**DW508 Problem set – Classes Part 2**

CASE STUDY – Postcodes

Purpose:

* Save the content of a CSV file containing details of Australian post codes to MYSQL database using OOP.
* Develop OO representation of the database
* The code for reading CSV file is already written for you. You will have to provide the classes that the code relies on:

//in index.php

<?php

require('./model/database.php');

require('./model/PostCode.php');

require('./model/PostCodeDB.php');

$lines = file('./ postcodes\_small.csv');

foreach ($lines as $line) {

if (substr($line, 0, strlen('postcode')) == 'postcode')

continue;//ignore the first line with headers

//postcode’s id is set to zero; it will be ignored   
 //when the data is saved to the database

PostCodeDB::add(PostCode::fromCSV('0,' . $line));

}

?>

Preparation

* In MYSQL create new database called postcodes to store information about Australian Postcodes. Use supplied sql file. It contains two records to simplify testing.
* They are two CVS files provided. The small one contains a small subset of the database for the testing purposes. The full database contains all AU postcodes records.

Create two classes: PostCode and PostCodeDB. The first one is responsible for storing information about a single postcode, the seconds one is responsible for database operations involving postcodes.

1. Create class PostCode. Use the following code as a starting point.

class PostCode {

private $id;

private $postcode;

private $suburb;

private $state;

private $lat;

private $lng;

……..

}

Provide **setters** and **getters** for all data members with and exception of $id (no set method as id will be set by database and as such not modifiable) plus a suitable constructor.

* 1. Implement a \_toString() that would convert the object to a single CSV string, for example:  
     800,DARWIN,NT,-12.801028,130.955789
  2. Add a **static** method public static function fromCSV($line) to create an object of the type PostCode from a string representing a single CSV line.

1. Implement PostCodeDB via **static** methods. The first two methods are given below. The starting code is in the source\_code directory. For Netbeans users, use PHP project from existing sources. Update database.php file.

class PostcodeDB {

public static function getPostCode($id) {

$db = Database::getDB();

$query = "SELECT \* FROM postcodes WHERE id = :id";

try {

$statement = $db->prepare($query);

$statement->bindValue(':id', $id);

$statement->execute();

$r = $statement->fetch();

$statement->closeCursor();

$postcode = new PostCode($r['id'], $r['postcode'], $r['suburb'],

$r['state'], $r['lat'], $r['lng']);

return $postcode;

} catch (PDOException $e) {

$error\_message = $e->getMessage();

echo $error\_message; exit();

}

}

public static function add($p) {

$db = Database::getDB();

$query = "INSERT INTO `postcodes`

( `postcode`, `suburb`, `state`, `lat`, `lon`)

VALUES (:postcode,:suburb,:state,:lat,:lng)";

try {

$statement = $db->prepare($query);

$statement->bindValue(':postcode', $p->getPostcode());

$statement->bindValue(':suburb', $p->getSuburb());

$statement->bindValue(':state', $p->getState());

$statement->bindValue(':lat', $p->getLat());

$statement->bindValue(':lng', $p->getLng());

$statement->execute();

} catch (PDOException $e) {

$error\_message = $e->getMessage();

echo $error\_message; exit();

}

}

1. If everything goes OK you should be able to run the code form the top of the previous page and update the database.
2. Implement additional methods for PostcodeDB.
   1. public static function get\_postcodes\_by\_postcode($code) - note: it returns an array of objects as more than one suburb can have the same postcode. After you get the $result (array of PDO objects) use the following

$postcodes = array();

foreach ($results as $r) {

Create new Postcode Object using PDO data

Store new object into the array

}

* 1. public static function is\_postcode\_valid($code), checks if given number represents a valid postcode. For example 801 yes; 809 no.
  2. public static function get\_postcodes\_by\_suburb($suburb) - note: it returns an array - use LIKE to support partial searches (for example “Foot” for Footscray).
  3. public static function get\_postcodes\_by\_state($state) - note: it returns an array
  4. public static function DELETE postcode
  5. public static function Update postcode

1. Test you methods.
2. Challenge: use get\_postcodes\_by\_postcode($code) and the sample file maps\_multiple\_markers.html to display a map that has markers for all suburbs belonging to postcode 810.