Putting profesional PHP code on the web. An example.

Kjell Bleivik 16 february 2010:

http://www.digitalpunkt.no/ and http://www.kjellbleivik.com/

Ed Lecky-Thompson, Steven D. Nowicki and Thomas Meyer: Professional PHP6

http://www.wrox.com/WileyCDA/WroxTitle/Professional-PHP6.productCd-0470395095.html

Testing the last example in chapter 07 on the web with minor modifications.

1. KBErrorReporting.php

```
<?php
/*

Used to override default server error reporting.  Include it where you need errors and
warnings.

To turn on Error Reporting in PHP you can set this anywhere in your PHP code, but it has to
be above the error or
else it will not work. This is the easiest way and will work in most hosting environments:

ini_set("display_errors", 1);
error_reporting(E_ALL|E_STRICT);

or if you only want to see Warning Messages and not Notice Messages:

ini_set('display_errors',1);
error_reporting(E_ALL); or error_reporting (E_ALL ^ E_NOTICE);

and if you just want all Error Reporting off, simply use this:

error_reporting(0);
*/
ini_set("display_errors", 1);  // To display error
error_reporting(E_ALL|E_STRICT);
?>
```

2. KBConnectionPDO.php

```
<?php
    require_once("class.PdoFactory.php");

    $strDSN = "pgsql:dbname=kbleivik_prophp6;host=localhost;port=5432";
      $objPDO = PDOFactory::GetPDO($strDSN, "kbleivik_kjell", "YourPassword",
                array());
      $objPDO->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

```
            echo "<br />" ;
            echo "Connected successfully to database." ;
            echo "<br />" ;
            echo "<br />" ;
?>
```

3. class.PDoFactory.php

```
<?php
class PDOFactory {

        public static function GetPDO($strDSN, $strUser, $strPass, $arParms) {

                $strKey = md5(serialize(array($strDSN, $strUser, $strPass, $arParms)));
                if (!($GLOBALS["PDOS"][$strKey] instanceof PDO)) {
                        $GLOBALS["PDOS"][$strKey] = new PDO($strDSN, $strUser,
$strPass, $arParms);
                };
                return($GLOBALS["PDOS"][$strKey]);
        }
}
?>
```

4. KBCreateTable.php

```
<?php
    require_once('../KBErrorReporting.php');
            require_once('../KBConnection.php');

             $sql1 =  "CREATE TABLE \"system_user\" (\"id\" SERIAL PRIMARY KEY NOT
NULL, \"first_name\" character varying(64), \"last_name\" character varying(128),
\"username\" character varying(32), \"md5_pw\" character varying(32), \"email_address\"
character varying(128), \"date_last_login\" date, \"time_last_login\" time,
\"date_account_created\" date, \"time_account_created\" time)";
             $rs1 = pg_query($hDB, $sql1);
     if(! is_resource($rs1)) {
       throw new Exception("An error occurred selecting from the database.");
     }
             else
                     echo "Table created sucessfully";
             echo "<br />" ;
             echo "<br />" ;

             $sql2 =  "GRANT ALL PRIVILEGES ON \"system_user\" TO \"kbleivik_kjell\"";
             $rs2 = pg_query($hDB, $sql2);
     if(! is_resource($rs2)) {
       throw new Exception("An error occurred granting privileges to kbleivik_kjell on table
system_user");
     }
             else
```

```
                    echo "Granted privileges to kbleivik_kjell sucessfully on table system_user";
          echo "<br />" ;
          echo "<br />" ;

          $sql3 =  "GRANT ALL PRIVILEGES ON \"system_user_id_seq\" TO
\"kbleivik_kjell\"";
          $rs3 = pg_query($hDB, $sql3);
    if(! is_resource($rs3)) {
      throw new Exception("An error occurred granting privileges to kbleivik_kjell on
system_user_id_seq");
    }
          else
                    echo "Granted privileges to kbleivik_kjell sucessfully on system_user_id_seq";

          pg_close($hDB);
?>
```

5. KBDropTable.php

```
<?php
          require_once('../KBErrorReporting.php');
          require_once('../KBConnection.php');

          $sql3 =  "DROP TABLE \"system_user\"";
          $rs3 = pg_query($hDB, $sql3);
    if(! is_resource($rs3)) {
      throw new Exception("An error occurred selecting from the database.");
    }
          else
                    echo "Table dropped sucessfully";
          pg_close($hDB);

?>
```

6. class.DataBoundObject.php

```
<?php

abstract class DataBoundObject {

  protected $ID;
  protected $objPDO;
  protected $strTableName;
  protected $arRelationMap;
  protected $blForDeletion;
  protected $blIsLoaded;
  protected $arModifiedRelations;

  abstract protected function DefineTableName();
abstract protected function DefineRelationMap();

public function __construct(PDO $objPDO, $id = NULL) {
  $this->strTableName = $this->DefineTableName();
  $this->arRelationMap = $this->DefineRelationMap();
  $this->objPDO = $objPDO;
  $this->blIsLoaded = false;
  if (isset($id)) {
    $this->ID = $id;
  };
  $this->arModifiedRelations = array();
}

public function Load() {
  if (isset($this->ID)) {
              $strQuery = "SELECT ";
              foreach ($this->arRelationMap as $key => $value) {
                    $strQuery .= "\"" . $key . "\",";
              }
              $strQuery = substr($strQuery, 0, strlen($strQuery)-1);
              $strQuery .= " FROM " . $this->strTableName . " WHERE
                      \"id\" = :eid";
              $objStatement = $this->objPDO->prepare($strQuery);
              $objStatement->bindParam(':eid', $this->ID,
                                  PDO::PARAM_INT);
              $objStatement->execute();
              $arRow = $objStatement->fetch(PDO::FETCH_ASSOC);
              foreach($arRow as $key => $value) {
                    $strMember = $this->arRelationMap[$key];
                    if (property_exists($this, $strMember)) {
                          if (is_numeric($value)) {
                                eval('$this->' . $strMember .
                                   ' = ' . $value . ';');
                          } else {
                                eval('$this->' . $strMember .
                                   ' = "' . $value . '";');
                          };
                    };
              };
    $this->blIsLoaded = true;
  };
}

public function Save() {
  if (isset($this->ID)) {
    $strQuery = 'UPDATE "' . $this->strTableName . '" SET ';
              foreach ($this->arRelationMap as $key => $value) {
                    eval('$actualVal = &$this->' . $value . ';');
      if (array_key_exists($value, $this->arModifiedRelations)) {
                      $strQuery .= '"' . $key . "\" = :$value, ";
    };
            }
  $strQuery = substr($strQuery, 0, strlen($strQuery)-2);
  $strQuery .= ' WHERE "id" = :eid';
  unset($objStatement);
            $objStatement = $this->objPDO->prepare($strQuery);
  $objStatement->bindValue(':eid', $this->ID, PDO::PARAM_INT);
            foreach ($this->arRelationMap as $key => $value) {
                  eval('$actualVal = &$this->' . $value . ';');
      if (array_key_exists($value, $this->arModifiedRelations)) {
        if ((is_int($actualVal)) || ($actualVal == NULL)) {
          $objStatement->bindValue(':' . $value, $actualVal,
                          PDO::PARAM_INT);
        } else {
          $objStatement->bindValue(':' . $value, $actualVal,
                          PDO::PARAM_STR);
        };
      };
            };
  $objStatement->execute();
} else {
  $strValueList = "";
  $strQuery = 'INSERT INTO "' . $this->strTableName . '"(';
  foreach ($this->arRelationMap as $key => $value) {
                 eval('$actualVal = &$this->' . $value . ';');
    if (isset($actualVal)) {
      if (array_key_exists($value, $this->arModifiedRelations)) {
        $strQuery .= '"' . $key . '", ';
        $strValueList .= ":$value, ";
      };
    };
            }
  $strQuery = substr($strQuery, 0, strlen($strQuery) - 2);
  $strValueList = substr($strValueList, 0, strlen($strValueList) - 2);
  $strQuery .= ") VALUES (";
  $strQuery .= $strValueList;
  $strQuery .= ")";

  unset($objStatement);
  $objStatement = $this->objPDO->prepare($strQuery);
            foreach ($this->arRelationMap as $key => $value) {
                 eval('$actualVal = &$this->' . $value . ';');
                 if (isset($actualVal)) {
      if (array_key_exists($value, $this->arModifiedRelations)) {
        if ((is_int($actualVal)) || ($actualVal == NULL)) {
                               $objStatement->bindValue
                     (':' . $value, $actualVal, PDO::PARAM_INT);
                          } else {
                               $objStatement->bindValue
                     (':' . $value, $actualVal, PDO::PARAM_STR);
```

```
                                    };
                  };
                           };
                      }
        $objStatement->execute();
        $this->ID = $this->objPDO->lastInsertId($this->strTableName . "_id_seq");
      }
    }

    public function MarkForDeletion() {
      $this->blForDeletion = true;
    }

    public function __destruct() {
      if (isset($this->ID)) {
        if ($this->blForDeletion == true) {
          $strQuery = 'DELETE FROM "' . $this->strTableName . '" WHERE
                   "id" = :eid';
          $objStatement = $this->objPDO->prepare($strQuery);
          $objStatement->bindValue(':eid', $this->ID, PDO::PARAM_INT);
                  $objStatement->execute();
        };
      }
    }

    public function __call($strFunction, $arArguments) {  //The object overload method

      $strMethodType = substr($strFunction, 0, 3);
      $strMethodMember = substr($strFunction, 3);
      switch ($strMethodType) {
        case "set":
          return($this->SetAccessor($strMethodMember, $arArguments[0]));
          break;
        case "get":
          return($this->GetAccessor($strMethodMember));
      };
      return(false);
    }

    private function SetAccessor($strMember, $strNewValue) {  //Generic setter method.
      if (property_exists($this, $strMember)) {
        if (is_numeric($strNewValue)) {
          eval('$this->' . $strMember . ' = ' . $strNewValue . ';');
        } else {
          eval('$this->' . $strMember . ' = "' . $strNewValue . '";');
        };
        $this->arModifiedRelations[$strMember] = "1";
      } else {
        return(false);
      };
```

```
  }

  private function GetAccessor($strMember) {   //Generic getter method.
    if ($this->blIsLoaded != true) {
      $this->Load();
    }
    if (property_exists($this, $strMember)) {
      eval('$strRetVal = $this->' . $strMember . ';');
      return($strRetVal);
    } else {
      return(false);
    };
  }

}

?>
```

7. class.User3.php

```
<?php

class User extends DataBoundObject {

        protected $FirstName;
        protected $LastName;
        protected $Username;
        protected $Password;
        protected $EmailAddress;

        protected $DateLastLogin;
        protected $TimeLastLogin;
        protected $DateAccountCreated;
        protected $TimeAccountCreated;

        protected function DefineTableName() {
                return("system_user");
        }

        protected function DefineRelationMap() {   // Object relation mapping.
                return(array(
                        "id" => "ID",
                        "first_name" => "FirstName",
                        "last_name" => "LastName",
                        "username" => "Username",
                        "md5_pw" => "Password",
                        "email_address" => "EmailAddress",
                        "date_last_login" => "DateLastLogin",
                        "time_last_login" => "TimeLastLogin",
                  "date_account_created" => "DateAccountCreated",
                  "time_account_created" => "TimeAccountCreated"));
      }
}

?>
```

8. testUser3.php (The final script).

```
<?php
      /*

                  Notice the default error, warning and notices set by the server configuration are
used if.
                  Comment out the line(s) below to override the default settings.
                  require_once('../KBErrorReporting.php');  //Needed if error reporting is off.
Comment out next line if needed.
                  error_reporting (E_ALL ^ E_NOTICE);  //Overriding above setting and turns
off NOTICES but keep warnings.
                  error_reporting(0);  //Turns off all Error Reporting

                  */
                  require_once('../KBConnectionPDO.php');  // Defines $strDSN and $objPDO
                  require_once("class.DataBoundObject.php");
                  require_once("class.User3.php");

      print "Running...<br />";

      $objUser = new User($objPDO);

      $objUser->setFirstName("Steve");
      $objUser->setLastName("Nowicki");
      $objUser->setDateAccountCreated(date("Y-m-d"));

      print "First name is " . $objUser->getFirstName() . "<br />";
      print "Last name is " . $objUser->getLastName() . "<br />";

      print "Saving...<br />";

      $objUser->Save();

      $id = $objUser->getID();
      print "ID in database is " . $id . "<br />";

      print "Destroying object...<br />";
      unset($objUser);

      print "Recreating object from ID $id<br />";
      $objUser = new User($objPDO, $id);
```

```
        print "First name is " . $objUser->getFirstName() . "<br />";
        print "Last name is " . $objUser->getLastName() . "<br />";

        print "Committing a change.... Steve will become Steven,
              Nowicki will become Nowickcow<br/>";
        $objUser->setFirstName("Steven");
        $objUser->setLastName("Nowickcow");
        print "Saving...<br />";
        $objUser->Save();
?>
```

Tested february 16. 2010.

PHP 5.2.9.Output is correct:

```
Connected successfully to database.

Running...
First name is Steve
Last name is Nowicki
Saving...
ID in database is 13
Destroying object...
Recreating object from ID 13
First name is Steve
Last name is Nowicki
Committing a change.... Steve will become Steven, Nowicki will become Nowickcow
Saving...
```

I dropped the table and recreated it.

PHP 6.0.0. Dev Output is wrong:

```
Connected successfully to database.

Running...
First name is Steve
Last name is Nowicki
Saving...
ID in database is 8
Destroying object...
Recreating object from ID 8
First name is   Missing
Last name is   Missing
Committing a change.... Steve will become Steven, Nowicki will become Nowickcow
Saving...
```