يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

بســـم اللـــه الرحمـــن الرحيـــم

سلسلة أنماط التصاميم البرمجية Desgin Patterns, المجموعة الكاملة للدروس للاخ الحسين جعلها الله فى ميزان حسناته

مقدمه:

سوف نناقش هذه السلسلة ان شاء الله علي عدة حلقات لأنه موضوع يطول شرحه , وهو موضوع لا يختص بشرح أساسيات الphp
وأرجو التأكيد علي أهمية هذا الموضوع لمبرمجي الويب عامة(php أو asp أو jsp أو perl إلخ) , إلا أني سوف أقتصر الشرح بالأمثلة علي لغة البي أتش بيphp

ملاحظة:
من لديه الرغبة أن يضيف اي معلومة في هذا الموضوع لم أقم بشرحها أو اي شئ من هذا القبيل , فالباب مفتوح للجميع حتي تعم الفائدة . وذلك على موقع الفريق العربي للبرمجه www.arabteam2000-forum.com قسم الـPHP.

المتطلبات التي يجب توافرها في من سوف يتابع هذه السلسلة:
-أن يملك فكر برمجي بشكل عام ويفضل (ولا يلزم) مبرمجي الويب
-يعرف أساسيات البرمجة بالـphp
-يملك أساسيات البرمجة الكائنيةobject oriented

لكن ما هي Design Patterns أو أنماط التصاميم البرمجية وما أهمية هذا الموضوع

تعريف أنماط التصاميم البرمجية: Desgin Patterns
ممكن أن نعرفها بشكل مبسط أنها أساليب وأفكار برمجية تم التعارف عليها بين المبرمجين
وهي تختص بعرض حلول بشكل جيد للمشاكل البرمجية , كما سوف نري فيما بعد
وهي لا تختص لمبرمجي الويب فقط بل للمبرمجين بشكل عام , لذلك أرجو لمن سوف يتابع هذه السلسلة من المبرمجين غير مبرمجي الويب
أن يحاول أن يستخلص إسلوب هذا الفكر ولا يهمل الموضوع , ولولا ضيق الوقت لأفردت دروس خاصة لمبرمجي ال DotNet والجافا.

أهمية هذه السلسلة:
سوف تعطي مطوري الويب الإسلوب الصحيح لتطوير مشاريعهم وطرق إتخاذهم للقرار في اي الطرق أفضل لإتمام مهمة ما.

الحلقة الأولي Singleton Pattern :

يتم تطبيق هذا النمط عندما نريد عمل كائن واحد single instance ويكون علي هيئة متغير عام مشترك shared global variable

لكن ما هي الحاجة لعمل ذلك ؟
ربما تواجه حالات تحتاج فيها للتوفير في إستهلاك موارد الجهاز (resources مثلا مثل الذاكرة memory)
لذلك تلجأ إلي عمل متغير ما يكون مشترك shared بين عمليات أخري , ومن أهم هذه الحالات (وبالطبع يوجد حالات أخري)
عندما نريد التعامل مع الحالات التالية
-ربما عندما نريد أن نفتح ملف ما للقراءة أو الكتابة ويكون هذا الملف دائم الإستخدام , كما هو الحال مع ملفات المتابعة log files.
-عند إجراء إتصال بقاعدة بيانات ما , فلتجنب إجهاد محرك قاعدة البيانات database engine وأيضا لتجنب إستهلاك الذاكرة
يفضل عمل إتصال واحد يكون عام ومشترك للإستخدام لجميع العمليات التي تتم علي قاعدة البيانات , دون اللجوء إلي فتح إتصال جديد بقاعدة البيانات كلما نريد تنفيذ أمرا ما علي قاعدة البيانات

ملاحظة : قد تم الإلتفات إلي هذه الحالة في ال php وهي الأن لا تسبب مشكلة ولا داعي لإستخدام نمط singleton لأن دوال التعامل مع قواعد البيانات معظمها الأن يوفر الإتصال المستمر , persistent connection كما هو الحال مثلا عند إستخدام الدالةmysql_pconnect وكما نعلم أن وظيفة الدالة mysql_pconnect يكافئ الدالة mysql_connect ولكن تختلف في التالي:
+أنه عند طلب إجراء فتح إتصال جديد بقاعدة البيانات يتم البحث أولا علي إتصال موجود بالفعل يحمل نفس خصائص الإتصال المراد فتحه (اسم السيرفر المضيف host واسم مستخدم قاعدة البيانات وكلمة المرور الخاصة به)
فإذا وجد إتصال مماثل يتم إستخدامه دون اللجوء إلي عمل إتصال جديد

+لذلك لن يتم إغلاق هذا الإتصال بعد تنفيذ اي أمر علي قاعدة البيانات
بالإضافة إلي أنه أيضا لن يتم إغلاق الإتصال (الذي تم فتحه من خلال mysql_pconnect) حتي عند إستدعاء الدالة mysql_close .

لاحظ أن التعامل مع السيشن session هو مثال حي لتطبيق مفهوم singleton pattern

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

مثال تطبيقي لتوضيح هذا النمط

نريد تصميم صفحة تقوم بجلب وعرض بيانات المنتجات products من جدول المنتجات التالي

```
#
# Table structure for table `products`
#

CREATE TABLE products (
 PRODUCTID int(11) NOT NULL auto_increment,
 PRODUCTNAME varchar(255) NOT NULL default '',
 UNITPRICE varchar(255) NOT NULL default '',
 UNITSINSTOCK varchar(255) NOT NULL default '',
 DISCONTINUED varchar(255) NOT NULL default '',
 PRIMARY KEY  (PRODUCTID)
) TYPE=MyISAM;

#
# Dumping data for table `products`
#

INSERT INTO products VALUES (1, 'Chai', '18', '39', '0');
INSERT INTO products VALUES (2, 'Chang', '19', '17', '0');
INSERT INTO products VALUES (3, 'Aniseed Syrup', '10', '13', '0');
INSERT INTO products VALUES (4, 'Chef Antons Cajun Seasoning', '22', '53', '0');
INSERT INTO products VALUES (5, 'Chef Anton Gumbo Mix', '21.35', '0', '1');
INSERT INTO products VALUES (6, 'Grandma Boysenberry Spread', '25', '120', '0');
INSERT INTO products VALUES (7, 'Uncle Bob Organic Dried Pears', '30', '15', '0');
INSERT INTO products VALUES (8, 'Northwoods Cranberry Sauce', '40', '6', '0');
INSERT INTO products VALUES (9, 'Mishi Kobe Niku', '97', '29', '1');
INSERT INTO products VALUES (10, 'Ikura', '31', '31', '0');
```

بافتراض أننا لا نملك أوامر الإتصال المستمر بقاعدة البيانات

الحل:

-كما أشرنا من قبل أن إجراء عملية فتح إتصال جديد بقاعدة البيانات , هو أمر مكلف في وقت التنفيذ وإستهلاك في الذاكرة

-لذلك نحتاج لعمل إتصال واحد فقط بحيث يمكننا إستخدامه كلما إستدعت الحاجة لذلك

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

سنقوم بعمل صفحة باسم Connection.php تحتوي علي الفئة Connection التي تحتوي الدالة getConnection الخاصة لإرجاع معرف الإتصال بقاعدة البيانات
كما يلي

```
<?
class Connection {

  /**
   * getConnection
   *
   * get only one object refere to link identifier resource
   * to avoid multi-connection open
   *
   * @return link identifier
   */
  function getConnection() {
     static $conn;

     if( !isset($conn) ) {
        echo "<h1>Singleton Object Created</h1>";
        $conn = mysql_connect("localhost","root","");
        mysql_select_db("test", $conn);
     }

     return $conn;
  }
}
?>
```

تكمن الفكرة كما نلاحظ بالكود السابق في تعريف متغير ساكن static variable بداخل الدالةgetConnection باسم conn كما يلي:

```
  function getConnection() {
     static $conn;
```

وهذا المتغير يتم الإحتفاظ بقيمة حتي بعد إنتهاء تنفيذ الدالة ولا يتم تدميره لأنه لا يعتبر متغير محليlocal variable

ثم نقوم بعمل صفحة باسم Products.php الخاصة بعرض بيانات المنتجات كما يلي:

```
<?
include_once "Connection.php";
?>
<h1>product List</h1>
<?
  $sql = "select * from products limit 10";
  $rs = mysql_query($sql, Connection::getConnection() );
  while( $row = mysql_fetch_array($rs) ) {
     echo $row["PRODUCTID"] . " : " . $row["PRODUCTNAME"] . "<br>";
  }
?>
<?
  $sql = "select * from products limit 10,5";
  $rs = mysql_query($sql, Connection::getConnection() );
  while( $row = mysql_fetch_array($rs) ) {
     echo $row["PRODUCTID"] . " : " . $row["PRODUCTNAME"] . "<br>";
  }
?>
```

كما هو ملاحظ بالمثال فقد قمنا بتنفيذ جملتين إستعلام علي نفس قاعدة البيانات دون اللجوء إلي فتح أكثر من إتصال إستنادا علي الدالة getConnection الموجودة بالفئة Connection الموجودة بالملف Connection.php وهو مثال بسيط لتوضيح الفكرة فقط , أرجو أن يكون هذا المثال واضح

كيف يمكننا تطبيق نمط Singleton في الإصدار الخامس:php5

تم تحسين هذه النسخة بوضع إمكانية عمل أعضاء ساكنة static members بفئة ما
وبذلك يمكننا تطبيق نمط ال Singleton بشكل أفضل كما يلي سوف نعيد المثال السابق ولكن علي الإصدار الجديدphp5
وسوف نعدل فقط الملفConnection.php

```
<?
class Connection {

  private static $conn = null;

  /**
   * getConnection
   *
   * get only one object refere to link identifier resource
   * to avoid multi-connection open
   *
   * @return link identifier
   */
  public static function getConnection() {

     if( self::$conn == null ) {
        echo "<h1>Singleton Object Created</h1>";
        self::$conn = mysql_connect("localhost","root","");
        mysql_select_db("test", self::$conn);
     }

     return self::$conn;
  }
}
?>
```

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

كما نلاحظ فقد قمنا بتعريف عضو ساكن بالفئة Connection باسم conn كما يلي

```
<?
class Connection {

  private static $conn = null;
```

وهذا المتغير يحتفظ بقيمة دائما أثناء تنفيذ البرنامج (ولا يرتبط بالكائنات التي يتم إنشائها من هذه الفئة , لذلك يطلق علي الأعضاء الساكنه باسم Class Members اي أعضاء الفئة)

الحلقة الثانيةCommand Pattern :

نمط الأوامر أو , Command Pattern يوفر لنا هذا النمط تنفيذ وظيفة ما , Command ويختلف أداء هذه الوظيفة علي حسب طريقة تناولهاimplementation علي سبيل المثال ربما تكون تملك مروحة كهربائية Fan ومصباح كهربائي كلاهما بهما وظيفة الفتح CommandOn ووظيفة الإغلاقCommandOff ولكن عند تنفيذ هذه الأوامر سوف تختلف نتيجة التنفيذ حسب طبيعة الحاله , فلو نفذنا مثلا الأمر CommandOff علي المصباح فسوف ينطفئ النور , ولو تم تنفيذه علي المروحة لتوقفت المروحة عن الدوران , وهكذا

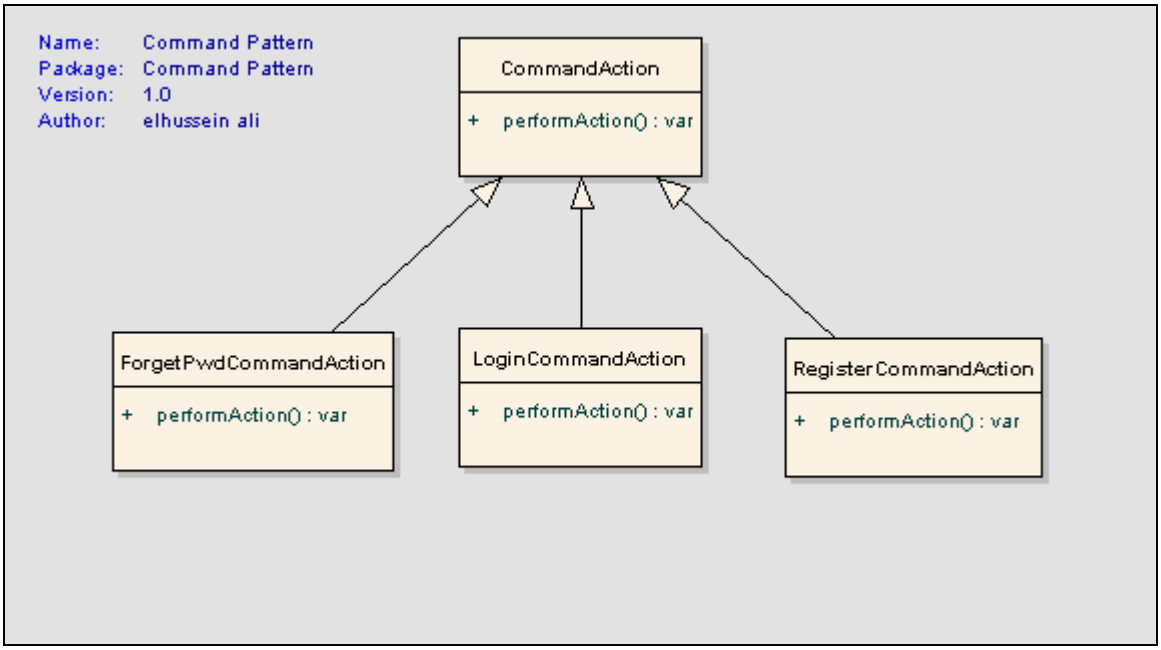
لذلك نخلص مما سبق أننا ربما أثناء إجراء معالجة برمجية لحالة ما , ربما نواجه حالات متشابهه في أوامرها ولكن تختلف في أدائها , لذلك نلجئ في هذه الحالات لتطبيق مفهوم نمط الأوامر Command Pattern

مثال تطبيقي لتوضيح هذا النمط

دعنا نتناول برمجة صفحة تسجيل دخول بسيطة login page حتي تتضح الصورة

معطيات الحالة:
يجب أن توفر هذه الصفحة الأوامر Commands التالية
- التأكد من بيانات تسجيل الدخول LoginCommandAction
- إستعادة كلمة السر ForgetPwdCommandAction
- تسجيل بيانات عضو جديدRegisterCommandAction

العمل:
يمكننا ملاحظة أن هذه المعطيات ما هي إلا أوامر تجمعها وجهة واحدة وهي تنفيذ عملها Action عند حدوث إرسال للبياناتData Submiting لذلك لو أخذنا برهة من التفكير , لوجدنا أن انسب حل لهذه الحالة يتوافق مع نمط الأوامر Command Pattern



الحل:

```
<?
class CommandAction
{
  function performAction() {
     die("performAction method must be implemented in " . get_class($this) . " class");
  }
}

class LoginCommandAction extends CommandAction
{
  function performAction() {
     if( $_POST["loginID"] == "elh"
        && $_POST["pwd"] == "123" ) {
        header("location: welcome.php");
        exit();
     }else {
        echo "<font color=red>sorry try again</font>";
     }
  }

}

class ForgetPwdCommandAction extends CommandAction
{
  function performAction() {
     header("location: ForgetPwd.php");
     exit();
  }

}

class RegisterCommandAction extends CommandAction
{
  function performAction() {
     header("location: register.php");
     exit();
  }

}

// at Post Back
if( count($_POST) > 0 ) {
  $commands = array("login" => "LoginCommandAction",
                    "forgetpwd" => "ForgetPwdCommandAction",
                    "register" => "RegisterCommandAction");

  foreach( $commands as $key => $value) {
     if( $_POST["act"] == $key ) {
        $cmd = new $value();  /* CommandAction Object */

        $cmd->performAction();
        break;
     }
  }

}
?>
```

```
<script language="javascript">

function OnPostBack(action) {
  var actElement = document.getElementById("act");
  actElement.value = action;
}
</script>

<form method="post">
<input type="hidden" id="act" name="act" value="" />

user name : <input type="text" name="loginID" /><br>
password : <input type="password" name="pwd" />

<input type="submit" value="log in" onclick="OnPostBack('login')" />
<input type="submit" value="forget password" onclick="OnPostBack('forgetpwd')" />
<input type="submit" value="register" onclick="OnPostBack('register')" />
</form>
```

كما نلاحظ من المثال السابق فقد قمنا بوضع الأوامر التي تتم في هذه الصفحة في مصفوفة تسمي commands كما يلي

```
  $commands = array("login" => "LoginCommandAction",
                    "forgetpwd" => "ForgetPwdCommandAction",
                    "register" => "RegisterCommandAction");
```

وقد قمنا بعمل فئة باسم CommandAction تكون الفئة الأم base class تحتوي علي وظيفة باسمperformAction
يتم إعادة كتابة الكود بها في كل مرة يتم إشتقاق هذه الفئة بفئة أخري

```
class CommandAction
{
  function performAction() {
     die("performAction method must be implemented in " . get_class($this) . " class");
  }
}
```

ومن المعطيات تم تحديد الأوامر السابقة وتم عمل فئة خاصة لكل أمر علي حدي مشتقة من الفئة الأم CommandAction

```
class LoginCommandAction extends CommandAction
{
  function performAction() {
     if( $_POST["loginID"] == "elh"
        && $_POST["pwd"] == "123" ) {
        header("location: welcome.php");
        exit();
     }else {
        echo "<font color=red>sorry try again</font>";
     }
  }

}

class ForgetPwdCommandAction extends CommandAction
{
```

```
  function performAction() {
     header("location: ForgetPwd.php");
     exit();
  }

}

class RegisterCommandAction extends CommandAction
{
  function performAction() {
     header("location: register.php");
     exit();
  }

}
```

كيف يمكننا تطبيق Command Pattern في الإصدار الخامس:php5

نتيجة التحسينات التي أدخلت علي هذا الإصدار فيمكننا إستبدال الفئة CommandAction بالواجهة ICommandAction وبهذا يكون البناء بشكل أفضل
ويكون المثال كما يلي

```
<?
interface ICommandAction
{
  public function performAction();
}

class LoginCommandAction implements ICommandAction
{
  public function performAction() {
     if( $_POST["loginID"] == "elh"
        && $_POST["pwd"] == "123" ) {
        header("location: welcome.php");
        exit();
     }else {
        echo "<font color=red>sorry try again</font>";
     }
  }

}

class ForgetPwdCommandAction implements ICommandAction
{
  public function performAction() {
     header("location: ForgetPwd.php");
     exit();
  }

}

class RegisterCommandAction implements ICommandAction
{
  public function performAction() {
     header("location: register.php");
     exit();
  }
```

```
}


// at Post Back
if( count($_POST) > 0 ) {
  $commands = array("login" => "LoginCommandAction",
                    "forgetpwd" => "ForgetPwdCommandAction",
                    "register" => "RegisterCommandAction");

  foreach( $commands as $key => $value) {
     if( $_POST["act"] == $key ) {
        $cmd = new $value();  /* CommandAction Object */

        $cmd->performAction();
        break;
     }
  }

}
?>

<script language="javascript">

function OnPostBack(action) {
  var actElement = document.getElementById("act");
  actElement.value = action;
}
</script>

<form method="post">
<input type="hidden" id="act" name="act" value="" />

user name : <input type="text" name="loginID" /><br>
password : <input type="password" name="pwd" />

<input type="submit" value="log in" onclick="OnPostBack('login')" />
<input type="submit" value="forget password" onclick="OnPostBack('forgetpwd')" />
<input type="submit" value="register" onclick="OnPostBack('register')" />
</form>
```

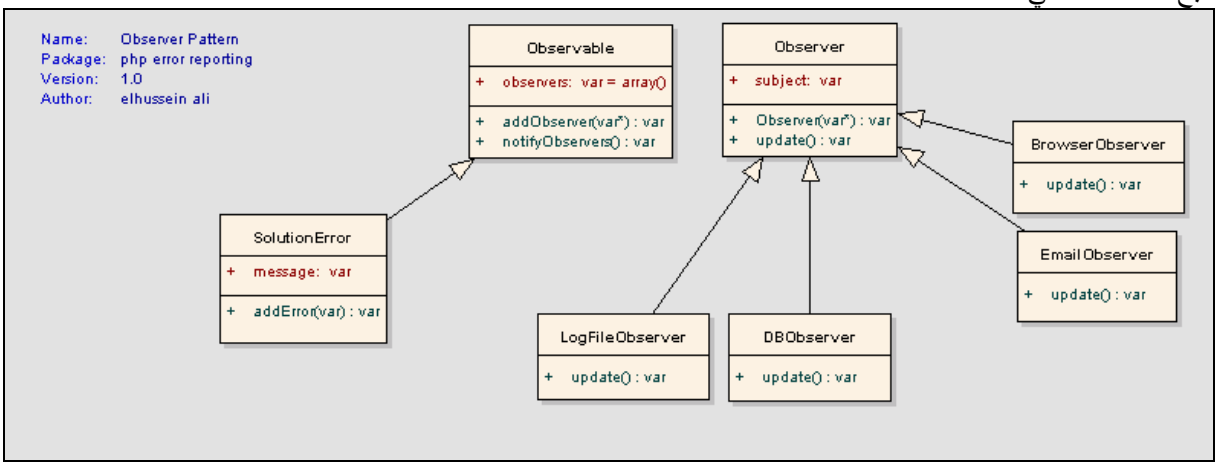
الحلقة الثالثة Observer Pattern : أو نمط المراقب

يعبر هذا النمط عن حالات المراقبه بحيث يمكننا وضع كائن ما تحت المراقبة ويكون هذا الكائن Observable Object اي كائن قابل للمراقبة من خلال
كائن أخر (أو أكثر) يلعب دور المراقب لذلك يطلق علي هذا الكائن Observer اي المراقب

لكن ما هي الحاجة لهذا النمط ؟
بكل صراحة لا أخفي عليكم قولا أن هذا النمط يكون تطبيقة أكثر فاعلية في برامج سطح المكتبdesktop application
إلا أننا يمكننا إستخدامه في الحالات التالية:
-تدوين الأخطأ error reporting
-التغلب علي حالات فقد قاعدة بيانات MySQL لبعض الخصائص مثل المنبهات triggers

مثال تطبيقي بسيط لتوضيح فكرة هذا النمط
نريد رصد الأخطأ الناتجة من البرنامج بحيث يتم تدوينها في التالي:
-ملف متابعة log file
-وأيضا يتم إرسالها إلي البريد الإليكتروني الخاص بك
-وأيضا يتم تسجيل هذا الخطأ في جدول بقاعدة البيانات (اسم الجدول)SolutionErrors
-طباعة هذا الخطأ بالمتصفح

الحل:
تابع المخطط التالي



الجزء الأول : الكائن الذي يتم مراقبتهObservable Object
من المخطط السابق يتضح لنا أننا قمنا بعمل فئة باسم Observable وهي بمثابة الفئة الأم للكائنات التي يتم وضعها تحت المراقبة
وتتميز الفئة Observable بالتالي:
-هي فئة مجردةabstract class
-تحتوي علي مصفوفة باسم observers تحمل الكائنات التي تقوم بمراقبة هذا الكائن
)وكل عنصر بهذه المصفوفة يعبر عن كائن من نوع الفئة الأم Observer كما سنري لاحقا(
-وظيفة باسم addObserver تمكننا من إضافة كائن للمراقبة (Observer Object) في المصفوفةobservers
(وهي وظيفة داخلية internal or package method يتم إستدعائها من خلال كائنات المراقبة)
-وظيفة باسم notifyObservers تقوم باستدعاء الوظيفة update الخاص بكل كائن يقوم بمراقبة هذا الكائن (وهذه الكائنات هي محتوي المصفوفة(observers
(وهي وظيفة خاصة يتم توريثها protected method يتم إستدعائها من خلال الكائن الذي يتم مراقبته)

وهي كالتالي

```
<?php
/**
*  Base Observerable class
*/
class Observable {
  /**
   * private
   * $observers an array of Observer objects to notify
   */
  var $observers = array();

  /**
   * addObserver
   *
   * Register the reference to an object object
   * @return void
   */
  function addObserver( &$observer ) {
     $this->observers[] = &$observer;
  }

  /**
   * notifyObservers
   *
   * Calls the update() function using the reference to each
   * registered observer - used by children of Observable
   * @return void
   */
  function notifyObservers() {
     $observersCount = count($this->observers);

     for($i=0; $i<$observersCount; $i++) {
        $this->observers[$i]->update();
     }
  }

}
?>
```

وكما يتضح من المخطط السابق أننا قمنا بعمل فئة باسم SolutionError ترث هذه الفئة من الفئةObservable
اي أننا يمكننا إجراء مراقبة لكائنات هذه الفئة
وتتميز الفئة SolutionError بالتالي:
-تحتوي علي متغير باسم message يتم تحميله بقيمة رسالة الخطأ
-تحتوي علي الوظيفة addError يتم من خلالها تحميل قيمة رسالة الخطأ , ثم يتم تبليغ الكائنات التي تراقب هذا الكائن بأن هذا الكائن حدث به تغير ما من خلال إستدعاء الوظيفةnotifyObservers

وهي كما يلي

```
<?php
/**
*  Class SolutionError
*/
class SolutionError extends Observable {
   var $message;

  /**
   * addError
   *
   * add error and notifies observers
   * @return void
   */
  function addError($message) {
     $this->message = $message;

     // Call the parent function to notify all observers
     $this->notifyObservers();
  }
}
?>
```

الجزء الثاني : الكائنات المراقبةObservers Object

من المخطط السابق يتضح لنا أننا قمنا بعمل فئة باسم Observer وهي بمثابة الفئة الأم للكائنات التي تلعب دور المراقب وتتميز الفئة Observer بالتالي:

-هي فئة مجردةabstract class

- تحتوي علي متغير باسم subject يعبر عن الكائن الذي يتم مراقبته

-الوظيفة (update وهي وظيفة مجردة abstract method اي يجب إعادة كتابتها في الفئات المشتقة من هذه الفئة) يتم إستدعائها بشكل تلقائي من الكائن الذي يتم مراقبته (من خلال إستدعاء الوظيفة(notifyObservers

وهي كما يلي

```
<?php
/**
*  Base Observer class
*/
class Observer {
  /**
   * private
   * $subject a child of class Observable that we're observing
   */
  var $subject;

  /**
   * Constructs the Observer
   * @param $subject the object to observe
   */
  function Observer( &$subject ) {
     $this->subject = &$subject;

     // Register this object so subject can notify it
     $subject->addObserver($this);
  }

  /**
```

```
  * update
  *
  * Abstract function implemented by children to respond to
  * changes in Observable subject
  * @return void
  */
  function update() {
     trigger_error ('Update not implemented');
  }
}
?>
```

ومن بعد ذلك تمكننا من عمل فئات المراقبة المشتقة من الفئةObserver وهم كالتالي:

1- BrowserObserver

وهذه الفئة خاصة بمراقبة الأخطاء , ثم في حالة وجود خطأ يتم تدوينها بالمتصفح

وهي كما يلي

```
<?php
/**
*  Class BrowserObserver
*/
class BrowserObserver extends Observer {

  /**
   * Implement the parent update() method
   * write the error message into the browser
   * @return void
   */
  function update () {
     echo $this->subject->message;
  }

}
?>
```

2- EmailObserver

وهذه الفئة خاصة بمراقبة الأخطاء , ثم في حالة وجود خطأ يتم إرسال هذا الخطأ إلي البريد الإليكتروني

وهي كما يلي

```
<?php
/**
*  Class EmailObserver
*/
class EmailObserver extends Observer {

  /**
   * Implement the parent update() method
   * send the error message into the administrator email
   * @return void
   */
  function update () {
     @mail("a_elhussein@hotmail.com", "email error observer", $this->subject->message );
  }
}
?>
```

LogFileObserver -3

وهذه الفئة خاصة بمراقبة الأخطاء , ثم في حالة وجود خطأ يتم تدوينها في ملف المتابعة باسمlog.txt
وهي كما يلي

```
<?php
/**
*  Class LogFileObserver
*/
class LogFileObserver extends Observer {

  /**
   * Implement the parent update() method
   * write the error message into the log file
   * @return void
   */
  function update () {
     $fp = @fopen("log.txt" , "ab+");

     fwrite($fp,  date("Y-m-d") . "\t" . $this->subject->message . "\r\n" );
  }

}
?>
```

DBObserver -4

وهذه الفئة خاصة بمراقبة الأخطاء , ثم في حالة وجود خطأ يتم تسجيله في جدول SolutionErrors الموجود بقاعدة البيانات test لمزود قاعدة البياناتmysql
وهي كما يلي

```
<?php
/**
*  Class DBObserver
*/
class DBObserver extends Observer {

  /**
   * Implement the parent update() method
   * add new field into the database table [SolutionErrors]
   * @return void
   */
  function update () {
     // Connect into mysql database server
     $link = @mysql_pconnect("localhost" ,"root" ,"");

     if( is_resource($link) ) {
        // Activate database
        if( @mysql_select_db("test" ,$link) ) {
           $errMessage = str_replace("'", "''", $this->subject->message);
           $sql = "insert into SolutionErrors values(CURDATE(),'$errMessage')";

           mysql_query($sql , $link );
        }
     }
  }
}
?>
```

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

الحلقة الرابعIterator Pattern :
أو نمط الحلقات التكرارية أو نمط إعادة التكرار

تابع معي النقاط التالية
النقطة الأولي:
دائما ما نحتاج عمل حلقة تكرارية Looping باستخدام جمل ال for أو حلقات ال while أوdo while

فلو طلبت منك مثلا أن تقوم بكتابة كود يقوم بطباعة مجموع الأعداد لمصفوفة ما
ربما لأسرعت وقمت باستخدام حلقة تكرارية من نوع for للقيام بذلك
ولكني أري علي الطرف الأخر مبرمج أخر قد قام بكتابة الكود مستخدما جملة while أو ربما جملةdo while
وكلا له فكره الذي يوصله للنتيجة

دعني أعرض عليك أمر أخر , ماذا لو قلت لك أني أريدك أن تقوم بتصميم كود يقوم بسرد أسماء الملفات الموجودة بفهرس ما folder وليكنc:\\windows
ثم قلت لك أني أريدك أيضا أن تقوم بسرد أسماء الفهارس الموجودة بفهرس ما
فربما سنجد من يقوم بعمل حلقة تكرارية مستخدما جملة for والأغلبية سوف يستخدم جملةwhile

ثم هب أنني طلبت منك طلب أخر (وأرجو أن تتحملني بصدر رحب)
أريدك أن تقوم بسرد محتويات جدول ما وليكن اسمه students مرة من قاعدة بيانات من نوع مزود الخدمة MySQL
ومرة من قاعدة بيانات Access ومرة من نوع SqlServer و كمان Oracle ربما سوف تميل لإستخدام جملةwhile

بعد كل هذه الحالات السابقة , ربما يتبادر في أذهننا تسائل هل يوجد طريقة ما لتوحيد وسيلة إجراء هذه المهام ؟؟؟!!!

النقطة الثانية:
تخيل معي الحالة التالية , أنك تعمل كعضو في فريق عمل كبير ربما لا يري أو يعرف بعضهم بعض (وربما تتعرض لهذا يوما ما)
ولنقول أن هذا الفريق مقسم إلي قسمين مثلا هما قسم A وقسم B
وأعضاء الفريق بقسم A هم الأعضاء الأكثر خبرة والمسئولين عن تصميم البنية التحتية للمشروع اي هم المسئولين عن تصميم المكاتب البرمجية التي سوف يستخدمها أعضاء الفريق B لإتمام مهامهم
فهب أنه قد طلب من العضو رقم 3 بالفريق B أن يقوم بتصميم صفحة تقوم بعرض الوظائف الجديدة من جدول باسمjobs
لذلك سوف يقوم هذا العضو بالبحث في وثائق مخطط الدوال والفئات التي قام بتصميمها الفريق, A
عن دالة function تقوم بإرجاع مصفوفة تحمل الوظائف الجديدة , وفعلا نجده بعد وقت قليل قد عثر علي وظيفة باسم getNewJobs تقوم بإرجاع مصفوفة تحمل الوظائف الجديدة , وقد تبقي له أن يقوم بإستدعاء هذه الدالة ثم يقوم بعمل حلقة تكرارية داخل هذه المصفوفة لعرضها بمحتوي الصفحة بنسق , html مممممممم ولكن ماذا بعد

فلو تركتك وقت ما حتي تفكر مع نفسك لوجدت أن العضو الموجود بالفريق A الذي قام بتصميم وبرمجة الدالة getNewJobs قد أدي لإيقاع العضو رقم 3 بالفريق B في ورطة , ما هي هذه الورطه ؟؟؟
تخيل معي أن الجدول المسمي jobs يحتوي علي 30 سجل للوظائف الجديدة
فكما نعلم أو نتوقع أن الكود الموجود بالدالة getNewJobs سوف يقوم بتعريف مصفوفة ما تكون المسئولة عن حمل محتوي الجدول السابق ثم سوف يتم عمل حلقة تكرارية وبالطبع سوف يتم ملء هذه المصفوفة داخل هذه الحلقة التي سوف تتكرر 30 مره , ثم بعد ذلك يتم إرجاع هذه المصفوفة

ثم يأتي العضو رقم 3 (المسكين) القاطن بالفريق B بإستدعاء هذه الدالة ثم يخزن القيمة العائده منها في مصفوفة جديدة وحتي يتمكن من عرض هذه السجلات بشكل منسق يقوم بعمل حلقة تكرارية أخري تلف داخل المصفوفة السابقة
وكم لاحظنا أنه سوف يتم الدوران لعدد 30 مره جديدة لعرض المحتوي بالصفحة لذلك قد تكلفنا في المحصله 60 دوره حتي نعرض 30 سجل فقط وهذه الورطه ربما في بعض الأحيان تكون بسيطة ولكن دائما ما تكون ورطه غير هينه لما يترتب عليها من تكلفه في وقت تنفيذ الكود

ولكن ما هو الحل ؟؟؟
أنتهي.

نقطه جانبية فلسفية (شخصية)

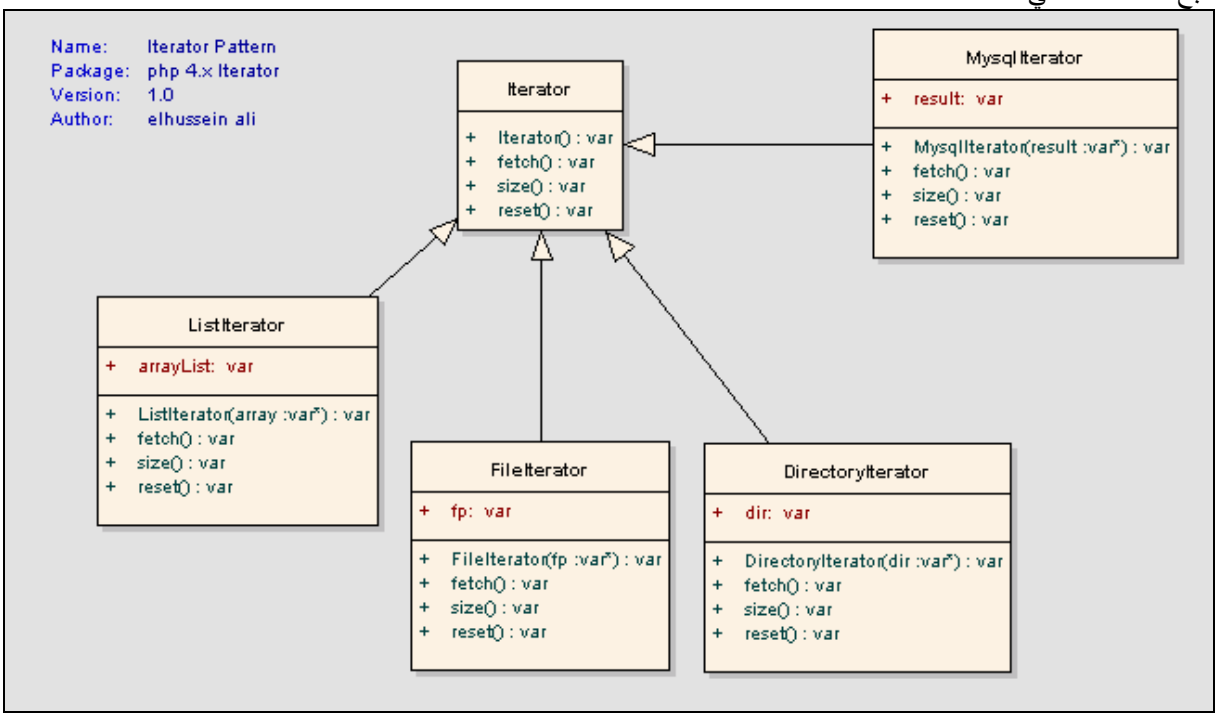
دائما ما تكون الحلول وليدة المشاكل فعندما تواجه في حياتك البرمجية أو حتي الشخصية مشكلة ما , فاعلم أن هذه المشكلة ستكون وليدة لشئ جديد في حياتك.

دعنا من الفلسفة الأن ولنعد سريعا إلي ما قد كنا بدئنا الحديث عنه

نمط Iterator Pattern أو نمط الحلقات التكرارية

تم تصميم فكرة هذا النمط حتي يتيح لنا التعامل بشكل موحد مع المشاكل البرمجية المختلفه التي نحتاج فيها القيام بعمل حلقات تكرارية.

تابع المخطط التالي



الفئة الأم Iterator

يتم في هذا النمط تصميم فئة باسم Iterator تكون بمثابة الفئة الأم لكل الفئات التي نريد أن يتوفر بها هذا النمط وتتميز هذه الفئة بالتالي:

-أنها فئة مجردة abstract class اي لا يمكن أن نقوم بتخليق كائنات منها (إن صح التعبير)

-تحتوي علي الوظائف التالية:

size : +وتقوم هذه الوظيفة بإعلامنا بعدد العناصر للمصفوفة الهدف

reset : +تقوم بإعادة موشر الإنتقال pointer إلي وضع الصفر اي وضع البداية

fetch : +تقوم بإرجاع عنصر واحد فقط من المصفوفة الهدف , ثم تقوم بتحريك مؤشر القراءة إلي الأمام (اي حركه تزايدية بمقدار عنصر واحد)

وهي كما يلي

```
<?
class Iterator
{
  function Iterator() {
     die("Iterator is abstract class");
  }

  /**
```

```
  * Fetches an element from the collection and moves the internal
  * pointer forward one
  *
  * @return variant
  */
  function fetch() {
     die("fetch method must be implemented");
  }

  /**
  * Returns the number of elements in the collection
  *
  * @return int
  */
  function size() {
     die("size method must be implemented");
  }

  /**
  * Resets the collection pointer to the start
  *
  * @return void
  */
  function reset() {
     die("reset method must be implemented");
  }
}
?>
```

أما بالنسبة للإصدار الخامس php5 يفضل أن نقوم بتصميم الفئة الأم Iterator بحيث تكون عبارة عن واجهة interface كما يلي

```
<?
interface Iterator
{
  /**
  * Fetches an element from the collection and moves the internal
  * pointer forward one
  *
  * @return variant
  */
  public function fetch();

  /**
  * Returns the number of elements in the collection
  *
  * @return int
  */
  public function size();

  /**
  * Resets the collection pointer to the start
  *
  * @return void
  */
  public function reset();
}
?>
```

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

ملحوظة : توفر لنا اللغات الحالية مثل الجافا ولغات الدوت نت والإصدار الخامس لل php فئة داخلية تكون بمثابة الفئة الأم لنمطIterator بشكل مختلف قليلا ولكن بنفس الفكر , وربما أتطرق لشرح هذا الأمر لاحقا كتابع لهذا الدرس.

الفئات المشتقةinherited classes

-فئةListIterator وهي فئة خاصة للتعامل مع الحلقات التكرارية الخاصة بالمصفوفات العادية , وهي كما يلي

```
<?
class ListIterator extends Iterator
{
  var $arrayList;

  function ListIterator(&$array) {
     $this->arrayList = &$array;
  }

  function fetch() {
     $element = each($this->arrayList);
     return $element['value'];
  }

  function size() {
     return count($this->arrayList);
  }

  function reset() {
     reset($this->arrayList);
  }
}
?>
```

-فئة MysqlIterator وهي فئة خاصة للتعامل مع الحلقات التكرارية الخاصة بقاعدة بيانات من نوعMysql وهي كما يلي

```
<?
class MysqlIterator extends Iterator
{
  var $result;

  function MysqlIterator( &$result ) {
     $this->result = &$result;
  }

  function fetch() {
     return mysql_fetch_array($this->result);
  }

  function size() {
     return mysql_num_rows($this->result);
  }

  function reset() {
     return mysql_data_seek($this->result,0);
  }
}
?>
```

-فئة FileIterator وهي فئة خاصة للتعامل مع الحلقات التكرارية الخاصة بقراءة محتويات ملف ما وهي كما يلي

```
<?
class FileIterator extends Iterator
{
  var $fp;

  function FileIterator( &$fp ) {
     $this->fp = &$fp;
  }

  function fetch() {
     if( !feof($this->fp) )
        return fgets($this->fp, 4096);
     else
        return false;
  }

  function size() {
     $i=0;

     $this->reset();
     while( $this->fetch() ) {
        $i++;
     }
     $this->reset();
     return $i;
  }

  function reset() {
     fseek ($this->fp, 0);
  }
}
?>
```

-فئةDirectoryIterator وهي فئة خاصة للتعامل مع الحلقات التكرارية الخاصة بالفهارسdirectories وهي كما يلي

```
<?
class DirectoryIterator extends Iterator
{
  var $dir;

  function DirectoryIterator( &$dir ) {
     $this->dir = &$dir;
  }

  function fetch() {
     return $this->dir->read();
  }

  function size() {
     $i=0;
```

```
    $this->reset();
    while( $this->fetch() ) {
      $i++;
    }
    $this->reset();
    return $i;
  }

  function reset() {
    $this->dir->rewind();
  }
}
?>
```

ويمكننا عمل فئات أخري حسب الحاجة لذلك ولكنني أكتفي بهذه الفئات المشتقة لأنها تكفي معظم حاجاتنا

مثال تطبيقي بسيط

نريد تصميم صفحة بسيطة جدا تقوم بسرد محتويات فهرس ما وليكن الفهرس الهدف هو c:\php
الحل إستناد علي الفئات السابقة يمكننا فعل التالي:

DirectoryTest.php

```
<?
require_once "lib/DirectoryIterator.php";

// Modify this to some directory
$dir = dir("c:\\php");
$iterator = new DirectoryIterator($dir);

echo ( "<b>Number of results:</b> ".$iterator->size()."<br />\n" );
while ( $element = $iterator->fetch() ) {
  echo( $element . "<br/>" );
}
$dir->close();
?>
```

مثال تطبيقي بسيط أخر

نريد تصميم صفحة بسيطة تقوم بسرد محتويات ملف ما وليكن الملف التالي c:\php\install.txt

الحل إستناد علي الفئات السابقة يمكننا فعل التالي:
FileIteratorTest.php

```
<?
require_once("lib/FileIterator.php");

// Modify this to point at a real file
$fp=fopen("c:\\php\\install.txt",'r');
$iterator = new FileIterator($fp);

echo ( "<b>Number of Lines:</b> ".$iterator->size()."<br />\n" );
while ( $element = $iterator->fetch() ) {
  echo($element . "<br/>");
}

fclose($fp);
?>
```

مثال تطبيقي بسيط أخر

نريد تصميم صفحة بسيطة تقوم بسرد محتويات جدول ما بقاعدة بيانات من نوع mysql
وليكن الجدول الهدف هو جدول user الموجود بقاعدة البيانات mysql

الحل إستناد علي الفئات السابقة يمكننا فعل التالي:

MysqlIteratorTest.php

```
<?
require_once("lib/MysqlIterator.php");

// Modift these and the query to some database / table of your own
mysql_connect('localhost', 'root', '') or die(mysql_error());
mysql_select_db('mysql') or die(mysql_error());

$sql = "SELECT * FROM user";
$result = mysql_query($sql);
$iterator = new MysqlIterator($result);

echo ( "<b>Number of results:</b> ".$iterator->size()."<br />\n" );
while ( $element = $iterator->fetch() ) {
  echo( $element["Host"] . "<br/>");
}

mysql_free_result($result);
?>
```

أعتقد أننا لاحظنا من الأمثلة السابقة أننا قد وحدنا فكرنا, والأن نقوم بعمل الحلقات التكرارية علي اي هدف بنفس الطريقة,

دعنا الأن نعود لمثال الفريق A والفريق B حتي تزداد الصوره وضوح

مثال تطبيقي أخر

كما قد عرضنا سابقا حالة هذا المثال المتنقل بين الفريقين A و B
وسوف يكون الجدول المراد التعامل معه هو جدول ال Jobs وهو كما يلي

```
#
# Table structure for table 'jobs'
#
CREATE TABLE jobs (
 JobID tinyint(3) unsigned zerofill NOT NULL auto_increment,
 JobTitle varchar(255) default NULL,
 IsNew tinyint(1) unsigned default NULL,
 PRIMARY KEY  (JobID),
 UNIQUE KEY JobID (JobID)
) TYPE=MyISAM;

#
# Dumping data for table 'jobs'
#
INSERT INTO jobs VALUES("001", "مطلوب مبرمج php", "1");
INSERT INTO jobs VALUES("002", "مطلوب مبرمج asp", "1");
INSERT INTO jobs VALUES("003", "0" ,"مطلوب بائع لمحل مجوهرات");
INSERT INTO jobs VALUES("004", "1" ,"مطلوب مضيف ليس حسن المظهر");
```

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

الحل بالنسبة للعضو بالفريق A
سوف يقوم بتصميم دالة باسم getNewJobs تقوم بإرجاع كائن من نوع الفئة Iterator
وفي مثالنا هذا سوف يكون الكائن من نوع الفئة MysqlIterator
وهي كما يلي

Jobs.php

```
<?
require_once("lib/MysqlIterator.php");

function getNewJobs() {
  // Modift these and the query to some database / table of your own
  mysql_connect('localhost', 'root', '') or die(mysql_error());
  mysql_select_db('test') or die(mysql_error());

  $sql = "SELECT * FROM jobs where IsNew=1";
  $result = mysql_query($sql);

  return new MysqlIterator($result);
}
?>
```

ثم يقوم العضو رقم 3 بالفريق B بتصميم الصفحة الخاصة بعرض أحدث الوظائف كما يلي

NewJobsView.php

```
<?
include_once "Jobs.php";

$iterator = getNewJobs();
while ( $element = $iterator->fetch() ) {
  echo( $element["JobTitle"] . "<br/>");
}
?>
```

وهكذا لا يتم عمل مضاعفة في عدد الدورات التي تم التنبيه عنها سابقا

الحلقة الخامسة DataAccess Pattern : أو نمط طبقة التعامل مع البيانات

يتميز هذا النمط بالقيام بعمل طبقة منفصلة **Data Access Layer -- DAL** تقوم بعزل البرنامج عن مصدر البيانات **Data Source**
وتكون وظيفة هذه الطبقة هي
-الإتصال المباشر مع مصدر البيانات (مثلا قاعدة بيانات ما أو ملف نصي عادي plain text أو ملف بنسق xml إلخ.)
-جلب البيانات من مصدر البيانات.
-التعامل مع البيانات المخزنة بمصدر البيانات من حذف وتعديل وإضافة و بحث إلخ.

ولكن ما فائدة هذا النمط

-بناء طبقة منفصلة عن المشروع الذي يتم تطويره تكون مسئولة عن التعامل مع مصدر البيانات
وبالطبع يمكننا إستخدام تلك الطبقة في تطوير مشاريع برمجية أخري , لأن هذه الطبقة ليس لها دور أساسي لمشروع ما بعينه ولكنها تكون بمثابة الوسيط بين المشروع و مصدر البيانات , وبذلك تكون وسيلة جيدة لدمجها للعمل بمشاريع أخري scaleability بالإضافة لفائدتها في تيسير عمل صيانة للمشروع maintenance في النقاط المسئولة عن التعامل مع مصدر البيانات.

-تخيل معي أخي الكريم أمر أخر ,أنك قمت بتصميم موقع ما وقمت ببنائه مستخدما مصدر للبيانات من نوع مزود الخدمة mysql لقواعد البيانات وبعد الإنتهاء من المشروع , لسبب أو لأخر تم تغير نوع مزود الخدمة لقاعدة البيانات إلي Oracle أو Postegre أو SQLServer إلخ
ما الحل إذن لو لم نراعي هذا الأمر , بالطبع سوف نلجأ للمرور بكل صفحة تم برمجتها وإعادة تمكينها للتعامل مع مصدر البيانات الجديد وهذا بالطبع أمر غير صحيح عند تطوير المشاريع.

-بالإضافة إلي أنه سوف تتضح فائدة هذا النمط عندما نشرح نمط MVC Pattern وتكامله مع نمط DataAccess Patternوهو أهم نمط سوف يتم شرحه بإذن الله

مثال تطبيقي بسيط

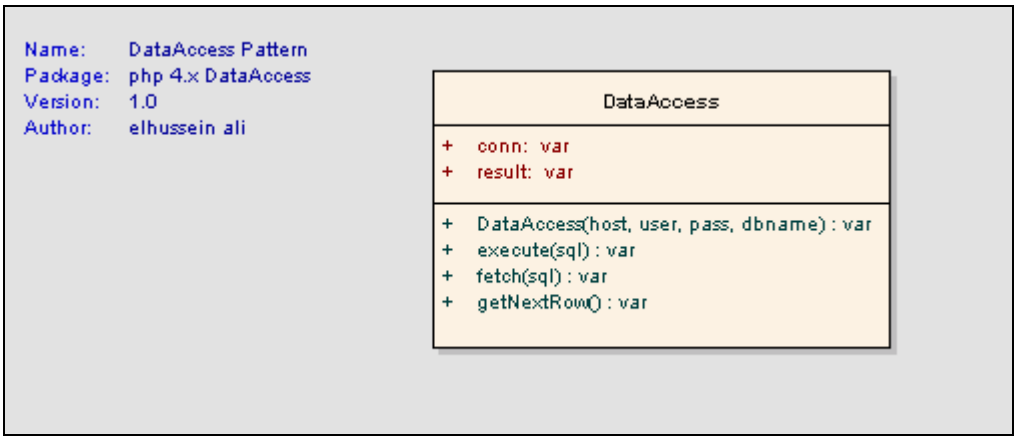
نريد تصميم صفحة باسم Products تقوم بعرض محتوي جدول ال Products التالي الموجود بقاعدة البياناتtest

```
#
# Table structure for table `products`
#
CREATE TABLE products (
 PRODUCTID int(11) NOT NULL auto_increment,
 PRODUCTNAME varchar(255) NOT NULL default '',
 UNITPRICE varchar(255) NOT NULL default '',
 UNITSINSTOCK varchar(255) NOT NULL default '',
 DISCONTINUED varchar(255) NOT NULL default '',
 PRIMARY KEY  (PRODUCTID)
) TYPE=MyISAM;

#
# Dumping data for table `products`
#
INSERT INTO products VALUES (1, 'Chai', '18', '39', '0');
INSERT INTO products VALUES (2, 'Chang', '19', '17', '0');
INSERT INTO products VALUES (3, 'Aniseed Syrup', '10', '13', '0');
INSERT INTO products VALUES (4, 'Chef Antons Cajun Seasoning', '22', '53', '0');
INSERT INTO products VALUES (5, 'Chef Anton Gumbo Mix', '21.35', '0', '1');
INSERT INTO products VALUES (6, 'Grandma Boysenberry Spread', '25', '120', '0');
INSERT INTO products VALUES (7, 'Uncle Bob Organic Dried Pears', '30', '15', '0');
INSERT INTO products VALUES (8, 'Northwoods Cranberry Sauce', '40', '6', '0');
```

الحل

بالطبع الأمر بسيط جدا ولكن هذه المرة لن نقوم ببرمجة هذه الصفحة بشكل تقليدي
ولكننا سوف نقوم بتصميم طبقة خاصة بالتعامل مع قاعدة البيانات(test بمزود خدمة MYSQL)
وتكون وظيفتها التعامل مع قاعدة البيانات فقط , تابع المخطط التالي



بالطبع هذا مخطط بسيط لهذا النمط ويمكننا أن نقوم بزيادة عدد الوظائف function الموجودة بتلك الطبقة
بحيث لا ندع فرصة لمن يقوم ببرمجة المشروع بالتعامل مع مصدر البيانات بطريقة مباشرة دون المرور علي طبقة
التعامل مع البيانات DAL

```
<?php
class DataAccess
{
  /**
  * private
  * stores a database resource
  */
  var $conn;

  /**
  * private
  * stores a query resource
  */
  var $result;

  /**
  * Constucts a new DataAccess object
  * @param $host string hostname for database server
  * @param $user string database server user
  * @param $pass string database server user password
  * @param $dbname string database name
  */
  function DataAccess ($host,$user,$pass,$dbname) {
     $this->conn = mysql_pconnect($host, $user, $pass);
     mysql_select_db($dbname, $this->conn);
  }

  /**
  * Execute a query like insert, delete or update
  * @param $sql string
  * @return void
  */
```

```
  function execute($sql) {
     mysql_query($sql,$this->conn);
  }

  /**
  * Fetches a query resources and stores it in a local member
  * @param $sql string
  * @return void
  */
  function fetch($sql) {
     $this->result = mysql_query($sql,$this->conn);
  }

  /**
  * Returns an associative array of a query row
  * @return Array
  */
  function getNextRow() {
     if( $row = mysql_fetch_array($this->result) )
        return $row;
     else
        return null;
  }
}
?>
```

كما أحب أن أشير إلي أنه يمكننا دمج نمط ال DataAccess مع نمط Iterator الذي تم شرحه سابقا

بعد الإنتهاء من إعداد هذه الطبقة
نقوم باستخدامها في الصفحة المسئولة عن عرض المنتجات التالية

products.php

```
<?
include_once "lib\DataAccess.php";

$dao = new DataAccess("localhost", "root", "", "test");

$dao->fetch("Select * From products");
while( ($curRow = $dao->getNextRow()) != null )
{
  echo $curRow["PRODUCTNAME"] . "<br/>";
}
?>
```

نخلص من المثال السابق بالتالي:
-أن كل تعاملتنا مع قاعدة البيانات تتم من خلال طبقةDataAccess
-يمكننا تزويد هذه الطبقة بمهمة سرد الأخطأ الناتجة عن التعامل مع قاعدة البيانات في ملف متابعة مثلا
-في حالة تم تغير مزود خدمة قاعدة البيانات إلي نوع أخر , يمكننا أن نقوم بتعديل هذه الطبقة فقط دون اللجوء إلي تعديل محتوي المشروع

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

الحلقة السادسة Data Model Pattern : أو Business Logic أو نمط نموذج التعامل مع البيانات

مقدمة:

تعلمنا في الحلقة السابقة (DataAccess Pattern) النقاط التالية :
-كيفية عمل طبقة خاصة للتعامل مع مصدر البيانات Data Access Layer -- DAL
-فائدة عمل طبقة منفصلة تقوم بعزل البرنامج عن مصدر البيانات , Data Source تكون هي المسئولة عن إجراء الإتصال المباشر مع مصدر البيانات

إنطلاق من الحلقة السابقة
دعونا نري ماذا حدث بعدما تمكنا من عمل الطبقة الخاصة للتعامل مع مصدر البياناتDAL
-بالفعل فقد تمكنا من فصل البرنامج الذي نقوم بإعداده عن مصدر البيانات , وأصبحت الطريقة الوحيدة لنا حتي نتمكن من التعامل مع مصدر البيانات هي المرور علي طبقة DAL
وإستخدامها للوصول إلي مصدر البيانات (مثلا قاعدة بيانات , mysql ولكن بهذا لم ينتهي الأمر بعد)
لكن ماذا ينتظرنا بعد ذلك!!!
-دعونا نتخيل الأن أننا نقف أمام فورم Windows Form أو صفحة ويب Web Form تقوم بوظائف ما ومن ضمن هذه الوظائف التعامل مع مصدر البيانات
دعني أتركك تفكر في السؤال التالي : هل تعتقد أننا سوف نقوم بإجراء الوظائف الخاصة بالتعامل مع مصدر البيانات بدون المرور علي طبقة DAL ؟
بالطبع لن نفعل ذلك , لكن هل تعتقد أننا سوف نقوم بإستدعاء الوظائف الموجودة بطبقة DAL من داخل صفحة الويب

بطريقة مباشرة ؟ (Y/N)
+ربما تكون إجابتك بنعم : وربما تشتد علي بالقول قائلا ولما هذا السؤال الذي إستفذني , إذا تمهل معي قليلا ولا تتعجل
+أو ربما تحاول التفكير بشكل جديد وتقول لا , وربما قد دار بذهنك هذا الإستفسار
فعلا لماذا لا نقوم بعمل طبقة جديدة لا تختص بالتعامل مع مصدر البيانات بل تكون المسئولة عن التعامل مع البيانات Data فقط بغض النظر عن مصدر تلك البياناتData Source
ولكن كيف يتم هذا ؟ , إذا تابع معي الحلقة التالية

يختص هذا النمط بعمل طبقة خاصة للتعامل مع البيانات Data فحسب , بغض النظر عن مصدر تلك البيانات
فمثلا دعنا نتخيل الحالة التالية : أننا سوف نقوم بعمل موقع ويب يتعامل مع قاعدة بيانات ما(مصدر بيانات) وتحتوي هذه القاعدة علي جدولين باسم table1 وtable2 فكيف لنا أن نحقق مفهوم هذا النمط في مثل تلك الحالة ؟
- 1سوف نقوم بعمل طبقة خاصة للتعامل مع مصدر البيانات وهي طبقة(DAL كما قد شرحت سابقا)
- 2سوف نقوم بعمل طبقة خاصة للتعامل مع البيانات Data علي النحو التالي:
+سوف نمكن تلك الطبقة من الوصول لمصدر البيانات من خلال المرور علي طبقةDAL
+ثم بعد ذلك نقوم بإعداد نماذج البيانات Data Model أو بمعني أخر الفئات classes الخاصة بالتعامل مع البيانات
فعلي سبيل المثال سوف نقوم بعمل فئة باسم table1 تكون المسئولة عن جلب البيانات من الجدول table1 الموجود بمصدر البيانات , وبالطبع لن تقتصر وظائف تلك الفئة علي جلب البيانات فحسب بل سوف نمكنها من عمل وظائف أخري تتم علي هذا الجدول مثل الحذف والإضافة والتعديل والبحث إلخ حسب الحاجة لذلك
,وهكذا يتم عمل طبقة كاملة من الفئات تكون مسئولة فقط عن التعامل مع البيانات فحسب ولا تقوم بالتعامل مع مصدر البيانات بشكل مباشر إلا من خلال طبقة DAL.

ولكن ما فائدة هذا النمط

كما كنا قد أشرنا أن فكرة هذا النمط تعتمد علي عمل طبقة خاصة للتعامل مع البيانات مما يزيد من كفاءة صيانة البرنامج نتيجة عدم دمج هذه الوظائف داخل البرنامج
كما أنه تخيل معي الأمر التالي , أنك بعدما أنتهيت من تصميم الموقع الذي تقوم ببرمجة مثلا طلب منك أن تحوله إلي برنامج لسطح المكتبDesktop application أعتقد أنك في مثل هذه الحالة لو إتبعت تطبيق هذا النمط فسوف تواجهك فقط مشكلة تحويل النماذج وشاشات الموقع إلي فورم فقط , ثم تقوم بدمج طبقة DAL وطبقة التعامل مع البيانات السابقة بدون تغيرات تذكر
وربما هذه الحالة تكون واضحة أكثر لمبرمجي الجافا والدوت نت.

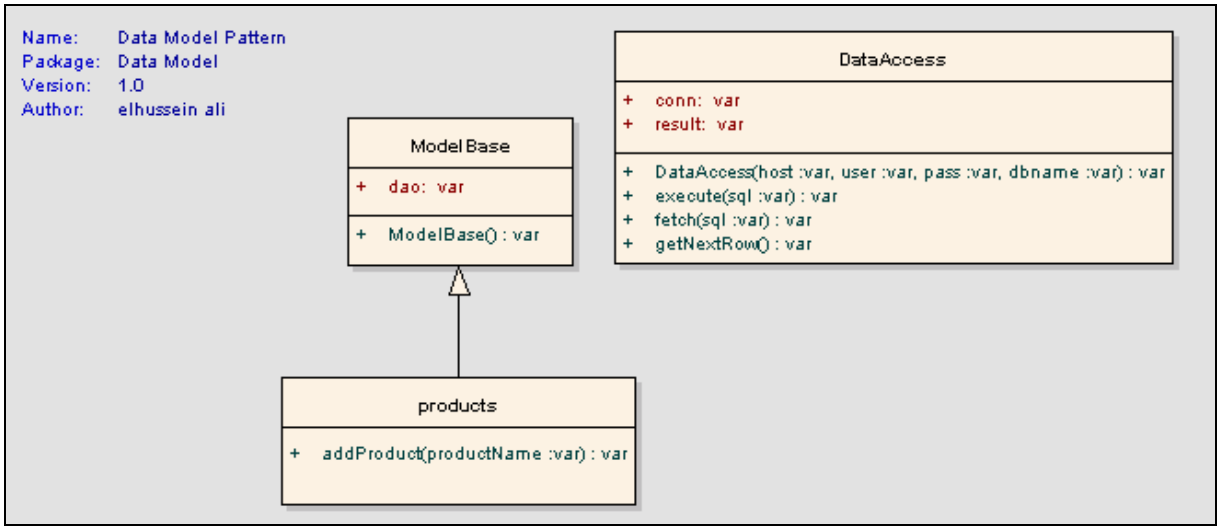
مثال تطبيقي بسيط

نريد تصميم صفحة باسم productPage نقوم من خلالها بتعبئة محتوي جدول ال Products التالي الموجود بقاعدة البيانات test لمزود خدمةmysql

```
#
# Table structure for table `products`
#
CREATE TABLE products (
PRODUCTID int(11) NOT NULL auto_increment,
PRODUCTNAME varchar(255) NOT NULL default '',
PRIMARY KEY  (PRODUCTID)
) TYPE=MyISAM;
```

الحل

سوف نعتمد في حلنا هذا علي طبقة DAL التي تم نقاشها بالحلقة السابقة
لذلك يتبقي لنا إعداد طبقة البيانات ولذلك تابعي معي المخطط التالي



كما يتضح بالشكل السابق فقد قمنا بعمل التالي:
-تم تكوين فئة باسم **ModelBase** وتتميز هذه الفئة بالتالي:
+هي الفئة الأم base class لجميع الفئات التي تندرج داخل الطبقة الخاصة بالتعامل مع البيانات
+تتميز بوجود متغير خاص private field باسم $dao إختصار (Data Access Object وهو كائن يتم إنشائه من الفئةDataAccess
وهي الفئة الموجودة بالطبقة DAL الخاصة بالإتصال المباشر مع مصدر البيانات , وبالطبع يتم إستخدام هذا الكائن من داخل الفئة ModelBase حتي نتمكن من إجراء الإتصال بمصدر البيانات

وهي كما يلي

```
<?php
include_once "DataAccess.php";

/**
*  the Model Base Class
*/
class ModelBase
{
 /**
  * Private
  * $dao an instance of the DataAccess class
  */
  var $dao;

 /**
  * constructor
  */
  function ModelBase() {
     $this->dao = new DataAccess("localhost", "root", "", "test");
  }

}
?>
```

- ثم قمنا بعد ذلك بإعداد أول فئة خاصة بالتعامل مع البيانات وهي الفئة **products** وتتميز بالتالي:
  +هي فئة مشتقة من الفئة الأمModelBase
  +تتميز هذه الفئة بإجراء التعامل مع بيانات الجدول المسمي products ومن ضمن تلك العمليات
  *الوظيفة : addProduct تمكننا من إضافة سجل جديد في جدول المنتجات
  وهكذا يمكننا إنشاء وظائف أخري تتم علي بيانات جدول المنتجات داخل تلك الفئة (مثلا حذف منتج , تعديل أو البحث أو جلب المنتجات الجديدة إلخ(
  وهي كما يلي

```
<?php
include_once "ModelBase.php";

class products extends ModelBase
{
 /**
  * addProduct
  * @param productName string
  * @return void
  */
function addProduct($productName) {
 $sql = "insert into products(productName) values('$productName')";

 $this->dao->execute($sql);
}

}
?>
```

بعد الإنتهاء من إعداد هذه الطبقة
نقوم باستخدامها في الصفحة المسئولة عن تعبئة جدول المنتجات التالية

يمكن نسخ وتوزيع وطباعة هذا الدرس حتى تعم الفائده
ولكن تبقى جميع الحقوق محفوظه للفريق العربي للبرمجة

productPage.php

```
<?php
include_once "lib\products.php";

// at post back
if( count($_POST) > 0 ) {
$productBusObject = new products();
$productBusObject->addProduct( $_POST["productName"] );

echo "okay product added successfully";
}
?>
<form method="post">
enter new product :
product name : <input type="text" name="productName">
<input type="submit" value=" add ">
</form>
```

أرجو من الله أن يكون هذا المثال قد زاد الأمر وضوحا

الحلقة السابعة و الأخيرة MVC Pattern :
Model View Controller Pattern أو نمط النموذج و العرض و الموجه

مقدمة :

قبل الدخول في الحديث عن هذا النمط أعيروني السمع في الخبر التالي
في غضون الشهور السابقة قد طالعتنا وكالة أنباء الأنترنت أن فريق المبرمجون العرب arabteam2000 قد قام بشكل شرعي بتغير شكل ومظهر المنتدي الخاص به إلي تصميم جديد أكثر جاذبية , إنتهي الخبر
أمممممممممم ولكن ربما سوف يتلقي هذا الخبر شخص ما بين إثنين
- أحدهما سوف يعجب بالمظهر الجديد لهذا المنتدي فقط .
- والأخر سوف يحاول التمعن في ما ربما قد يكون حدث من خلف الستار
وربما تبادر في ذهنه بعض الأسئلة التالية :
هل التغير في مظهر وطريقة عرض هذا المنتدي هل يتبعه تأثير علي بيانات ومحتوي المنتدي من مقالات وموضوعات ؟
وبعد تفكير بسيط سوف يتدارك هذا السؤال بتعقيبه , لا طبعا , لن يكون لطريقة العرض تأثير علي البيانات , وبرهان ذلك أن المنتدي مازال محتفظ ببياناته
وربما تبادر إلي ذهنه السؤال الأهم
+هل يمكننا فصل الجزء الخاص بتكوين شكل وإسلوب عرض مشروع ما في طبقة منفصلة , بحيث يمكننا تغير هذه الطبقة فقط عندما نريد تحديث وتغير إسلوب العرض كما حدث بذلك المنتدي ؟
إذا تابع معي الحلقة التالية

يستخدم هذا النمط كفكر في إدراة المشاريع حيث يتم فيه تقسيم المشروع إلي عدة طبقات وهما كما يلي
-طبقة التعامل مع البيانات وقد تم الحديث عن كيفية بناء هذه الطبقة عندما تحدثنا عن نمط Data Model Pattern
-طبقة خاصة بالعرض : View وتختص هذه الطبقة بإسلوب عرض البيانات
-طبقة خاصة بالتوجيه : Controller وتختص هذه الطبقة بتحديد نوع العرض وليس شكل العرض

ولكن ما فائدة هذا النمط

دائما ما نتحدث عن فصل البرنامج إلي عدة طبقات , ولكن ما هي فائدة هذا الفصل أو ما فائدة تلك الطبقات
فإذا تبادر لك هذا الإستفسار , فسوف أطرح عليك قصة صغيرة , يمكنك من بعدها , إستنتاج فوائد هذا النمط بنفسك

قصة صغيرة من تأليفي

بعدما أنتهيت من العمل , قمت بالعودة للمنزل وبعد 10 دقائق وجدت الباب يطرق , وعند فتحي الباب كان هناك أحد أصدقائي المقربين
وبعدما رحبت به , قولت أقدم له حاجة (كرم ضيافه)
فقلت له : ماذا لو قمنا بإحضار كوب أو قدح من الشاي , ثم أتبعته بسؤال أخر بقولي : وكم معلقة من السكر تريدها علي الشاي ؟
وبعد سماعي لإجابته بقوله أحب 3 معلقات سكر (ذي تمام) قمت **بمزج** السكر مع الشاي , وقبل أن أستدير كي أعطي له هذا القدح قلت له تفضل كوب من الشاي يحتوي علي 3 معالق سكر
ولكنه عندئذ قد صدمني بقوله أنا لم أطلب إلا معلقتين من السكر فقط
إذا ما الحل ؟ هل أقوم بزيادة الشاي حتي يتناسب مع الطلب (حتي يتم تقليل نسبة السكر بالشاي) , ولا يعني أقوم بفصل مزيج السكر عن الشاي مرة أخري (طبعا مش هعرف أعملها)
وبعدما إنسرف غاضبا مني , جلست وحدي بأطراف الغرفة أفكر في حل كي أتفادي حدوث تلك المشكلة مرة أخري
وبعد تفكير قلت , أنا مش هوجع دماغي تاني في سؤالي السابق عن عدد قطع أو معالق السكر
طيب أنا سوف أقدم للزائر كوب من الشاي بدون سكر وبجانب هذا الكوب سوف أضع مصدر للسكر يأخذ منه الزائر كما يشاء (ولا أيه رأيكم) وقد راق لي هذا الحل كثيرا,
ولكن بعد ذلك حاولت التمعن لفهم نوع هذه المشكلة , وكيف يمكنني أن أستفيد من ذلك الأمر في مشاكل أخري مشابه ربما تختلف مسمياتها وأشكالها ولكن لو أمعنا النظر لوجدنا أن الأمر كله واحد (فسبحان الواحد)
وسوف أترك لكم هذه المهمة في التفكير.
إنتهت القصة

مثال تطبيقي بسيط

نريد عمل المهام التالية علي جدول المنتجات products التالي

-سرد المنتجات كما بالشكل التالي

Name	Price
Chai	18
Chang	19
Aniseed Syrup	10
Chef Antons Cajun Seasoning	22
Chef Anton Gumbo Mix	21.35
Grandma Boysenberry Spread	25
Uncle Bob Organic Dried Pears	30

-وأمكانية عرض تفاصيل منتج ما كما بالشكل التالي

MVC Pattern Example --- Arabteam2000

show products

Name:Chai

Price:18

In Stock:39

MVC Product Example

تركيب جدول المنتجات كما يلي

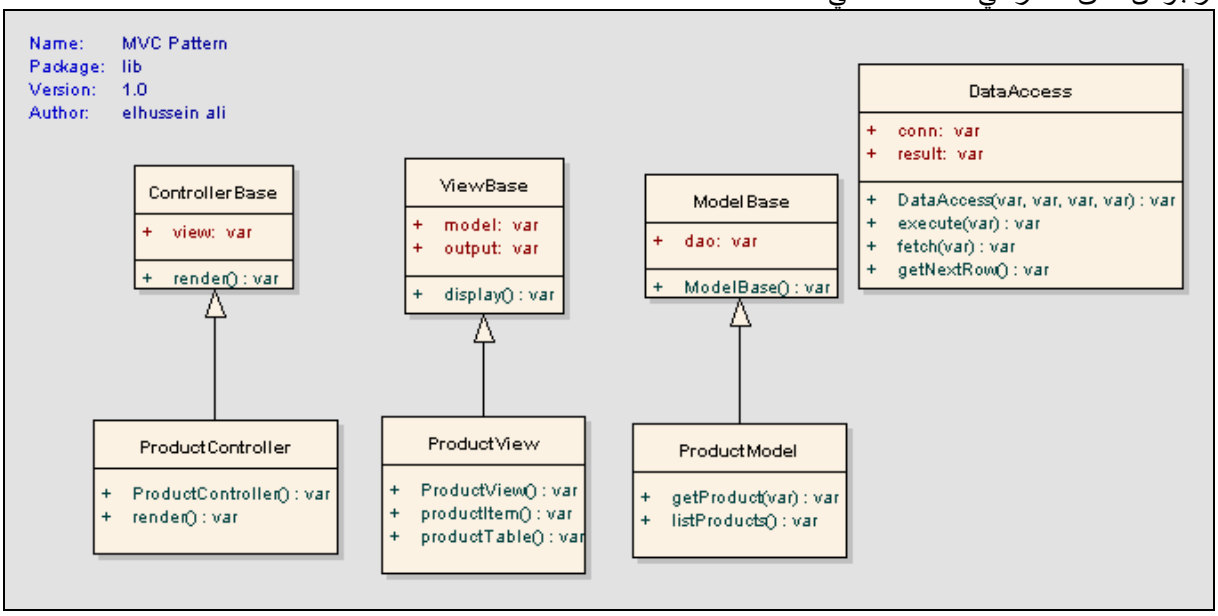
```
CODE

CREATE TABLE products (
 PRODUCTID int(11) NOT NULL auto_increment,
 PRODUCTNAME varchar(255) NOT NULL default '',
 UNITPRICE varchar(255) NOT NULL default '',
 UNITSINSTOCK varchar(255) NOT NULL default '',
 DISCONTINUED varchar(255) NOT NULL default '',
 PRIMARY KEY  (PRODUCTID)
) TYPE=MyISAM;

#
# Dumping data for table `products`
#
INSERT INTO products VALUES (1, 'Chai', '18', '39', '0');
INSERT INTO products VALUES (2, 'Chang', '19', '17', '0');
INSERT INTO products VALUES (3, 'Aniseed Syrup', '10', '13', '0');
INSERT INTO products VALUES (4, 'Chef Antons Cajun Seasoning', '22', '53', '0');
```

```
INSERT INTO products VALUES (5, 'Chef Anton Gumbo Mix', '21.35', '0', '1');
INSERT INTO products VALUES (6, 'Grandma Boysenberry Spread', '25', '120', '0');
INSERT INTO products VALUES (7, 'Uncle Bob Organic Dried Pears', '30', '15', '0');
INSERT INTO products VALUES (8, 'Northwoods Cranberry Sauce', '40', '6', '0');
INSERT INTO products VALUES (9, 'Mishi Kobe Niku', '97', '29', '1');
INSERT INTO products VALUES (10, 'Ikura', '31', '31', '0');
INSERT INTO products VALUES (11, 'Queso Cabrales', '21', '22', '0');
INSERT INTO products VALUES (12, 'Queso Manchego La Pastora', '38', '86', '0');
INSERT INTO products VALUES (13, 'Konbu', '6', '24', '0');
INSERT INTO products VALUES (14, 'Tofu', '23.25', '35', '0');
INSERT INTO products VALUES (15, 'Genen Shouyu', '15.5', '39', '0');
INSERT INTO products VALUES (16, 'Pavlova', '17.45', '29', '0');
INSERT INTO products VALUES (17, 'Alice Mutton', '39', '0', '1');
INSERT INTO products VALUES (18, 'Carnarvon Tigers', '62.5', '42', '0');
```

أرجو أن تدقق النظر في المخطط التالي



ولن أطيل عليكم في سرد طريقة حل هذا المثال , بل سوف أكتفي أن أتركك تفكر وتستنتج بمفردك كيف يمكنك عمل هذا
,أعلم بأني بهذا الشكل قد أكون مبالغ وغير محق أن أتركك هكذا , ولكن
يا شباب كل هذه السلسلة تعتمد علي التخيل ولا تعتمد علي التلقين لذلك أطلق خيالك في التفكير
و عشان محدش يزعل مني ستجدوا الحل في مرفقات هذا الدرس

تم الإنتهاء من شرح حلقات هذه السلسلة بحمد الله وتوفيقه
إلا أني أسئلكم الدعاء لأخيكم عن ظهر الغيب

المراجع التي إستعنت بها
-كتاب(THE DESIGN PATTERNS JAVA COMPANION (JAMES W. COOPER
-بالإضافه لبعض مقالات موقع http://www.phppatterns.com

والسلام عليكم ورحمة الله وبركاته

تم بحمد الله تعالى