyphp.com |  +----+----------+-----------+-------------+-------+---------+-----------+-------------------+  4 rows in set (0.00 sec)  Query OK, 0 rows affected (0.03 sec) |

예제로 작성한 mem1 프로시저에는 회원 목록을 출력하는 쿼리명령 하나 밖에 없습니다. 따라서 프로시저를 호출하면 회원 목록을 출력합니다.

위의 쿼리 실습에서는 프로시저 이름만 지정하여 프로시저를 호출 하였습니다. 또는 프로시저이름 다음에 소괄호를 추가하여 호출을 할 수도 있습니다.

**call mem1();**

**22.4 프로시저 매개변수**

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

프로시저를 호출할 때 단순하게 프로시저의 이름만 지정하는 방법과 이름 뒤에 소괄호를 붙여서 실행하는 방법 2가지를 배웠습니다.

프로시저 이름 뒤에 소괄호를 붙여서 호출을 한다는 것은 소괄호 안에 매개변수 인자와 같은 값을 전달 할 수 있다는 확장된 의미일 것입니다. 프로시저는 마치 함수와 유사하기도 합니다. 차이점이라면은 반환 값이 없다는 것입니다.

**22.4.1 매개변수 선언**

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

프로시저는 호출할 때 매개변수를 전달하여 실행을 할 수 있습니다. 매개변수는 프로시저 작성시에 이름명 뒤에 있는 소괄호 안에 작성을 하면 됩니다.

**쿼리문법)**

|  |
| --- |
| CREATE PROCEDURE 프로시저이름( 변수명 데이터타입 ) |

프로시저의 매개변수 선언은 **변수명 + 데이터타입** 형태로 선언합니다. 프로시저의 데이터 타입은 변수명 뒤에 선언하는 것이 기존 PHP함수와 약간의 차이점이 있습니다.

작성예제)

CREATE PROCEDURE mem(**a INT**)

BEGIN

SELECT \* FROM members WHERE Id=a;

END

**22.4.2 인자전달**

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

매개변수를 포함한 프로시저를 실행할 수 있습니다. 매개변수를 전달할 때는 프로시저에서 지정한 데이터의 타입과 동일한 형태의 값을 사용해야 합니다.

**쿼리문법)**

|  |
| --- |
| CALL 프로시저(값); |

매개변수를 포함하고 있는 프로시저는 인자 값을 넣어서 호출합니다. 인자 값은 프로시저명() 의 소괄호 안에 넣어 주시면 됩니다.

**22.4.3 쿼리 실습**

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

매개변수를 포함하는 프로시저를 생성해 보고 실행하 보도록 합니다.

|  |
| --- |
| // 프로시저를 호출합니다.  mysql> DELIMITER //  mysql> CREATE PROCEDURE mem(a INT)  -> BEGIN  -> SELECT \* FROM members WHERE Id=a;  -> END  -> //  Query OK, 0 rows affected (0.00 sec)  mysql> DELIMITER ; |

매개변수로는 회원의 Id 값을 전달 받습니다. 회원 아이디를 이용하여 회원의 정보를 출력합니다.

|  |
| --- |
| // 프로시저를 호출합니다.  mysql> CALL mem(1);  +----+----------+-----------+----------+-------+---------+-----------+-------------------+  | Id | LastName | FirstName | Address | City | Country | manager | email |  +----+----------+-----------+----------+-------+---------+-----------+-------------------+  | 1 | hojin | lee | shinchon | seoul | korea | infohojin | hojin@jinyphp.com |  +----+----------+-----------+----------+-------+---------+-----------+-------------------+  1 row in set (0.00 sec)  Query OK, 0 rows affected (0.01 sec) |

매개변수를 통하여 회원 목록을 출력을 하였습니다.

**22.4.4 PHP 실습**

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

PHP 코드를 통하여 프로시저 호출을 실습해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function callProcedure($name,$args)  {  if ($name) {  $queryString = "CALL $name($args);";  if($result = mysqli\_query($this->dbcon, $queryString)) {  $this->msgEcho("쿼리성공] ".$queryString);  $rowss = "";  $row\_cnt = mysqli\_num\_rows($result);  for ($i=0; $i<$row\_cnt; $i++) {  $rowss[$i] = mysqli\_fetch\_object($result);  }  $result->free();  return $rowss;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 프로시저명을 적어주세요.");  }  } |

**예제파일) pro-02.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  $name = "mem";  if($rowss = $db->callProcedure($name,1)){  echo "tables fields = ". count($rowss) . "<br>";    for($i=0;$i<count($rowss);$i++){  echo $i."=";  print\_r($rowss[$i]);  echo "<br>";  }    }    ?> |

출력화면)

mysql connected!

쿼리성공] CALL mem(1);

tables fields = 1

0=stdClass Object ( [Id] => 1 [LastName] => hojin [FirstName] => lee [Address] => shinchon [City] => seoul [Country] => Korea [manager] => infohojin [email] => hojin@jinyphp.com )

**22.5 프로시저 확인**

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

작성한 프로시저들은 데이터베이스의 프로시저 목록에 등록이 됩니다. 등록된 프로시저의 내용을 출력하여 확인할 수 있습니다. 프로시저의 내용을 출력할 때는 SHOW CREATE PROCEDURE 명령을 사용합니다.

**쿼리문법)**

|  |
| --- |
| SHOW CREATE PROCEDURE 프로시저명; |

**22.5.1 쿼리 실습**

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

데이터베이스의 프로시저 내용을 출력 확인해 보겠습니다.

**show create procedure mem;**

|  |
| --- |
| mysql> show create procedure mem;  +-----------+------------------------+------------------------------------------------------------------------------------------------------------+----------------------+----------------------+--------------------+  | Procedure | sql\_mode | Create Procedure | character\_set\_client | collation\_connection | Database Collation |  +-----------+------------------------+------------------------------------------------------------------------------------------------------------+----------------------+----------------------+--------------------+  | mem | NO\_ENGINE\_SUBSTITUTION | CREATE DEFINER=`root`@`localhost` PROCEDURE `mem`()  BEGIN  select \* from members;  select \* from orders;  END | utf8 | utf8\_general\_ci | utf8\_general\_ci |  +-----------+------------------------+------------------------------------------------------------------------------------------------------------+----------------------+----------------------+--------------------+  1 row in set (0.00 sec) |

작성한 프로시저의 내용이 출력됩니다.

**22.5.2 PHP 실습**

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

PHP 코드를 통하여 프로시저를 확인해 보도록 하겠습니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| public function isProcedure($name)  {  if ($name) {  $queryString = "SHOW CREATE PROCEDURE $name";    if($result = mysqli\_query($this->dbcon, $queryString)) {  $this->msgEcho("쿼리성공] ".$queryString);  $rowss = "";  $row\_cnt = mysqli\_num\_rows($result);  for ($i=0; $i<$row\_cnt; $i++) {  $rowss[$i] = mysqli\_fetch\_object($result);  }  $result->free();  return $rowss;  } else {  $this->msgEcho("Error] ".$queryString);  }  } else {  $this->msgEcho("Error] 프로시저명을 적어주세요.");  }  } |

**예제파일) pro-04.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  $name = "mem";    if ($rowss = $db->isProcedure($name)) {  echo "tables fields = ". count($rowss) . "<br>";    for ($i=0;$i<count($rowss);$i++) {  echo $i."=";  print\_r($rowss[$i]);  echo "<br>";  }  }    ?> |

출력화면)

mysql connected!

쿼리성공] SHOW CREATE PROCEDURE mem

tables fields = 1

0=stdClass Object ( [Procedure] => mem [sql\_mode] => NO\_ENGINE\_SUBSTITUTION [Create Procedure] => CREATE DEFINER=`root`@`localhost` PROCEDURE `mem`(a INT) BEGIN SELECT \* FROM members WHERE Id=a; END [character\_set\_client] => utf8 [collation\_connection] => utf8\_general\_ci [Database Collation] => utf8\_general\_ci )

**22.6 프로시저 삭제**

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

작성한 프로시저를 삭제할 수 있습니다. 프로시저 삭제를 할 때에도 기존에 익숙한 DROP 명령어를 이용하여 삭제를 할 수 있습니다.

**쿼리문법)**

|  |
| --- |
| DROP PROCEDURE 프로시저명; |

**22.6.1 쿼리 실습**

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

기존에 실습하던 프로시저를 삭제해 보도록 하겠습니다.

**drop procedure mem1;**

|  |
| --- |
| mysql> drop procedure mem1;  Query OK, 0 rows affected (0.00 sec) |

정상적으로 프로시저가 삭제가 되었습니다.

**22.6.2 PHP 실습**

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

PHP 코드를 통하여 작성한 프로시저를 삭제해 봅니다.

**PHP 예제)**

mysql.class.php 파일에 메서드 예제를 추가합니다.

|  |
| --- |
| // 생성한 프로시저를 삭제합니다.  public function dropProcedure($name)  {  if ($name) {  $queryString = "DROP PROCEDURE $name";    // 쿼리를 전송합니다.  if (mysqli\_query($this->dbcon, $queryString)=== TRUE) {  $this->msgEcho("쿼리성공] ".$queryString);  $this->msgEcho(" 프로시저 삭제!");  return true;  } else {  $this->msgEcho("Error] ".$queryString);  return false;  }  } else {  $this->msgEcho("Error] 프로시저명을 적어주세요.");  }  } |

**예제파일) pro-05.php**

|  |
| --- |
| <?php  include "dbinfo.php";  include "mysql.class.php";    // ++ Mysqli DB 연결.  $db = new JinyMysql();  $name = "mem";  if ($db->dropProcedure($name)) {  echo $name." 프로시저 삭제성공! <br>";  if ($rowss = $db->isProcedure($name)) {  echo "tables fields = ". count($rowss) . "<br>";    for ($i=0;$i<count($rowss);$i++) {  echo $i."=";  print\_r($rowss[$i]);  echo "<br>";  }  } else {  echo $name." 프로시저를 찾을 수 없습니다.";  }    } else {  echo "프로시저를 삭제 할 수 없습니다.<br>";  }    ?> |

출력화면)

mysql connected!

쿼리성공] DROP PROCEDURE mem

프로시저 삭제!

mem 프로시저 삭제성공!

Error] SHOW CREATE PROCEDURE mem

mem 프로시저를 찾을 수 없습니다.