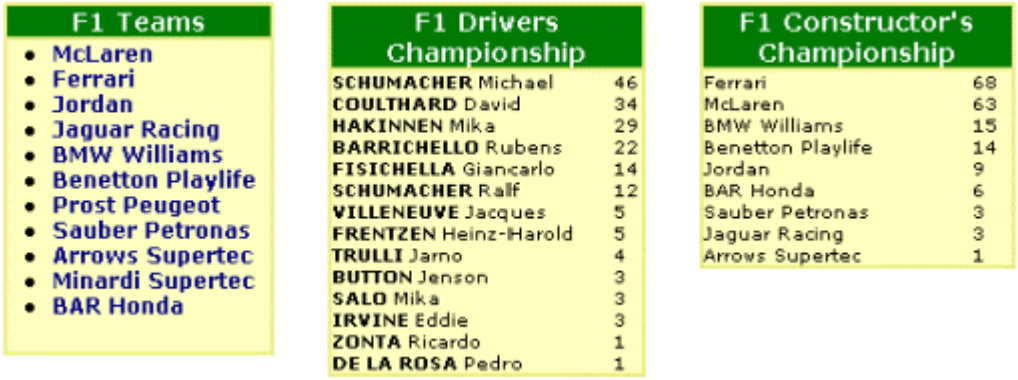
Overview

OK, so you've heard all about it, you think you've got your head around it. Everyone you speak to says it's the way real developers develop. How do you do it? What's in it for you? And how the hell do you do it with PHP?

Broadly speaking, all of the above statements are true, but I guess if your not familiar with OO you want to see a concrete example - and ideally something, which you can fit into your own page(s). Well, here goes.

Let's start off by seeing exactly what we're aiming for.



Basically, we have three boxes (F1 Teams, Drivers Championship and Constructors Championship).

F1 Teams

Quite simply, this is a box with an un-ordered list in it. Each item is a link to the Team's web site

Drivers Championship

This is an informational box, showing the current points standing in the F1 Drivers Championship. There are no links within this box.

Constructors Championship

Like the Drivers Championship box, this is also an informational box, showing the current points standing fo the F1 Constructors Championship.

Now, before you say it, I know you probably don't want details on the current F1 championship, but this does serve as a sample application. At the end of the day, the data is simply taken from a database (in this case I am using MySQL), so you can use whatever data you like. The point is these boxes are "great" portal-type tools for showing lots of focused data. Whether you are a fan of them or not, they do work.

How do we create them?

In my mind there are a number of ways we can create these:

- Hard code the data into a web page. There is nothing wrong with this method if you just want the box(es) to appear on one page, but once you start spreading the across pages, the administrative overhead does become tedious.
- Set the boxes up as a server side include file. Again, there is nothing wrong with this; you can then include the box(es) on any number of pages, simply by referencing the include file.
- Create the boxes as Objects. Initially, this does take a little longer to build, but it does make for portable code (I'm talking about code which walks from site to site, not just page to page!). Additionally, we create interfaces to both the data source and the layout of the box(es), which means we have "easier" control of their data and layout.

As you've probably guessed, we're going with the OO method. What would be the point otherwise?

So, let's start at the top.

Setting the requirements. For the purpose of this exercise, I'm going to define my requirements as:

- Data source must be variable and controllable.
- Layout dimensioning must be variable and controllable.
- Colour must be variable and customisable.
- Font Face must be controllable.

There are a number of ways we can meet these criteria; and for the latter two, it is probably easiest to make use of CSS, which is what we'll do. As, for the first two, so the story begins:

To build this box we are going to use 7 separate files:

- `index.phtml`
- `mysqldb.obj`
- `infobox.obj`
- `linkbox.obj`
- `resultbox.obj`
- `constants.inc`
- `main.css`

Standards

All .obj files are class declarations. I use one file per class/subclass. The .phtml file is the file we are gong to insert our box(es) into. The .inc file is a generic server side include file and the CSS file is exactly that - a Style Sheet.

Development Process

As always, we will start by defining our global constants, then we'll move onto create our data schema (and database), followed by our data class, then our box classes and finally, we'll create the .phtml file, which will bring it all together into a working example - or at least, that's the plan at the moment!

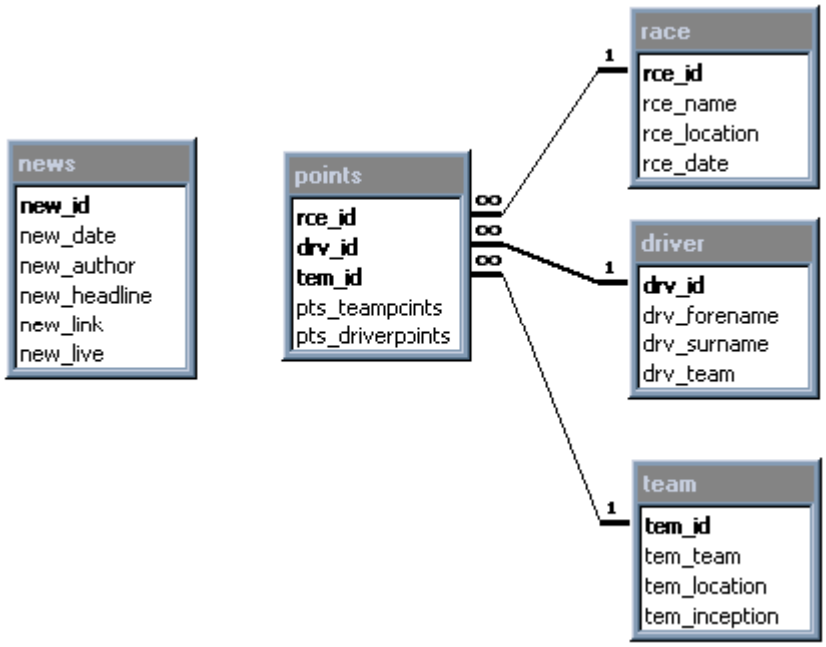
Constants

This is a straightforward PHP file, which I create to define Constant values I can use across a whole site. By using this - appropriately - I can change the look/feel of a site from one place, rather than having to chase variable values across multiple pages. Here's our constants.inc sample:

```
1:    <?
2:    // Database Constants
3:
4:    $HOST               =           "localhost";
5:    $DB                 =           "testing";
6:    $WEBUSER            =           "root";
7:    $WEBPASSWORD        =           "";
8:
9:    // Colour Constants
10:
11:   $COLOR_PRIMARY            =           "#037B0B";
12:   $COLOR_SECONDARY    =           "#FFFFC0";
13:   $COLOR_TERTIARY           =           "#ECED81";
14:
15:   // Value Constants
16:
17:   $TRUE               =           1;
18:   $FALSE              =           0;
19:
20:   // Application Specific
21:   $TITLE              =           "Object Orientation Demonstration";
22:   $ADMINEMAIL         =           "webmaster@domin.com";
23:
24:   // CSS Plug-in Values
25:
26:   $CSSBOXTITLE        =           "boxtitle";
27:
28:   ?>
```

Database Schema

Under no circumstances should this be taken as a de-facto database schema, but it works for the purposes of this example:



As you can see, the data for News is in a table in its own right, whilst the race, driver, team and points details are the result of related tables. Suffice to say, if you do not understand this schema, you should go and read up on some RDBMS introductory material before going any further.

mysqldb Class

So, now we have defined our RDBMS and we know where our data is stored. It's time to start playing around with OO!!!!
If you have played around with PHP for any length of time, you will be well aware there are many functions available for interfacing to a number of RDBMS. Essentially, this class is going to serve as a "wrapper" to some of the MySQL functions - of course, if you are writing for Postgres (or Oracle, or any other RDBMS) you can modify these accordingly.

Here is the code:

```
1:     <?
2:
3:             class mysqldb {
4:
5:                     var $host;
6:                     var $db;
7:                     var $dbuser;
8:                     var $dbpassword;
9:                     var $sql;
10:                    var $numberrows;
11:                    var $dbopenstatus;
12:                    var $dbconnection;
13:
14:            // Property Get & Set
15:
16:                    function gethost() {
17:                           return $this->dbhost;
18:                    }
19:
20:                    function sethost($req_host) {
```

```
21:                         $this->dbhost = $req_host;
22:
23:                 }
24:
25:                 function getdb() {
26:                         return $this->db;
27:                 }
28:
29:                 function setdb($req_db) {
30:                         $this->db = $req_db;
31:                 }
32:
33:                 function getdbuser() {
34:                         return $this->dbuser;
35:                 }
36:
37:                 function setdbuser($req_user) {
38:                         $this->dbuser = $req_user;
39:                 }
40:
41:                 function getdbpassword() {
42:                         return $this->dbpassword;
43:                 }
44:
45:                 function setdbpassword($req_password) {
46:                         $this->dbpassword = $req_password;
47:                 }
48:
49:                 function getsql() {
50:                         return $this->sql;
51:                 }
52:
53:                 function setsql($req_sql) {
54:                         $this->sql = $req_sql;
55:                 }
56:
57:                 function getnumberrows() {
58:                         return $this->numberrows;
59:                 }
60:
61:                 function setnumberrows($req_numberresults) {
62:                         $this->numberesults = $req_numberresults;
63:                 }
64:
65:                 function setdbconnection($req_dbconnection) {
66:                         $this->dbconnection = $req_connection;
67:                 }
68:
69:                 function getdbconnection() {
70:                         return $this->dbconnection;
71:                 }
72:
73:         // Constructor
74:
75:                 function mysqldb() {
76:
77:                         global $HOST, $DB, $WEBUSER, $WEBPASSWORD;
78:                         global $TRUE, $FALSE;
79:
80:                         $this->sethost($HOST);
81:                         $this->setdb($DB);
82:                         $this->setdbuser($WEBUSER);
83:                         $this->setdbpassword($WEBPASSWORD);
84:                         $this->setdbconnection($FALSE);
85:
86:                 }
87:
88:         // Methods
```

```
89:
90:                 function opendbconnection() {
91:
92:                         global $TRUE, $FALSE;
93:
94:                         $this->dbconnection     =     mysql_connect("$this-
>dbhost", "$this->dbuser", "$this->dbuserpassword");
95:                         if ($this->dbconnection == $TRUE) {
96:                                 $this->db = mysql_select_db("$this->db");
97:                                 $this->setdbconnection($TRUE);
98:                         } else {
99:                                 $this->setdbconnection($FALSE);
100:                                return false;
101:                        }
102:                        return true;
103:                }
104:
105:                function closedbconnection() {
106:
107:                        if ($this->dbconnection = $TRUE) {
108:                                mysql_close($this->dbconnection);
109:                        }
110:
111:                }
112:
113:                function selectquery() {
114:
115:                        global $TRUE, $FALSE;
116:
117:                        if ($this->dbconnection == $FALSE) {
118:                                $this->opendbconnection();
119:                        }
120:
121:                        $this->qry = mysql_query($this->sql);
122:                        if (!$this->qry) {
123:                                return false;
124:                        } else {
125:                                $this->numberrows  =  mysql_num_rows($this-
>qry);
126:                                if ($this->numberrows > 0) {
127:                                        for($x = 0; $x < $this->numberrows;
$x++) {
128:                                                $this->result[$x]            =
mysql_fetch_row($this->qry);
129:                                        }
130:                                } else {
131:                                        echo("[Error:] Retrieving data");
132:                                        return false;
133:                                }
134:                                return true;
135:                        }
136:                }
137:
138:        }
139:   ?>
```

Explanation:

Lines 3 - 13 Set up the Object, reserving memory space for variables

Lines 16 - 72 Use these functions to get and set the values of this objects variables. This is good OO practice, as it means that datatype checking can be completed and errors raised accordingly.

Lines 75 - 87 This is the constructor for the object. In this case I have set the initial values of a number of the object properties to those values declared in the global constants.inc. By doing this, I only need to change the values of these properties for specific operations, which we will not need to do throughout this example

Lines 90 - 136 These are the methods for the object. They provide for opening a connection to the database, closing a connection and executing a SELECT query. Of course, these can be expanded upon to allow for INSERT's, UPDATE's and DELETE's etc...

Line 138 Closure of the class declaration
That's it for the data class. Now we need to set up the class's for the actual layout - the boxes!

genericinfo Class

In this particular scenario, we are going to create three classes. That's right, three classes. One for managing the data and two sub-classes for drawing the boxes. Let's start with the managing data class (which we'll call genericinfo). Here's the code:

```
1:     <?
2:
3:             class genericinfo {
4:
5:                     var $outerwidth;
6:                     var $outerbordercolor;
7:                     var $outerborderwidth;
8:                     var $titlebgcolor;
9:                     var $innerwidth;
10:                    var $innerbgcolor;
11:
12:                    // Textual variables
13:                    var $title;
14:
15:                    // Style vairables
16:                    var $cssboxtitle;
17:
18:            // Property Get & Set
19:
20:                    function setouterwidth($req_outerwidth) {
21:                            $this->outerwidth = $req_outerwidth;
22:                    }
23:
24:                    function getouterwidth() {
25:                            return $this->getouterwidth;
26:                    }
27:
28:                    function setouterbordercolor($req_outerbordercolor) {
29:                            $this->outerbordercolor = $req_outerbordercolor;
30:                    }
31:
32:                    function getouterbordercolor() {
33:                            return $this->outerbordercolor;
34:                    }
35:
36:                    function setouterborderwidth($req_outerborderwidth) {
37:                            $this->outerborderwidth = $req_outerborderwidth;
38:                    }
39:
```

```
40:                 function getouterborderwidth() {
41:                       return $this->outerborderwidth;
42:                 }
43:
44:                 function settitlebgcolor($req_titlebgcolor) {
45:                       $this->titlebgcolor = $req_titlebgcolor;
46:                 }
47:
48:                 function gettitlebgcolor() {
49:                       return $this->titlebgcolor;
50:                 }
51:
52:                 function setinnerwidth($req_innerwidth) {
53:                       $this->innerwidth = $req_innerwidth;
54:                 }
55:
56:                 function getinnerwidth() {
57:                       return $this->innerwidth;
58:                 }
59:
60:                 function setinnerbgcolor($req_innerbgcolor) {
61:                       $this->innerbgcolor = $req_innerbgcolor;
62:                 }
63:
64:                 function getinnerbgcolor() {
65:                       return $this->innerbgcolor;
66:                 }
67:
68:                 function settitle($req_title) {
69:                       $this->title = $req_title;
70:                 }
71:
72:                 function gettitle() {
73:                       return $this->title;
74:                 }
75:
76:                 function setcssboxtitle($req_cssboxtitle) {
77:                       $this->cssboxtitle = $req_cssboxtitle;
78:                 }
79:
80:                 function getcssboxtitle() {
81:                       return $this->cssboxtitle;
82:                 }
83:
84:           // Constructor
85:
86:                 function genericinfo() {
87:
88:                       global      $COLOR_PRIMARY,      $COLOR_SECONDARY,
$COLOR_TERTIARY;
89:                       global $CSSBOXTITLE;
90:
91:                       $this->setouterwidth(150);
92:                       $this->setouterbordercolor($COLOR_TERTIARY);
93:                       $this->setouterborderwidth(1);
94:                       $this->settitlebgcolor($COLOR_PRIMARY);
95:                       $this->setinnerwidth(146);
96:                       $this->setinnerbgcolor($COLOR_SECONDARY);
97:
98:                       if (isset($CSSBOXTITLE)) {
99:                             $this->setcssboxtitle($CSSBOXTITLE);
100:                      }
101:
102:                }
103:
104:          // Methods
105:
106:          }
```

Explanation:

Lines 3 - 19 Set up the Object, reserving memory space for variables

Lines 20 - 82 Use these functions to get and set the values of this objects variables. This is good OO practice, as it means that datatype checking can be completed and errors raised accordingly.

Lines 86 - 102 This is the constructor for the object. In this case I have set the initial values of a number of the object properties to those values declared in the global constants.inc. By doing this, I only need to change the values of these properties for specific operations, which we will not need to do throughout this example

Line 106 Closure of the class declaration.

You will no doubt notice from this that there are no methods for the class. Why? Well, I guess in a practical situation you might well choose to combine these three objects into one, simply changing the methods according to the desired box. However, this is a real-life example of using classes and sub-classes ("inheritance") and the reason I have chosen to sub-class is so that as I add more flavours of box (eg, curved corners), I can create further sub-class(es); increasing the portability of the right code.

linkbox Class

This class will take the data, which it is supplied with and then generate the linkbox for us. Essentially, a multi-part array is parsed to the linkbox. This means the array will consist of each piece of data, which is made up of two parts - the bit you want the user to see and the bit, which is the actual URL.

```
1:     <?
2:
3:           class linkbox extends genericinfo {
4:
5:
6:                 function linkbox() {
7:
8:                       $this->genericinfo();
9:                 }
10:
11:                function drawlinkbox() {
12:
13:                      echo("<TABLE    BORDER=\"$this->outerborderwidth\"
CELLPADDING=\"0\"       CELLSPACING=\"0\"       WIDTH=\"$this->outerwidth\"
BORDERCOLOR=\"$this->outerbordercolor\" BGCOLOR=\"$this->titlebgcolor\">");
14:                               echo("<TR>");
15:                                     echo("<TD>");
16:                                           if (isset($this->cssboxtitle))
{
17:                                                 echo("<DIV  CLASS=\""  .
$this->getcssboxtitle() . "\">");
18:                                                       echo($this-
>title);
19:                                                 echo("</DIV>");
20:                                           } else {
```

```
21:                                                     echo($this->title);
22:                                                 }
23:                                         echo("</TD>");
24:                                 echo("</TR>");
25:                                 echo("<TR>");
26:                                         echo("<TD>");
27:                                                 echo("<TABLE      BORDER=\"0\"
CELLPADDING=\"0\"        CELLSPACING=\"0\"        WIDTH=\"$this->innerwidth\"
BGCOLOR=\"$this->innerbgcolor\">");
28:                                                     echo("<TR>");
29:                                                         echo("<TD>");
30:
        echo("<UL>");
31:                                                                         for
($x = 0; $x < count($this->data); $x++) {
32:
        echo("<LI>data[$x][1] . "\">" . $this->data[$x][0] . "");
33:                                                                         }
34:
        echo("</UL>");
35:                                                             echo("</TD>");
36:                                                     echo("</TR>");
37:                                                 echo("</TABLE>");
38:                                         echo("</TD>");
39:                                 echo("</TR>");
40:                         echo("</TABLE>");
41:                 }
42:         }
43:     ?>
```

Explanation:

Lines 3 - 5 Set up the Object. You will notice, we have not reserved an memory space for variables. In this circumstance it is not necessary.

Lines 6 - 9 This is the constructor for the linkbox. The only thing this does is to call the constructor of the parent class. Why? Well, whilst PHP manages a certain part of OO, one of the bits it falls down on (at the moment) is constructors within sub-classes. So, to be sure that the sub-class is instantiated with the constructor of the parent class, I simply call the parent constructor. Of course, if I then wanted to override any of the values, I could easily do so.

Lines 11 - 30 This is the only method within the class. Quite simply, as you can see it draws the table(s), placing the required data in the appropriate place.

Line 42 Closure of the class declaration.

resultbox Class

This is not dissimilar to the linkbox class, except there is not HREF link involved. Instead we create a two-column nested table and place the parsed data into it.

```
1:      <?
2:
3:              class resultbox extends genericinfo {
4:
```

```
5:
6:                  function resultbox() {
7:
8:                        $this->genericinfo();
9:                  }
10:
11:                 function drawresultbox() {
12:
13:                       echo("<TABLE    BORDER=\"$this->outerborderwidth\"
CELLPADDING=\"0\"        CELLSPACING=\"0\"        WIDTH=\"$this->outerwidth\"
BORDERCOLOR=\"$this->outerbordercolor\" BGCOLOR=\"$this->titlebgcolor\">");
14:                               echo("<TR>");
15:                                     echo("<TD>");
16:                                           if (isset($this->cssboxtitle))
{
17:                                                 echo("<DIV  CLASS=\""  .
$this->getcssboxtitle() . "\">");
18:                                                       echo($this-
>title);
19:                                                 echo("</DIV>");
20:                                           } else {
21:                                                 echo($this->title);
22:                                           }
23:                                     echo("</TD>");
24:                               echo("</TR>");
25:                               echo("<TR>");
26:                                     echo("<TD>");
27:                                           echo("<TABLE      BORDER=\"0\"
CELLPADDING=\"0\"        CELLSPACING=\"0\"        WIDTH=\"$this->innerwidth\"
BGCOLOR=\"$this->innerbgcolor\">");
28:                                                 for   ($x   =   0;   $x   <
count($this->data); $x++) {
29:                                                       echo("<TR>");
30:
      echo("<TD>");
31:
      echo($this->data[$x][0]);
32:
      echo("</TD>");
33:
      echo("<TD>");
34:
      echo($this->data[$x][1]);
35:
      echo("</TD>");
36:                                                       echo("</TR>");
37:                                                 }
38:                                           echo("</TABLE>");
39:                                     echo("</TD>");
40:                               echo("</TR>");
41:                         echo("</TABLE>");
42:                 }
43:         }
44:
45:   ?>
```

Explanation:

Lines 3 - 5 Set up the Object. You will notice, we have not reserved an memory space for variables. In this circumstance it is not necessary.

Lines 6 - 9 This is the constructor for the linkbox. The only thing this does is to call the constructor of the parent class. Why? Well, whilst PHP manages a certain part of OO, one of the bits it falls

down on (at the moment) is constructors within sub-classes. So, to be sure that the sub-class is instantiated with the constructor of the parent class, I simply call the parent constructor. Of course, if I then wanted to change any of the values, I could easily do this.

Lines 11 - 42 This is the only method within the class. Quite simply, as you can see it draws the table(s), placing the required data in the appropriate place.

Line 43 Closure of the class declaration.

Index.phtml

How do all these fit together? Well, this is probably best shown by building the page, which we are going to use to present the content to our users, namely index.phtml

```
1:     <?
2:            include "constants.inc";
3:            include "mysqldb.obj";
4:            include "genericinfo.obj";
5:            include "linkbox.obj";
6:            include "resultbox.obj"
7:     ?>
8:     <HTML>
9:            <HEAD>
10:                  <TITLE>
11:                        <? echo($TITLE); ?>
12:                  </TITLE>
13:                  <LINK TYPE="text/css" REL="stylesheet" HREF="main.css">
14:           </HEAD>
15:
16:           <BODY BGCOLOR="#FFFFFF">
17:
18:           <TABLE BORDER="0" CELLPADDING="10" CELLSPACING="10">
19:                  <TR VALIGN="top">
20:                        <TD>
21:           <?
22:
23:                  $db0 = new mysqldb();
24:                  $db0->setsql("SELECT  tem_team,  tem_url  FROM  team  ORDER
BY tem_id");
25:                  if ($db0->selectquery()) {
26:                        $lnk = new linkbox();
27:                        $lnk->settitle("F1 Teams");
28:                        $lnk->data = $db0->result;
29:                        $lnk->drawlinkbox();
30:                  } else {
31:                        echo("[Error:] Unable to connect");
32:                  }
33:
34:           ?>
35:                        </TD>
36:                        <TD>
37:           <?
38:
39:                  $db1 = new mysqldb();
40:                  $db1->setsql("      SELECT
41:
       CONCAT(\"<B>\",UPPER(driver.drv_surname), \" \", driver.drv_forename),
42:                                       SUM(points.pts_teampoints)         as
totdriverpoints
43:                                FROM points
```

```
44:                                     LEFT JOIN driver ON points.drv_id =
driver.drv_id
45:                               GROUP         BY         driver.drv_surname,
driver.drv_forename
46:                               HAVING totdriverpoints <> 0
47:                               ORDER BY totdriverpoints DESC");
48:
49:                  if ($db1->selectquery()) {
50:                        $rst = new resultbox();
51:                        $rst->setouterwidth(175);
52:                        $rst->setinnerwidth(171);
53:                        $rst->settitle("F1 Drivers Championship");
54:                        $rst->data = $db1->result;
55:                        $rst->drawresultbox();
56:                  } else {
57:                        echo("[Error:] Unable to connect");
58:                  }
59:
60:           ?>
61:                        </TD>
62:                        <TD>
63:                  <?
64:
65:                  $db2 = new mysqldb();
66:                  $db2->setsql("      SELECT
67:                                      team.tem_team,
68:                                      SUM(points.pts_teampoints)        as
totteampoints
69:                               FROM points
70:                                      LEFT JOIN team ON points.tem_id =
team.tem_id
71:                               GROUP BY team.tem_team
72:                               HAVING totteampoints > 0
73:                               ORDER BY totteampoints DESC");
74:
75:                  if ($db2->selectquery()) {
76:                        $rst = new resultbox();
77:                        $rst->setouterwidth(175);
78:                        $rst->setinnerwidth(171);
79:                        $rst->settitle("F1 Constructor's Championship");
80:                        $rst->data = $db2->result;
81:                        $rst->drawresultbox();
82:                  } else {
83:                        echo("[Error:] Unable to connect");
84:                  }
85:
86:           ?>
87:                        </TD>
88:                  </TR>
89:           </TABLE>
90:           </BODY>
91:    </HTML>
```

Explanation:

Lines 1 - 7 These lines bring in all the files we have previously written, thereby making their contents available. Notice, how the constants.inc file is the first to be included. If it's not, it's not going to be global!

Lines 8 - 20 Here we break out into good ol' HTML, to setup the page and bring in the style sheet.

Lines 21 - 34 Back into PHP to start using the objects. The first item we are going to create is the F1 Teams box.

Line 23 Instantiates the mysqldb class, creating an object of type mysqldb; to be referenced by $db0. This will therefore execute the constructor function in the mysqldb class, setting initial values to those set by the global constants in constants.inc.

Line 24 Calls the setsql($req_sql) function to set the sql property on the $db0 object.

Line 25 Is an interesting line. Basically, if the call to selectquery() within the mysqldb object fails (or returns as false), execution skips to line 30.

Assuming line 25 returns true, we have the data! Not so bad eh? Now let's work with drawing the box.

Line 26 instantiates the linkbox() class creating an object of type linkbox to be referenced by $lnk. This will therefore execute the constructor function of the linkbox class, which in turn executes the constructor function of the parent class (genericinfo).

Line 27 sets the title of the linkbox, by calling the settitle($req_title) of the parent class.

Line 28 makes sure that the data available to the $lnk is the same as that which is currently in the $db0 object.

Line 29 calls the drawlinkbox() function of the linkbox class. Because $lnk now has direct access to all of the data, it is able to complete the drawing of the linkbox. One point to note in the drawlinkbox function is on lines 16 - 22. If the global constant $CSSBOXTITLE is set, then the style will be included, otherwise it won't.

That's it! You now have the News box drawn on screen!

Lines 35 - 36 Back into HTML to close the table cell and open a new one.

At this points, you should be able to walk through the remainder of the page to see how the other objects are drawn.

main.css

Finally, here is the Style Sheet, which will be used throughout! *Not much more to say on this ;-)*

```
1:     <STYLE>
2:           .bugresolver {
3:
4:           }
5:           TD {
6:                 font-family:        verdana, arial, courier;
7:                 font-size:          10;
8:           }
9:
```

```
10:           P {
11:                 font-family:        verdana, arial, courier;
12:                 font-size:          20;
13:           }
14:
15:           A {
16:                 font-family:        verdana, arial, courier;
17:                 font-size:          12;
18:                 color:              #000084;
19:                 text-decoration:    none;
20:                 font-weight:        bold;
21:                 text-align:         right;
22:           }
23:
24:           A:hover {
25:                 font-family:        verdana, arial, courier;
26:                 font-size:          12;
27:                 color:              #990000;
28:                 background-color:   #DCDADA;
29:                 text-decoration:    none;
30:                 font-weight:        bold;
31:                 text-align:         right;
32:           }
33:
34:           UL {
35:                 margin-left:        25;
36:           }
37:           .boxtitle {
38:                 font-family:        verdana, arial, courier;
39:                 font-size:          14;
40:                 color:              #FFFFFF;
41:                 font-weight:        bold;
42:                 text-align:         center;
43:           }
44:
45:           .lnkBox {
46:                 font-family:        verdana, arial, courier;
47:                 font-size:          10;
48:                 color:              #000084;
49:                 margin-left:        5px;
50:                 text-align:         left;
51:           }
52:     </STYLE>
```

OO Experience

To Those Of You With OO Experience

I am well aware there are a number of "holes" in ths code. For example, I have not completed and data validation when setting and returning property values. However, bear in mind the purpose of this exercise is to give people a real-world" example (which they can follow) of the process and function of OO within PHP.

To Those Of You Without OO Experience

This is a "real-world" example of the process and function of OO within PHP. There are a number of "holes" within the code. That is no to sat the code is incorrect or wrong, but its function is to show you how to work with OO.

If you're looking for further guidance on developing this code into your site, you should revisit many of the get and set functions within these pages and make sure you develop routines to handle the data being parsed; make sure it is of the correct data-type eg it is numeric when it should be and it is a valid SQL statement when it should be). Bear in mind, the data these classes are working with is from a pre-populated database, where I knew the data was valid etc....

For more examples of where these type of boxes are used, please visit http://e-sphere.net, http://f1circle.com and http://markaw.com.

About the Author

Professionally, Mark Williams, is, at the moment, the Senior Technical Consultant to one of Europe's top 10 Insurance/Assurance companies. He is working on a number of Internet/e-commerce initiatives including WAP technology, Video Conferencing, Windows 2000, Linux etc....

To relax, mark is a great F1 fan and on every other Sunday (during the season) you will always find him keeping up to date with the latest events.

More information is available from http://markaw.com, http://e-sphere.net and http://f1circle.com.