# Phần 1 : Giới thiệu :

## Các thành phần của Zend\_db :

* Zend\_Db\_Adapter
* Zend\_Db\_Exception
* Zend\_Db\_Expr
* Zend\_Db\_Statement
* Zend\_Db\_Profiler
* Zend\_Db\_Select
* Zend\_Db\_Table

## Chi tiết:

### Các thành phần quan trọng :

#### Zend\_Db\_Adapter

Dùng để tạo kết nối giữa ứng dụng php và máy chủ chứa Database.

#### Zend\_Db\_Select

Đại diện cho một câu lệnh SQL truy vấn SELECT. Class có phương thức để thêm bộ phận riêng lẻ để truy vấn. Có thể xác định các bộ phận truy vấn thông qua phương thức PHP kết hợp với cấu trúc hình thức SQL. Sau khi xây dựng một truy vấn, có thể thực hiện truy vấn như thể viết 1 chuỗi truy vấn.

#### Zend\_Db\_Table

 Là một giao diện hướng đối tượng vào các bảng cơ sở dữ liệu. Nó cung cấp phương pháp cho các hoạt động phổ biến trên bảng. Các lớp cơ sở được mở rộng, vì vậy có thể thêm logic tùy chỉnh.

#### Zend\_Db\_Statement

Sử dụng để tạo ra nhiều lựa chọn cách trình bày câu truy vấn và lấy kết quả. Nó mô tả làm sao để thể hiện một đối tượng Statement, và làm sao để sử dụng chúng.

### Các thành phần không quan trọng :

#### Zend\_Db\_Profiler

Có nhiệm vụ kiểm tra các câu truy vấn:

* Kiểm tra tổng số các truy vấn đã được Profiler.
* Trả về tổng số giây trôi qua cho tất cả các truy vấn profiled .
* Trả về một mảng của tất cả các câu lệnh truy vấn.
* Trả về câu lệnh truy vấn cuối cùng (gần đây nhất), bất kể có hay không truy vấn đã hoàn thành (nếu nó đã không hoàn thành, thời gian kết thúc sẽ được NULL ).
* Xóa các kiểm tra câu lệnh.

Ngoài ra còn thực hiện nhiệm vụ lọc các câu truy vấn trước khi thực hiện trong Database:

* Lọc theo thời gian truy vấn.
* Lọc theo loại truy vấn.
* Khôi phục các Profiler theo loại truy vấn.

#### Zend\_Db\_Exception

Lớp này có nhiệm vụ bắt biệt lệ khi kết nối, sử dụng Database.

#### Zend\_Db\_Expr

Lớp này chỉ đơn giản là tổ chức một chuỗi, do đó các mảnh vỡ của các câu lệnh SQL có thể được phân biệt với các định dạng và giá trị cần được trích dẫn ngầm khi nội suy vào các câu lệnh SQL.

Ví dụ, khi xác định một giá trị khóa chính khi chèn vào một hàng mới, một số thương hiệu RDBMS có thể yêu cầu bạn phải sử dụng một biểu thức để tạo ra giá trị mới của một chuỗi. Nếu biểu hiện này được coi là một định danh, nó sẽ được trích dẫn và biểu thức sẽ không được đánh giá. Một ví dụ khác là bạn có thể sử dụng Zend\_Db\_Expr trong Zend\_Db\_Select:: thứ tự () để đặt hàng bằng sự biểu hiện thay vì chỉ đơn giản là một tên cột.

Nó hoạt động như rằng trong mỗi ngữ cảnh trong đó một tên cột có thể được quy định cụ thể các phương pháp của các lớp học Zend\_Db, nếu giá trị là một thể hiện của Zend\_Db\_Expr thay vì một chuỗi đồng bằng, sau đó biểu thức là không trích dẫn. Nếu nó là một chuỗi đồng bằng, nó được giả định là một tên cột đơn giản.

## Cách sử dụng các thành phần quan trọng trong zend:

### Zend\_Db\_Adapter

#### Kết nối :

Các thuộc tính cần biết :

* host: chuỗi có tên máy chủ hoặc địa chỉ IP máy chủ chứa Database : 'localhost' hoặc '127.0.0.1'.
* username: tên đăng nhập trên máy chủ chứa Database.
* password: Mật khẩu tài khoản đăng nhập cào máy chủ chứa Database.
* dbname: Tên của Database trên máy chủ
* port: một số máy chủ chứa Database chấp nhận kết nối qua các cổng xác định, tham số này để xác định số cổng mà ta sẽ sử dụng kết nối.
* charset: định dạng các ký tự được sử dụng trong kết nối Database
* options: cung cấp 1 mảng các tùy chọn ta thiết lập cho class Zend\_Db\_Adapter
* driver\_options: cung cấp 1 mảng các tùy chọn dùng để thiết lập các tham số cho việc điều khiển PDO.
* adapterNamespace: phần tên ban đầu của tên lớp kết nối thay vì sử dụng ‘Zend\_Db\_Adapter’. Sử dụng khi cần sử dụng phương thức factory() để load 1 lớp không phải là kết nối Zend database
* Tạo kết nối bằng cấu trúc Zend\_Db\_Adapter :

$db = **new** Zend\_Db\_Adapter\_Pdo\_Mysql([array](http://www.php.net/array)(

    'host'     => '127.0.0.1',

    'username' => 'webuser',

    'password' => 'xxxxxxxx',

    'dbname'   => 'test'

));

* Tạo kết nối với Zend\_Db Factory : Phương pháp này tự động tải các tập tin lớp học bộ chuyển đổi trên nhu cầu sử dụng [Zend\_Loader:: loadClass ()](http://translate.googleusercontent.com/translate_c?hl=vi&prev=/search%3Fq%3Dzend_db%26hl%3Dvi%26prmd%3Dimvns&rurl=translate.google.com.vn&sl=en&twu=1&u=http://framework.zend.com/manual/en/zend.loader.load.html&usg=ALkJrhiTQeyizNBZ2J934d5zCGIeVEbAnw#zend.loader.load.class) .

$db = Zend\_Db::factory('Pdo\_Mysql', [array](http://www.php.net/array)(

    'host'     => '127.0.0.1',

    'username' => 'webuser',

    'password' => 'xxxxxxxx',

    'dbname'   => 'test'

));

* Tạo kết nối sử dụng Zend\_Config với Zend\_Db factory :

$config = **new** Zend\_Config(

[array](http://www.php.net/array)(

        'database' => [array](http://www.php.net/array)(

            'adapter' => 'Mysqli',

            'params'  => [array](http://www.php.net/array)(

                'host'     => '127.0.0.1',

                'dbname'   => 'test',

                'username' => 'webuser',

                'password' => 'secret',

            )

        )

    )

);

$db = Zend\_Db::factory($config->database);

* Kiểm tra kết nối :

try {

    $db = Zend\_Db::factory('Pdo\_Mysql', $parameters);

    $db->getConnection();

} catch (Zend\_Db\_Adapter\_Exception $e) {

*//khi có lỗi từ kết nối với máy chủ*

} catch (Zend\_Exception $e) {

*//khi có lỗi từ kết nối từ factory*

}

#### Các câu lệnh:

* setFetchMode ($constant) :

FETCH\_OBJ Trả kết quả truy vấn thành các đối tượng  
FETCH\_ASSOC Trả kết quả truy vấn thành những mảng liên tục, giữ nguyên tên của field  
FETCH\_NUM Trả kết quả truy vấn thành mảng, tên field sẽ được chuyển thành số liên tục  
FETCH\_BOTH Trả kết quả truy vấn thành mảng, mảng bao gồm kết quả của 2 kiểu FETCH\_ASSOC và FETCH\_NUM

*Code:*

$db = Zend\_Db::factory('Pdo\_Mysql', array ('host' => 'localhost',   
                     'username' => 'root',   
                     'password' => '',   
                     'dbname' => 'company\_2'));  
$db->setFetchMode(Zend\_Db::FETCH\_OBJ);

* fetchAll($sql,$bind) :

Phương thức fetchAll là phương thức để lấy dữ liệu trong CSDL. Nó tương tự như câu lệnh Select của MySQL  
  
*Tham số:*  
$sql: Câu lệnh truy vấn  
$bind: Mảng giá trị truyền vào khi truy vấn

*Code :*

$sql = "SELECT member\_id, member\_name, user\_name    
        FROM members   
        WHERE group\_id = ?       
        AND member\_status = ?";  
$db->fetchAll($sql,array('g002',1));

*hay:*

$sql = "SELECT member\_id, member\_name, user\_name    
        FROM members   
        WHERE group\_id = 'g002'      
        AND member\_status = 1";  
$db->fetchAll($sql);

* fetchAssoc ($sql,$bind):

Phương thức fetchAssoc tương tự như fetchAll là phương thức để lấy dữ liệu trong CSDL nhưng nó sẽ trả về một mảng dữ liệu cho dù setFetchMode() ở chế độ nào đi nữa  
  
*Tham số:*$sql: Câu lệnh truy vấn  
$bind: Mảng giá trị truyền vào khi truy vấn

*Code :*

$db->setFetchMode(Zend\_Db::FETCH\_OBJ);  
$sql = "SELECT member\_id, member\_name, user\_name    
        FROM members   
        WHERE group\_id = ?       
        AND member\_status = ?";  
$db->fetchAssoc($sql,array('g002',1));

* fetchCol ($sql,$bind) :

Phương thức fetchCol là phương thức để lấy dữ liệu trong CSDL nhưng nó chỉ trả về giá trị của field đầu tiên trong kết quả truy vấn  
  
*Tham số:*  
$sql: Câu lệnh truy vấn  
$bind: Mảng giá trị truyền vào khi truy vấn

*Code :*

$sql = "SELECT \*     
        FROM members   
        WHERE group\_id = ?       
        AND member\_status = ?";  
$db-> fetchCol($sql,array('g002',1));

* fetchPairs($sql,$bind) :

Phương thức fetchPairs giống phương thức fetchCol để lấy dữ liệu trong CSDL nhưng nó chỉ trả về giá trị của field thứ 2 trong kết quả truy vấn  
  
*Tham số:*  
$sql: Câu lệnh truy vấn  
$bind: Mảng giá trị truyền vào khi truy vấn

*Code* :

$sql = "SELECT \*     
        FROM members   
        WHERE group\_id = ?       
        AND member\_status = ?";  
$db-> fetchPairs($sql,array('g002',1));

* fetchRow($sql,$bind) :

Phương thức fetchRow là phương thức để lấy dữ liệu trong CSDL nhưng nó chỉ trả về một dòng dữ liệu duy nhất  
  
*Tham số:*  
$sql: Câu lệnh truy vấn  
$bind: Mảng giá trị truyền vào khi truy vấn

*Code:*

$sql = "SELECT \*     
        FROM members   
        WHERE group\_id = ?       
        AND member\_status = ?";  
$db-> fetchRow($sql,array('g002',1));

* fetchOne ($sql,$bind) :

Phương thức fetchOne là phương thức để lấy dữ liệu trong CSDL nhưng nó chỉ trả về một giá trị duy nhất  
  
*Tham số:*  
$sql: Câu lệnh truy vấn  
$bind: Mảng giá trị truyền vào khi truy vấn

*Code:*

$sql = "SELECT \*     
        FROM members   
        WHERE group\_id = ?       
        AND member\_status = ?";  
$db-> fetchOne($sql,array('g002',1))

* insert($table, $bind) :

Phương thức insert là phương thức để đưa dữ liệu vào CSDL  
  
*Tham số:*  
$table: tên bảng   
$bind: Mảng giá trị đưa vào CSDL

*Code:*

$data = array(  
      'group\_id'      => 'g004',  
      'group\_name'    => 'Nghien cuu',  
      'group\_status'  => 1  
      );  
        
$db->insert('members', $data);

* lastInsertId ($table, $primaryKey) :

Phương thức lastInsertId là phương thức lấy ID của một record vừa được thêm vào bằng phương thức insert()  
  
*Tham số:*  
$table: tên bảng   
$primaryKey: Khóa chính của bảng

*Code:*

$data = array(  
               'member\_name'         => 'Pham Vu Khanh',  
               'user\_name'       => 'KhanhPham',  
               'email'        => 'vukhanh2212@gmail.com',  
               'password'        => md5('123456'),  
            'register\_date'    => date('Y-m-d 00:00:00'),  
            'member\_status'    => 1,  
            'group\_id'       => 'g001'  
              );        
     
$db->insert('members', $data);  
echo '<br>' . $id = $db->lastInsertId();

* update($table, $data,$where) :

Phương thức update là phương thức để cập nhật dữ liệu vào CSDL  
  
*Tham số:*

$table: tên bảng   
$data: Mảng giá trị đưa vào CSDL  
$where: điều kiện để cập nhật

*Code :*

$where = " group\_id = 'g004'";  
$data = array(                    
          'group\_name'    => 'Research',  
          'group\_status'  => 1  
      );  
        
$db->update('groups', $data, $where);

### Zend\_Db\_Statement

* So sánh với cách sử dụng phương thức adapter:
* Câu lệnh dùng phương thức của lớp adapter:

$stmt = $db->query(

            'SELECT \* FROM bugs WHERE reported\_by = ? AND bug\_status = ?',

[array](http://www.php.net/array)('goofy', 'FIXED')

        );

* Câu lệnh sử dụng Zend\_Db\_Statement :
* Sử dụng phương thức execute():

$sql = 'SELECT \* FROM bugs WHERE reported\_by = ? AND bug\_status = ?';

$stmt = **new** Zend\_Db\_Statement\_Mysqli($db, $sql);

$stmt->execute([array](http://www.php.net/array)('goofy', 'FIXED'));

* Sử dụng đặt tên tham số:

$sql = 'SELECT \* FROM bugs WHERE ' .

       'reported\_by = :reporter AND bug\_status = :status';

$stmt = **new** Zend\_Db\_Statement\_Mysqli($db, $sql);

$stmt->execute([array](http://www.php.net/array)(':reporter' => 'goofy', ':status' => 'FIXED'));

* Kết quả lấy từ 1 câu lệnh Select :
* Lấy 1 dòng từ 1 tập hợp kết quả :

$ Stmt = $ db -> query ("SELECT \* FROM lỗi ');

trong khi ($ row = $ stmt -> lấy ()) {

[echo](http://translate.googleusercontent.com/translate_c?hl=vi&prev=/search%3Fq%3Dzend_db%26hl%3Dvi%26prmd%3Dimvns&rurl=translate.google.com&sl=en&twu=1&u=http://www.php.net/echo&usg=ALkJrhiAA9K04ovu09nPcIxYzs3WRgZl9A) hàng $ ['bug\_description'];

}

* Lấy kết quả đầy đủ:

$ Stmt = $ db -> query ("SELECT \* FROM bugs ');

$rows  = $ stmt -> fetchAll ();

[echo](http://translate.googleusercontent.com/translate_c?hl=vi&prev=/search%3Fq%3Dzend_db%26hl%3Dvi%26prmd%3Dimvns&rurl=translate.google.com&sl=en&twu=1&u=http://www.php.net/echo&usg=ALkJrhiAA9K04ovu09nPcIxYzs3WRgZl9A) $rows [0] ['bug\_description'];

* Thay đổi các chế độ Fletch:

$stmt = $db->query('SELECT \* FROM bugs');

$stmt->setFetchMode(Zend\_Db::FETCH\_NUM);

$rows = $stmt->fetchAll();

[echo](http://www.php.net/echo) $rows[0][0];

* Lấy 1 cột từ 1 tập hợp kết quả :

$stmt = $db->query('SELECT bug\_id, bug\_description, bug\_status FROM bugs');

$bug\_status = $stmt->fetchColumn(2);

* Lấy 1 dòng như 1 đối tượng:

$stmt = $db->query('SELECT bug\_id, bug\_description, bug\_status FROM bugs');

$obj = $stmt->fetchObject();

[echo](http://www.php.net/echo) $obj->bug\_description;

### Zend\_Db\_Table

* Định nghĩa 1 lớp Table:
  + - Xác định tên Table và tên lược đồ

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bugs';

}

Nếu không đặt tên bảng ( bugs ) thì nó sẽ mặc định là tên của class

**class** bugs **extends** Zend\_Db\_Table\_Abstract

{

*// table name matches class name*

}

Khai báo các schema:

*// First alternative:*

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_schema = 'bug\_db';

    protected $\_name   = 'bugs';

}

*// Second alternative:*

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bug\_db.bugs';

}

*// If schemas are specified in both $\_name and $\_schema, the one*

*// specified in $\_name takes precedence:*

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name   = 'bug\_db.bugs';

    protected $\_schema = 'ignored';

}

*hoặc :*

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

}

*// First alternative:*

$tableBugs = **new** Bugs([array](http://www.php.net/array)('name' => 'bugs', 'schema' =>'bug\_db'));

*// Second alternative:*

$tableBugs = **new** Bugs([array](http://www.php.net/array)('name' => 'bug\_db.bugs'));

*// If schemas are specified in both 'name' and 'schema', the one*

*// specified in 'name' takes precedence:*

$tableBugs = **new** Bugs([array](http://www.php.net/array)('name' => 'bug\_db.bugs',

                            'schema' => 'ignored'));

* + - Xác định khóa chính của Table

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bugs';

    protected $\_primary = 'bug\_id';

}

* + - Cài đặt Table

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected **function** \_setupTableName()

    {

        $this->\_name = 'bugs';

        parent::\_setupTableName();

    }

}

Các phương thức :

**\_setupDatabaseAdapter()**

**\_setupTableName()**

**\_setupMetadata()**

**\_setupPrimaryKey()**

* + - Khởi tạo Table

Dùng phương thức *Init()* thay vì *\_construct()* khi không cần thay đổi dữ liệu trong chương trình

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_observer;

**public** **function** init()

    {

        $this->\_observer = **new** MyObserverClass();

    }

}

* Tạo phương thức cho Table
  + Tạo kết nối Database cho Table:

$db = Zend\_Db::factory('PDO\_MYSQL', $options);

$table = **new** Bugs([array](http://www.php.net/array)('db' => $db));

* + Thiết lập kết nối Database mặc định

$db = Zend\_Db::factory('PDO\_MYSQL', $options);

Zend\_Db\_Table\_Abstract::setDefaultAdapter($db);

*// Later...*

$table = **new** Bugs();

* + Lưu trữ kết nối Database trong Registry

$db = Zend\_Db::factory('PDO\_MYSQL', $options);

Zend\_Registry::set('my\_db', $db);

*// Later...*

$table = **new** Bugs([array](http://www.php.net/array)('db' => 'my\_db'));

* Chèn hàng vào Table ( mặc định tự động tạo khóa chính )

$table = **new** Bugs();

$data = [array](http://www.php.net/array)(

    'created\_on'      => '2007-03-22',

    'bug\_description' => 'Something wrong',

    'bug\_status'      => 'NEW'

);

$table->insert($data);

*hay: chèn 1 biểu thức vào Table*

$table = **new** Bugs();

$data = [array](http://www.php.net/array)(

    'created\_on'      => **new** Zend\_Db\_Expr('CURDATE()'),

    'bug\_description' => 'Something wrong',

    'bug\_status'      => 'NEW'

);

* + - Tạo Table tự điền khóa\_ Auto-incrementing:

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bugs';

*// This is the default in the Zend\_Db\_Table\_Abstract class;*

*// you do not need to define this.*

    protected $\_sequence = **true**;

}

* + - Xác định trình tự tạo khóa tự động:

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bugs';

    protected $\_sequence = 'bug\_sequence';

}

* + - Sử dụng Table với khóa tự nhiên

**class** BugStatus **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bug\_status';

    protected $\_sequence = **false**;

}

* Cập nhật các dòng trong 1 Table

$table = **new** Bugs();

$data = [array](http://www.php.net/array)(

    'updated\_on'      => '2007-03-23',

    'bug\_status'      => 'FIXED'

);

$where = $table->getAdapter()->quoteInto('bug\_id = ?', 1234);

$table->update($data, $where);

* Xóa các dòng trong 1 Table

$table = **new** Bugs();

$where = $table->getAdapter()->quoteInto('bug\_id = ?', 1235);

$table->delete($where);

* Tìm dòng trong Table với khóa chính

$table = **new** Bugs();

*// Find a single row*

*// Returns a Rowset*

$rows = $table->find(1234);

*// Find multiple rows*

*// Also returns a Rowset*

$rows = $table->find([array](http://www.php.net/array)(1234, 5678));

*Hàng đầu tiên phải có giá trị khóa chính (1234, 'ABC'), và hàng thứ hai phải có giá trị khóa chính (5678, 'DEF').:*

**class** BugsProducts **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bugs\_products';

    protected $\_primary = [array](http://www.php.net/array)('bug\_id', 'product\_id');

}

$table = **new** BugsProducts();

*// Find a single row with a compound primary key*

*// Returns a Rowset*

$rows = $table->find(1234, 'ABC');

*// Find multiple rows with compound primary keys*

*// Also returns a Rowset*

$rows = $table->find([array](http://www.php.net/array)(1234, 5678), [array](http://www.php.net/array)('ABC', 'DEF'));

* Câu truy vấn cho tập hợp các hàng
  + Chọn API

*/\*\**

*\* Fetching a rowset*

*\*/*

$rows = $table->fetchAll(

    'bug\_status = "NEW"',

    'bug\_id ASC',

    10,

    0

    );

$rows = $table->fetchAll(

    $table->select()

        ->where('bug\_status = ?', 'NEW')

        ->order('bug\_id ASC')

        ->limit(10, 0)

    );

*// or with binding*

$rows = $table->fetchAll(

    $table->select()

        ->where('bug\_status = :status')

        ->bind([array](http://www.php.net/array)(':status'=>'NEW')

        ->order('bug\_id ASC')

        ->limit(10, 0)

    );

*/\*\**

*\* Fetching a single row*

*\*/*

$row = $table->fetchRow(

    'bug\_status = "NEW"',

    'bug\_id ASC'

    );

$row = $table->fetchRow(

    $table->select()

        ->where('bug\_status = ?', 'NEW')

        ->order('bug\_id ASC')

    );

*// or with binding*

$row = $table->fetchRow(

    $table->select()

        ->where('bug\_status = :status')

        ->bind([array](http://www.php.net/array)(':status'=>'NEW')

        ->order('bug\_id ASC')

    );

Cách đơn giản :

$table = **new** Bugs();

$select = $table->select();

$select->where('bug\_status = ?', 'NEW');

$rows = $table->fetchAll($select);

*Hay dành cho những người thông thạo:*

$table = **new** Bugs();

$rows =

$table->fetchAll($table->select()->where('bug\_status = ?','NEW'));

* + Lấy 1 rowset

$table = **new** Bugs();

$select = $table->select()->where('bug\_status = ?', 'NEW');

$rows = $table->fetchAll($select);

*ví dụ hàm tìm kiếm từ 1 biểu thức :*

$table = **new** Bugs();

$order  = 'bug\_id';

*// Return the 21st through 30th rows*

$count  = 10;

$offset = 20;

$select = $table->select()->where('bug\_status = ?', 'NEW')

                          ->order($order)

                          ->limit($count, $offset);

$rows = $table->fetchAll($select);

* + Nâng cao

Lấy hàng cụ thể :

$table = **new** Bugs();

$select = $table->select();

$select->from($table, [array](http://www.php.net/array)('bug\_id', 'bug\_description'))

       ->where('bug\_status = ?', 'NEW');

$rows = $table->fetchAll($select);

Lấy biểu thức như các cột:

$table = **new** Bugs();

$select = $table->select();

$select->from($table,

[array](http://www.php.net/array)('COUNT(reported\_by) as `count`', 'reported\_by'))

       ->where('bug\_status = ?', 'NEW')

       ->group('reported\_by');

$rows = $table->fetchAll($select);

Sử dụng một Table tra cứu để tinh chỉnh kết quả của fetchAll ()

$table = **new** Bugs();

*// retrieve with from part set, important when joining*

$select = $table->select(Zend\_Db\_Table::SELECT\_WITH\_FROM\_PART);

$select->setIntegrityCheck(**false**)

       ->where('bug\_status = ?', 'NEW')

       ->join('accounts', 'accounts.account\_name = bugs.reported\_by')

       ->where('accounts.account\_name = ?', 'Bob');

$rows = $table->fetchAll($select);

Loại bỏ các kiểm tra tính toàn vẹn trên Zend\_Db\_Table\_Select để kết nối các hàng

$table = **new** Bugs();

$select = $table->select(Zend\_Db\_Table::SELECT\_WITH\_FROM\_PART)

->setIntegrityCheck(**false**);

$select->where('bug\_status = ?', 'NEW')

->join('accounts',

'accounts.account\_name = bugs.reported\_by',

'account\_name')

->where('accounts.account\_name = ?', 'Bob');

$rows = $table->fetchAll($select);

* Câu truy vấn cho 1 hàng

$table = **new** Bugs();

$select  = $table->select()->where('bug\_status = ?', 'NEW')

                           ->order('bug\_id');

$row = $table->fetchRow($select);

* Lấy thông tin từ Table

$table = **new** Bugs();

$info = $table->info();

[echo](http://www.php.net/echo) "The table name is " . $info['name'] . "**\n**";

*Trong đó:*

name  : tên Table

cols : mảng tên các cột của Table

primary : mảng tên các cột khóa chính

metadata  mảng kết hợp tên, thông tin các cột

rowClass : tên của lớp xác định đối tượng Zend\_Db\_Table\_Row

rowsetClass : tên của lớp xác định đối tượng Zend\_Db\_Table\_Rowset

referenceMap : mảng kết hợp các thông tin về các Table “cha”

dependentTables : tên các lớp của các Table liên quan tới table này

schema  : tên lược đồ hoặc database của Table

* Lưu trữ dữ liệu từ Table

Sử dụng bộ nhớ Cache cho tất cả các đối tượng:

*// First, set up the Cache*

$frontendOptions = [array](http://www.php.net/array)(

    'automatic\_serialization' => **true**

    );

$backendOptions  = [array](http://www.php.net/array)(

    'cache\_dir'                => 'cacheDir'

    );

$cache = Zend\_Cache::factory('Core',

                             'File',

                             $frontendOptions,

                             $backendOptions);

*// Next, set the cache to be used with all table objects*

Zend\_Db\_Table\_Abstract::setDefaultMetadataCache($cache);

*// A table class is also needed*

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

*// ...*

}

*// Each instance of Bugs now uses the default metadata cache*

$bugs = **new** Bugs();

Sử dụng bộ nhớ Cache cho đối tượng cụ thể :

*// First, set up the Cache*

$frontendOptions = [array](http://www.php.net/array)(

    'automatic\_serialization' => **true**

    );

$backendOptions  = [array](http://www.php.net/array)(

    'cache\_dir'                => 'cacheDir'

    );

$cache = Zend\_Cache::factory('Core',

                             'File',

                             $frontendOptions,

                             $backendOptions);

*// A table class is also needed*

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

*// ...*

}

*// Configure an instance upon instantiation*

$bugs = **new** Bugs([array](http://www.php.net/array)('metadataCache' => $cache));

* Nhập bằng tay

Thiết lập cấu trúc :

protected $\_metadata = [array](http://www.php.net/array)(

'<column\_name>' => [array](http://www.php.net/array)(

'SCHEMA\_NAME'      => <string>,

'TABLE\_NAME'       => <string>,

'COLUMN\_NAME'      => <string>,

'COLUMN\_POSITION'  => <int>,

'DATA\_TYPE'        => <string>,

'DEFAULT'          => NULL|<value>,

'NULLABLE'         => <bool>,

'LENGTH'           => <string - length>,

'SCALE'            => NULL|<value>,

'PRECISION'        => NULL|<value>,

'UNSIGNED'         => NULL|<bool>,

'PRIMARY'          => <bool>,

'PRIMARY\_POSITION' => <int>,

'IDENTITY'         => <bool>,

),

*// additional columns...*

);

* Tùy biến và mở rộng các lớp Table
  + - Sử dụng các lớp Row hoặc Rowset

**class** My\_Row **extends** Zend\_Db\_Table\_Row\_Abstract

{

    ...

}

**class** My\_Rowset **extends** Zend\_Db\_Table\_Rowset\_Abstract

{

    ...

}

$table = **new** Bugs(

[array](http://www.php.net/array)(

        'rowClass'    => 'My\_Row',

        'rowsetClass' => 'My\_Rowset'

    )

);

$where = $table->getAdapter()->quoteInto('bug\_status = ?', 'NEW')

*// Returns an object of type My\_Rowset,*

*// containing an array of objects of type My\_Row.*

$rows = $table->fetchAll($where);

*Thay đổi Row và Rowset:*

$table = **new** Bugs();

$where = $table->getAdapter()->quoteInto('bug\_status = ?', 'NEW')

*// Returns an object of type Zend\_Db\_Table\_Rowset*

*// containing an array of objects of type Zend\_Db\_Table\_Row.*

$rowsStandard = $table->fetchAll($where);

$table->setRowClass('My\_Row');

$table->setRowsetClass('My\_Rowset');

*// Returns an object of type My\_Rowset,*

*// containing an array of objects of type My\_Row.*

$rowsCustom = $table->fetchAll($where);

*// The $rowsStandard object still exists, and it is unchanged.*

* + - Thiết lập logic cho Insert, Update, Delete

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

    protected $\_name = 'bugs';

**public** **function** insert([array](http://www.php.net/array) $data)

    {

*// add a timestamp*

        if ([empty](http://www.php.net/empty)($data['created\_on'])) {

            $data['created\_on'] = [time](http://www.php.net/time)();

        }

        return parent::insert($data);

    }

**public** **function** update([array](http://www.php.net/array) $data, $where)

    {

*// add a timestamp*

        if ([empty](http://www.php.net/empty)($data['updated\_on'])) {

            $data['updated\_on'] = [time](http://www.php.net/time)();

        }

        return parent::update($data, $where);

    }

}

* + - Thiết lập phương pháp tìm kiếm cho Zend\_Db\_Table

**class** Bugs **extends** Zend\_Db\_Table\_Abstract

{

$where = $this->getAdapter()->quoteInto('bug\_status = ?',$status);

            protected $\_name = 'bugs';

**public** **function** findByStatus($status)

    {

        return $this->fetchAll($where, 'bug\_id');

    }

}

* + - Thiết lập thay đổi trong Zend\_Db\_Table

abstract **class** MyAbstractTable **extends** Zend\_Db\_Table\_Abstract

{

protected **function** \_setupTableName()

{

if (!$this->\_name) {

$this->\_name = myCustomInflector([get\_class](http://www.php.net/get_class)($this));

}

parent::\_setupTableName();

}

}

**class** BugsProducts **extends** MyAbstractTable

{

}

### Zend\_Db\_Select

* So sánh với cách sử dụng phương thức adapter:
* Câu lệnh dùng phương thức của lớp adapter:

$db = Zend\_Db::factory( ...options... );

$select = $db->select();

* Câu lệnh sử dụng Zend\_Db\_Statement :

$db = Zend\_Db::factory( ...options... );

$select = **new** Zend\_Db\_Select($db);

* Xây dựng các truy vấn Select:

$select = $db->select()

    ->from( ...specify table and columns... )

    ->where( ...specify search criteria... )

    ->order( ...specify sorting criteria... );

*hay :*

$select = $db->select();

$select->from( ...specify table and columns... )

$select->where( ...specify search criteria... )

$select->order( ...specify sorting criteria... );

* Thêm 1 mệnh đề Form:

$select = $db->select()

             ->from( 'products' );

*hay: trong sử dụng bí danh ta dùng*

$select = $db->select()

             ->from( [array](http://www.php.net/array)('p' => 'products') );

* Thêm côt:

*Dạng 1:*

*// Build this query:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id', 'product\_name'));

*Dạng 2:*

*// Build the same query, specifying correlation names:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('p.product\_id', 'p.product\_name'));

*Dạng 3:*

*// Build this query with an alias for one column:*

*//   SELECT p."product\_id" AS prodno, p."product\_name"*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('prodno' => 'product\_id','product\_name'));

* Thêm cột biểu hiện:

*Dạng 1 :*

*// Build this query:*

*//   SELECT p."product\_id", LOWER(product\_name)*

*//   FROM "products" AS p*

*// An expression with parentheses implicitly becomes*

*// a Zend\_Db\_Expr.*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id', 'LOWER(product\_name)'));

*Dạng 2:*

*// Build this query:*

*//   SELECT p."product\_id", (p.cost \* 1.08) AS cost\_plus\_tax*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id',

                          'cost\_plus\_tax' => '(p.cost \* 1.08)')

                   );

*Dạng 3 :*

*// Build this query using Zend\_Db\_Expr explicitly:*

*//   SELECT p."product\_id", p.cost \* 1.08 AS cost\_plus\_tax*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id',

                          'cost\_plus\_tax' =>

**new** Zend\_Db\_Expr('p.cost \* 1.08'))

                    );

* Thêm cột từ 1 form hiện tại hoặc kết nối các bảng:

*Dạng 1:*

*// Build this query:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'), 'product\_id')

             ->columns('product\_name');

*Dạng 2 :*

*// Build the same query, specifying correlation names:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'), 'p.product\_id')

             ->columns('product\_name', 'p');

*// Alternatively use columns('p.product\_name')*

* Thêm 1 bảng khác vào truy vấn với kết nối:

*Dạng 1:*

*// Build this query:*

*//   SELECT p."product\_id", p."product\_name", l.\**

*//   FROM "products" AS p JOIN "line\_items" AS l*

*//     ON p.product\_id = l.product\_id*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id', 'product\_name'))

             ->join([array](http://www.php.net/array)('l' => 'line\_items'),

                    'p.product\_id = l.product\_id');

*Dạng 2:*

*// Build this query:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p JOIN "line\_items" AS l*

*//     ON p.product\_id = l.product\_id*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id', 'product\_name'))

             ->join([array](http://www.php.net/array)('l' => 'line\_items'),

                    'p.product\_id = l.product\_id',

[array](http://www.php.net/array)() ); *// empty list of columns*

Phương pháp hỗ trợ các kiểu tham gia :

**joinUsing(table, join, [columns])** and **joinInnerUsing(table, join, [columns])**

**joinLeftUsing(table, join, [columns])**

**joinRightUsing(table, join, [columns])**

**joinFullUsing(table, join, [columns])**

*// Build this query:*

*//   SELECT \**

*//   FROM "table1"*

*//   JOIN "table2"*

*//   ON "table1".column1 = "table2".column1*

*//   WHERE column2 = 'foo'*

$select = $db->select()

             ->from('table1')

             ->joinUsing('table2', 'column1')

             ->where('column2 = ?', 'foo');

* Thêm mệnh đề điều kiện\_Where:

*Dạng 1:*

*// Build this query:*

*//   SELECT product\_id, product\_name, price*

*//   FROM "products"*

*//   WHERE price > 100.00*

$select = $db->select()

             ->from('products',

[array](http://www.php.net/array)('product\_id', 'product\_name', 'price'))

             ->where('price > 100.00');

*Dạng 2:*

*// Build this query:*

*//   SELECT product\_id, product\_name, price*

*//   FROM "products"*

*//   WHERE (price > 100.00)*

$minimumPrice = 100;

$select = $db->select()

             ->from('products',

[array](http://www.php.net/array)('product\_id', 'product\_name', 'price'))

             ->where('price > ?', $minimumPrice);

*Dạng 3:*

*// Build this query:*

*//   SELECT product\_id, product\_name, price*

*//   FROM "products"*

*//   WHERE (product\_id IN (1, 2, 3))*

$productIds = [array](http://www.php.net/array)(1, 2, 3);

$select = $db->select()

             ->from('products',

[array](http://www.php.net/array)('product\_id', 'product\_name', 'price'))

             ->where('product\_id IN (?)', $productIds);

*Dạng 4:*

*// Build this query:*

*//   SELECT product\_id, product\_name, price*

*//   FROM "products"*

*//   WHERE (price > 100.00)*

*//     AND (price < 500.00)*

$minimumPrice = 100;

$maximumPrice = 500;

$select = $db->select()

             ->from('products',

[array](http://www.php.net/array)('product\_id', 'product\_name', 'price'))

             ->where('price > ?', $minimumPrice)

             ->where('price < ?', $maximumPrice);

*Dạng 5:*

*// Build this query:*

*//   SELECT product\_id, product\_name, price*

*//   FROM "products"*

*//   WHERE (price < 100.00)*

*//     OR (price > 500.00)*

$minimumPrice = 100;

$maximumPrice = 500;

$select = $db->select()

             ->from('products',

[array](http://www.php.net/array)('product\_id', 'product\_name', 'price'))

             ->where('price < ?', $minimumPrice)

             ->orWhere('price > ?', $maximumPrice);

*Dạng 6:*

*// Build this query:*

*//   SELECT product\_id, product\_name, price*

*//   FROM "products"*

*//   WHERE (price < 100.00 OR price > 500.00)*

*//     AND (product\_name = 'Apple')*

$minimumPrice = 100;

$maximumPrice = 500;

$prod = 'Apple';

$select = $db->select()

             ->from('products',

[array](http://www.php.net/array)('product\_id', 'product\_name', 'price'))

             ->where("price < $minimumPrice OR price > $maximumPrice")

             ->where('product\_name = ?', $prod);

* Thêm mệnh đề nhóm kết quả\_Group By:

*// Build this query:*

*//   SELECT p."product\_id", COUNT(\*) AS line\_items\_per\_product*

*//   FROM "products" AS p JOIN "line\_items" AS l*

*//     ON p.product\_id = l.product\_id*

*//   GROUP BY p.product\_id*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id'))

             ->join([array](http://www.php.net/array)('l' => 'line\_items'),

                    'p.product\_id = l.product\_id',

[array](http://www.php.net/array)('line\_items\_per\_product' => 'COUNT(\*)'))

             ->group('p.product\_id');

* Thêm điều kiện trong nhóm kết quả\_ Having :

*// Build this query:*

*//   SELECT p."product\_id", COUNT(\*) AS line\_items\_per\_product*

*//   FROM "products" AS p JOIN "line\_items" AS l*

*//     ON p.product\_id = l.product\_id*

*//   GROUP BY p.product\_id*

*//   HAVING line\_items\_per\_product > 10*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id'))

             ->join([array](http://www.php.net/array)('l' => 'line\_items'),

                    'p.product\_id = l.product\_id',

[array](http://www.php.net/array)('line\_items\_per\_product' => 'COUNT(\*)'))

             ->group('p.product\_id')

             ->having('line\_items\_per\_product > 10');

* Thêm sắp xếp kết quả\_Order By:

*// Build this query:*

*//   SELECT p."product\_id", COUNT(\*) AS line\_items\_per\_product*

*//   FROM "products" AS p JOIN "line\_items" AS l*

*//     ON p.product\_id = l.product\_id*

*//   GROUP BY p.product\_id*

*//   ORDER BY "line\_items\_per\_product" DESC, "product\_id"*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id'))

             ->join([array](http://www.php.net/array)('l' => 'line\_items'),

                    'p.product\_id = l.product\_id',

[array](http://www.php.net/array)('line\_items\_per\_product' => 'COUNT(\*)'))

             ->group('p.product\_id')

             ->order([array](http://www.php.net/array)('line\_items\_per\_product DESC',

                           'product\_id'));

* Thêm giới hạn kết quả\_Limit:

*// Build this query:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

*//   LIMIT 10, 20*

*// Equivalent to:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

*//   LIMIT 20 OFFSET 10*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id', 'product\_name'))

             ->limit(20, 10);

hay : phân trang các kết quả

*// Build this query:*

*//   SELECT p."product\_id", p."product\_name"*

*//   FROM "products" AS p*

*//   LIMIT 10, 20*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products'),

[array](http://www.php.net/array)('product\_id', 'product\_name'))

             ->limitPage(2, 10);

//limitPage( số trang, chiều dài trang )

* Thêm điều kiện tránh trùng lặp kết quả\_ Distinct :

*// Build this query:*

*//   SELECT DISTINCT p."product\_name"*

*//   FROM "products" AS p*

$select = $db->select()

             ->distinct()

             ->from([array](http://www.php.net/array)('p' => 'products'), 'product\_name');

* Thêm 1 cập nhật khi truy vấn kết quả\_For Update:

*// Build this query:*

*//   SELECT FOR UPDATE p.\**

*//   FROM "products" AS p*

$select = $db->select()

             ->forUpdate()

             ->from([array](http://www.php.net/array)('p' => 'products'));

* Xây dựng câu truy vấn Union:

$sql1 = $db->select();

$sql2 = "SELECT ...";

$select = $db->select()

    ->union([array](http://www.php.net/array)($sql1, $sql2))

    ->order("id");

* Thực hiện các truy vấn Select:
* Thực hiện truy vấn Select từ Adapter Db:

$select = $db->select()

             ->from('products');

$stmt = $db->query($select);

$result = $stmt->fetchAll();

* Thực hiện truy vấn Select từ các đối tượng:

$select = $db->select()

             ->from('products');

$stmt = $select->query();

$result = $stmt->fetchAll();

* Đổi 1 đối tượng Select sang chuỗi SQL:

$select = $db->select()

             ->from('products');

$sql = $select->\_\_toString();

[echo](http://www.php.net/echo) "$sql**\n**";

*// The output is the string:*

*//   SELECT \* FROM "products"*

* Các phương pháp khác :
* Lấy các bộ phận của đối tượng Select:

$select = $db->select()

             ->from('products')

             ->order('product\_id');

*// You can use a string literal to specify the part*

$orderData = $select->getPart( 'order' );

*// You can use a constant to specify the same part*

$orderData = $select->getPart( Zend\_Db\_Select::ORDER );

*// The return value may be an array structure, not a string.*

*// Each part has a different structure.*

[print\_r](http://www.php.net/print_r)( $orderData );

* Thiết lập lại bộ phận của đối tượng Select:

*// Build this query:*

*//   SELECT p.\**

*//   FROM "products" AS p*

*//   ORDER BY "product\_name"*

$select = $db->select()

             ->from([array](http://www.php.net/array)('p' => 'products')

             ->order('product\_name');

*// Changed requirement, instead order by a different columns:*

*//   SELECT p.\**

*//   FROM "products" AS p*

*//   ORDER BY "product\_id"*

*// Clear one part so we can redefine it*

$select->reset( Zend\_Db\_Select::ORDER );

*// And specify a different column*

$select->order('product\_id');

*// Clear all parts of the query*

$select->reset();

# Phần 2 : Sử dụng :

## Cách lấy nối và lưu kết nối ở boostrap.php :

#### Bước 1 :

trong file *application.ini* ta thêm các thông số thông thường để kết nối Database thông qua Zend\_Db\_Adapter :

Ví dụ :

[db]

resources.db.adapter = "mysqli"

resources.db.params.host = "localhost"

resources.db.params.username = "root"

resources.db.params.password = "123456"

resources.db.params.dbname = "cms"

#### Bước 2 :

Trong file *boostrap.php* ta thêm đoạn mã tạo liên kết đến Database bằng Zend\_Db\_Adapter:

protected function \_initDb() {  
      $dbOption = $this->getOption('resources');  
       $dbOption = $dbOption['db'];  
         
        // Setup database  
        $db = Zend\_Db::factory($dbOption['adapter'],$dbOption['params']);  
         
        $db->setFetchMode(Zend\_Db::FETCH\_ASSOC);  
        $db->query("SET NAMES 'utf8'");  
        $db->query("SET CHARACTER SET 'utf8'");         
         
        Zend\_Registry::set('connectDB',$db);  
                  
        //Khi thiet lap che do nay model moi co the su dung duoc  
        Zend\_Db\_Table::setDefaultAdapter($db);  
          
         
        return $db;  
   }

## Cách sử dụng : lấy kết nối ở controller, action, cách sử dụng câu truy vấn

Trong Action của controller, muốn thao tác tới database ta phải khai báo kết nối Database:

$db = Zend\_Registry::*get*('connectDB');

Sau đó ta có thể sử dụng các câu truy vấn. Ví dụ:

$sql = "SELECT member\_id, member\_name, user\_name    
        FROM members   
        WHERE group\_id = 'g002'      
        AND member\_status = 1";  
$db->fetchAll($sql);

## Demo :

Tài liệu tham khảo :

<http://framework.zend.com/manual/en/zend.db.html>

<http://zend.vn/forum/viewforum.php?f=17&sid=a06698d04fa166235c0b961c988948d8>