Database Access in Codeginiter using PHP

CS 183

11/3/2011

Basics

- Setup MYSQL completed
  - Download http://www.mysql.com/downloads/
  - Follow instructions and set it up
- Setup apache completed
  - Depends on OS
- Tooling for Database access / manipulation:
  - Setup PhpMyAdmin (see http://www.phpmyadmin.net/home_page/downloads.php)
  - Mac: SequelPro (see http://www.sequelpro.com/)
  - Design, develop, administer: http://www.mysql.com/products/workbench/

Create you database and tables

- Basics covered in previous lectures on table design
- Use the tools to create your database and tables
  - SQL commands in .sql file
  - Or: phpMyAdmin / SequelPro allow SQL commands as well as interactive GUI
- Now setup your database connection
  - Need user id / password
  - `application/config/database.php`
    - Don't check this file into SVN!
    - Keep your credentials save!

Sample SQL DB Creation

```
CREATE DATABASE `regform`;
USE `regform`;
CREATE TABLE `participants` (
`id` INT(3) UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY
KEY ,
`fname` VARCHAR(255) NOT NULL ,
`email` VARCHAR(255) NOT NULL ,
`comments` TEXT NOT NULL ,
`seminar_1` VARCHAR(255) NOT NULL ,
`seminar_2` VARCHAR(255) NOT NULL ,
`seminar_3` VARCHAR(255) NOT NULL ,
`seminar_4` VARCHAR(255) NOT NULL
) ENGINE = innodb;
```

database.php

```
/*
| -------------------------------------------------------------------
| DATABASE CONNECTIVITY SETTINGS
| -------------------------------------------------------------------
| This file will contain the settings needed to access your database.
|
| For complete instructions please consult the "Database Connection"
| page of the User Guide.
|

$active_group = "default";
$active_record = TRUE;

$db['default']['hostname'] = ”myserver.com";
$db['default']['username'] = "db897857_frontend";
$db['default']['password'] = "Li!B33kdk6EV8X";
$db['default']['database'] = "db897857_development";
$db['default']['dbdriver'] = "mysql";
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
```

Working with DB in CI

- Regular PHP DB
- Use ‘db’ object with low level SQL methods
- ActiveRecord
- IgniteRecord

- All of them work well
- ActiveRecord / IgniteRecord have advantage of higher level abstraction level

Need models for your tables …

- A model is a PHP class
- In application/models folder
- Model is a class, first name uppercase required
  - E.g., class User_model extends CI_Model
  - Corresponds to application/models/ user_model.php

```
class Blogmodel extends CI_Model {
    var $title   = '';
    var $content = '';
    var $date    = '';
    function __construct()
    {
        // Call the Model constructor
        parent::__construct();
    }
    function get_last_ten_entries()
    {
        $query = $this->db->get('entries', 10);
        return $query->result();
    }
    function insert_entry()
    {
        $this->title   = $_POST['title']; // please read the below note
        $this->content = $_POST['content'];
        $this->date    = time();

        $this->db->insert('entries', $this);
    }
    function update_entry()
    {
        $this->title   = $_POST['title'];
        $this->content = $_POST['content'];
        $this->date    = time();

        $this->db->update('entries', $this, array('id' => $_POST['id']));
    }
}
```

Model prototype

```
class Model_name extends CI_Model {

    function __construct()
    {
        parent::__construct();
    }
}
```

Loading a model

- In controller:

```
$this->load->model('Model_name');
```

- Access:

```
$this->Model_name->function();
```

Example

```
class Blog_controller extends CI_Controller {

    function blog()
    {
        $this->load->model('Blog');

        $data['query'] =
             $this->Blog->get_last_ten_entries();

        $this->load->view('blog', $data);
    }
}
```

Connection

- Model does not automatically connect to DB, however can be done as follows:
  - `$this->load->model('Model_name','',TRUE);`
- Alternative: auto-connect (recommended):
  - Feature enabled via application/config/autoload.php
  - Example: $autoload['libraries'] = array('database', 'ignitedrecord/Ignitedrecord','session','form_validation');
- Manuel connect also an option:
  - $this->load->database('group_name');
- More details:
  - http://codeigniter.com/user_guide/database/connecting.html
  - Also discusses multiple db connections, keeping them alive, closing etc.

Simple Queries

```
function list_participants()
  {
    $query =
      $this->db->query("SELECT * FROM
participants");

    return $query;
  }
```

- This is a simple select statement
- Controller can now obtain data, manipulate, and pass into view

- In controller:

```
function results()
  {
    $this->load->model('Form_model', '', TRUE);
    $data['query'] = $this->Form_model-
>list_participants();
    $this->load->view('results', $data);
  }
```

How to access data in view?

```
<?php
if($query->num_rows() > 0):
   foreach ($query->result() as $row):
?>
        <tr>
                <td><?php echo $row->id; ?></td>
                <td><?php echo $row->fname; ?></td>
                <td><?php echo $row->email; ?></td>
                <td><?php echo $row->seminar_1; ?></td>
                <td><?php echo $row->seminar_2; ?></td>
                <td><?php echo $row->seminar_3; ?></td>
                <td><?php echo $row->seminar_4; ?></td>
                <td><?php echo $row->comments; ?></td>
        </tr>
<?php
        endforeach;
endif;
?>
```

A full tutorial can be found here:
http://godbit.com/article/introduction-to-code-igniter-part-3

ActiveRecord Pattern

- See Wiki for background
  - http://en.wikipedia.org/wiki/Active_Record
- CI uses Active Record
- Advantages:
  - Simpler queries
  - Database independent apps
  - Security – values are automatically escaped

Function overview

- Retrieving data
- Inserting data
- Updating data
- Deleting data

db->get()

```
$query = $this->db->get('mytable');
// Produces: SELECT * FROM mytable


Then:

foreach ($query->result() as $row)
{
    echo $row->title;
}
```

db->get_where() db->where() db->where_or() db ->where_in()

```
$query =
$this->db->get_where('mytable', array('id' =>
$id), $limit, $offset);
```

OR:

```
$this->db->where('name', $name);
// Produces: WHERE name = 'Joe'
```

Note:

- If you use multiple where() they will be joined together using AND semantics
- You can add operator where('name !=', $name);
- Associative array is also possible, or just use regular SQL syntax (pass in as string)

db->select()

```
$this->db->select('title, content, date');

$query = $this->db->get('mytable');

// Produces: SELECT title, content, date FROM
mytable
```

db->from()

- `$this->db->select('title, content, date');`
- `$this->db->from('mytable');`

- `$query = $this->db->get();`

- `// Produces: SELECT title, content, date FROM mytable`

db->join()

```
$this->db->select('*');
$this->db->from('blogs');
$this->db->join('comments', 'comments.id = blogs.id');

$query = $this->db->get();

// Produces:
// SELECT * FROM blogs
// JOIN comments ON comments.id = blogs.id
```

Note: optional 3rd parameters allows to specify type of join (left, right, outer, inner, left outer, right outer)

db->like() db->not_like()

```
$this->db->like('title', 'match');
$this->db->or_like('body', $match);

// WHERE title LIKE '%match%' OR body
LIKE '%match%'
```

Other useful retrieval functions

- `db->select_max()`
- `db->select_min()`
- `db->select_avg()`
- `db->select_sum()`

- `db->where_not_in()`
- `db->or_where_not_in()`
- `db->or_not_like()`
- `db_->group_by()`
- `db->distinct()`
- `db->having() db->or_having()`
- `db->limit()`
- `db->count_all)results()`
- `db->count_all()`

Inserting data …

- Use db->insert()

```
$data = array(
   'title' => 'My title' ,
   'name' => 'My Name' ,
   'date' => 'My date'
);

$this->db->insert('mytable', $data);

// Produces: INSERT INTO mytable (title,
name, date) VALUES ('My title', 'My name',
'My date')
```

db->set()

- `$this->db->set('name', $name);`
- `$this->db->insert('mytable');`

- `// Produces: INSERT INTO mytable (name) VALUES ('{$name}')`

db->update()

```
$data = array(
                'title' => $title,
                'name' => $name,
                'date' => $date
            );

$this->db->where('id', $id);
$this->db->update('mytable', $data);

// Produces:
// UPDATE mytable
// SET title = '{$title}', name = '{$name}',
date = '{$date}'
// WHERE id = $id
```

Method Chaining

```
$this->db->select('title')->from('mytable')-
>where('id', $id)->limit(10, 20);


$query = $this->db->get();
```

Active Record Caching

- Allows to “cache” certain parts of a query
- Cached calls are cumulative

- $this->db->start_cache()
- $this->db->stop_cache()
- $this->db->flush_cache()

Caching Example

```
$this->db->start_cache();
$this->db->select('field1');
$this->db->stop_cache();
$this->db->get('tablename');
//Generates: SELECT `field1` FROM (`tablename`)

$this->db->select('field2');
$this->db->get('tablename');
//Generates: SELECT `field1`, `field2` FROM (`tablename`)

$this->db->flush_cache();
$this->db->select('field2');
$this->db->get('tablename');
//Generates: SELECT `field2` FROM (`tablename`)
```