Miles To Go Before I Sleep...

By Vikram Vaswani

Table of Contents

Online, All The Time

Web−based calendars are all the rage nowadays. Somehow, there's something very seductive about putting your schedule together on your computer, uploading it to the Web, and accessing it from anywhere on the planet. And with WAP quickly becoming commonplace, you don't even need a Web browser any more − your cellphone has everything you need to connect to your online calendar and make that meeting on time...

Sadly, this tutorial isn't going to teach you how to build a WAP−compatible online scheduler. Nor will it teach you how to synchronize your Palm VII date book with your Yahoo! calendar, your Java−powered wristwatch with your Jini−powered toaster, or any of a million other permutations.

What it will do, however, is offer you some insight into building a simple, extensible and modular PHP−based calendar by combining PHP's date and time functions with a mySQL database. You can then incorporate this calendar module into a larger application, or use it as your personal date book on an Internet Web site or the corporate intranet. Either way, it should offer you some insight into how to apply PHP's capabilities to a real−world problem, and create a simple and elegant solution.

Let's get started!

Building The Foundation

The calendar we're going to build should have both a "month view" and a "day view" − selecting a specific day from the month view should display scheduled appointments for that day. Options should be available to add new appointments to the list, or edit and delete existing appointments.

This is a good time for you to download the source code, so that you can refer to it throughout this tutorial (you will need a Web server capable of running PHP and a mySQL database in order to run the application).

calendar.zip

We'll begin with the month view, a file I'm going to call "month.view.php". The first thing to do is seta few variables which decide the month and year to be displayed − these variables will be used throughout the application, and are crucial to it functioning correctly. PHP's date() function is perfect for obtaining this information:

```
<?
// set up some variables to identify the month, date and year
to display
if(!$currYear) { $currYear = date("Y"); }
if(!$currMonth) { $currMonth = date("n"); }
if(!$currDay) { $currDay = date("j"); }
?>
```

Thus, the variables $currDay, $currMonth and $currYear will hold the values corresponding to the current date, month and year respectively. For example, on 25 January 2001, the variables would look like this:

```
<?
$currYear = 2001;
$currMonth = 1;
$currDay = 25;
?>
```

The date() function comes with numerous modifiers which allow you to extract just those specific segments of the date and time you need from a standard UNIX timestamp. This is a feature I'll be using a lot as I begin building my application, so if you're not familiar with it, take a look at the PHP4 date() function reference at before reading further.

Next, it's time to set up some friendly names for the various months of the year and days of the week − these will come in handy when displaying the calendar.

```
<?
// list of names for days and months
$days = array("Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday",
"Friday", "Saturday");

$months = array("", "January", "February", "March", "April",
"May", "June",
"July", "August", "September", "October", "November",
"December");

// number of days in each month
$totalDays = array(0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
30, 31);

// if leap year, modify $totaldays array appropriately
if (date("L", mktime(0,0,0,$currMonth,1,$currYear)))
{
$totalDays[2] = 29;
}
?>
```

Yup, the date() function even lets you find out if the year under consideration is a leap year − if it is, it's necessary to modify the $totalDays array for the month of February. Since the date() function only works on a correctly−formatted UNIX timestamp, the mktime() function is used to first convert the numeric month and year into an acceptable format.

Before getting into the calendar display, there's one more thing needed: the day of the week on which the first of the month falls.

```
<?
// find out which day the first of the month falls on
$firstDayOfMonth = date("w",
mktime(0,0,0,$currMonth,1,$currYear));
?>
```

The first day of the month (from the $firstDayOfMonth variable) and the last day (from the $totalDays array) provide the bounding values for the month view I'm going to be building.

Seven Days, Seven Nights

Let's move to the calendar display proper:

```
<?
<table border="0" cellpadding="2" cellspacing="5">
<!-- month display -->
<tr>
<td><font face="Arial" size="-2"><<</font></td>
<td colspan="5" align="CENTER"><font face="Arial"
size="-1"><b><? echo
$months[$currMonth] . " " . $currYear; ?></b></font></td>
<td><font face="Arial" size="-2">>></font></td>
</tr>

<!-- day names -->
<tr>
<?
for ($x=0; $x<7; $x++)
{
echo "<td><font face=Arial size=-2>" . substr($days[$x],0,3) .
"</font></td>";
}
?>
</tr>

<!-- start displaying dates -->
<tr>
<?
// display blank spaces until the first day of the month
for ($x=1; $x<=$firstDayOfMonth; $x++)
{
// this comes in handy to find the end of each 7-day block
$rowCount++;
echo "<td><font face=Arial size=-2> </font></td>\n";
}

// counter to track the current date
$dayCount=1;
while ($dayCount <= $totalDays[$currMonth])
{
// use this to find out when the 7-day block is complete and
display a new
row
if ($rowCount % 7 == 0)
{
echo "</tr>\n<tr>\n";
}

// if today, display in different colour

// print date
if ($dayCount == date("j") &$currYear == date("Y") &$currMonth
==
date("n"))
{
echo "<td align=center bgcolor=Silver><font face=Arial
size=−1>" .
$dayCount. "</font>";
}
else
{
echo "<td align=center><font face=Arial size=−1>" . $dayCount
. "</font>";
}

echo "</td>\n";
// increment counters
$dayCount++;
$rowCount++;
}
?>
</tr>

<tr>
<td align=right colspan=7>
<font face=Arial size=−2>
<a href="<? echo $PHP_SELF; ?>">this month</a>
</font>
</td>
</tr>

</table>
?>
```

I'll explain this table row by row. The first row contains "next" and "previous" links (inactive at this point), to allow the user to navigate to the next or previous month of the year, with the name of the current month sandwiched in between.

The next row contains seven cells, one for each day of the week −I've used the substr() function to display the first three letters of each day name from the $days array.

The next few rows are all generated automatically. The first order of business is to place the first day of the month on the corresponding day. Since I already have $firstDayOfMonth variable, I've used a simple loop to fill all the cells prior to that day with non−breaking spaces.

```
<?
// display blank spaces until the first day of the month
for ($x=1; $x<=$firstDayOfMonth; $x++)
{
// this comes in handy to find the end of each 7−day block
$rowCount++;
echo "<td><font face=Arial size=−2> </font></td>\n";
}
?>
```

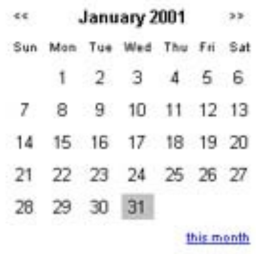
The $rowCount variable is simultaneously keeping track of the number of slots (cells) being filled up − I'll use this a little further down to determine when the end of the week has been reached.

Once the first day of the month is determined, another "for" loop (iterating from 1 to $totalDays[$currMonth]) is used to generate the remaining rows and cells of the table. The $rowCount and $dayCount variables are incremented at each stage, and the $rowCount variable is divided by 7 to find out when the seven slots available in each row are filled up.

```
<?
// counter to track the current date
$dayCount=1;
while ($dayCount <= $totalDays[$currMonth])
{
// use this to find out when the 7−day block is complete and
display a new
row
if ($rowCount % 7 == 0)
{
echo "</tr>\n<tr>\n";
}

// if today, display in different colour

// print date
if ($dayCount == date("j") &$currYear == date("Y") &$currMonth
==
date("n"))
{
echo "<td align=center bgcolor=Silver><font face=Arial
size=−1>" .
$dayCount. "</font>";
}
else
{
echo "<td align=center><font face=Arial size=−1>" . $dayCount
. "</font>";
}

echo "</td>\n";
// increment counters
$dayCount++;
$rowCount++;
}
?>
```

I've inserted an "if" statement into the loop to display the current date in a different colour, if a match is found. And the last row of the table simply contains a link to "this month" − in case you're checking out April 2030 and need a quick way to get back to the present day.

Here's what the result looks like.



<<		January 2001				>>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			
						this month

January To December, And Everything In Between

Now, by itself, this isn't very useful, since it only displays information for the current month (as determined by the server's clock). My next task, therefore, is to make it possible to move forwards and backwards through the year, by activating the "next" and "previous" links on the first row.

In order to do this, I need to define a few new variables, which will be used to identify the previous and next month and year.

```
<?
// set up variables to display previous and next months
correctly

// defaults for previous month
$prevMonth = $currMonth-1;
$prevYear = $currYear;

// if January, decrement year and set month to December
if ($prevMonth < 1)
{
$prevMonth=12;
$prevYear--;
}

// defaults for next month
$nextMonth = $currMonth+1;
$nextYear = $currYear;

// if December, increment year and set month to January
if ($nextMonth > 12)
{
$nextMonth=1;
$nextYear++;
}
?>
```

Note the correction that has to take place if the month in question is either January or December.

Once those variables are defined, it's a simple matter to activate the links on the top row of the calendar. The variables are passed back to the script using the GET method.

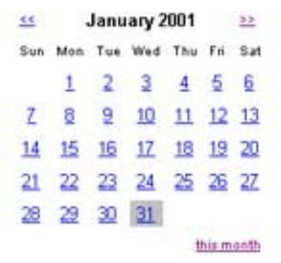
```
<?
<table border="0" cellpadding="2" cellspacing="5">
<!-- month display -->
<!-- this is the first row of the calendar, with links active
-->
<tr>
<td><a href="<? echo $PHP_SELF; ?>?currMonth=<? echo
$prevMonth;
?>? echo $prevYear; ?>"><font face="Arial"
size="-2"><<</font></a></td>

<td colspan="5" align="CENTER"><font face="Arial"
size="-1"><b><? echo
$months[$currMonth] . " " . $currYear; ?></b></font></td>

<td><a href="<? echo $PHP_SELF; ?>?currMonth=<? echo
$nextMonth;
?>? echo $nextYear; ?>"><font face="Arial"
size="-2">>></font></font></a></td>
</tr>
?>
```

You should now have a calendar capable of displaying information for any month of any year.

The next task is to convert each date on the calendar into an active link, which, when clicked, will display the user's current appointments for that day, together with the option to add new appointments. This intelligence will be built into a file called "day.view.php", which requires three parameters − the year, month and date under consideration. So let's make that modification to the code above:

```
<?
// if today, display in different colour

// print date, each date is now an active hyperlink
if ($dayCount == date("j") &$currYear == date("Y") &$currMonth
==
date("n"))
{
echo "<td align=center bgcolor=Silver><font face=Arial
size=-1><a
href=day.view.php?currYear=" . $currYear . ". $currMonth .
". $dayCount . ">" . $dayCount. "</a></font>";
}
else
{
echo "<td align=center><font face=Arial size=-1><a
href=day.view.php?currYear=" . $currYear . ". $currMonth .
". $dayCount . ">" . $dayCount . "</a></font>";
}
?>
```

All done? Here's what it looks like.



<< January 2001 >>

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

this month

Let's move on to the "day view".

Bringing In The Database

Since "day.view.php" is going to read the appointment list from a database, this is a good time to set up the table which will hold calendar data. Here's the structure I came up with − feel free to modify it to your requirements, but remember to alter the SQL queries as well.

```
# −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
#
# Table structure for table 'calendar'
#
CREATE TABLE calendar (
id int(10) unsigned NOT NULL auto_increment,
date date DEFAULT '0000−00−00' NOT NULL,
time time DEFAULT '00:00:00' NOT NULL,
comment text NOT NULL,
PRIMARY KEY (id)
);

# Column descriptions:
#
# id − unique identifier for each entry
# date − appointment date
# time − appointment time
# comment − appointment description
#
```

You might be wondering why I've split the date and time fields into two columns, rather than a single field. Keep reading − you'll see the reason soon enough.

Since I'll be connecting to the database quite frequently, and since I'm pretty lazy and dislike typing in more code than I have to, I've also created a single file, "config.php", which holds the mySQL user name, password and database name. This file is include()d whenever required to open a database connection.

```
<?
// config.php
$server = "localhost";
$user = "us564";
$pass = "he3423k4j";
$db = "calendar";
?>
```

Let's now move on to the "day.view.php" script. As you saw on the previous page, "day.view.php" receives the date, month and year via the URL GET method; it will then use these three variables within a SELECT query to find out if there are any previously scheduled appointments for that day.

```
<?
// format date for entry into database
$this_date = $currYear . "−" . sprintf("%02d", $currMonth) .
"−" .
sprintf("%02d", $currDay);
?>
<table border="0" cellpadding="2" cellspacing="5">

<tr>
<td colspan=2 align=center>
<font face=Arial size=−1>
<b><? echo date("D M d Y",
mktime(0,0,0,$currMonth,$currDay,$currYear));
?></b>
</font>
</td>
</tr>

<?
include("config.php");

// open a connection to the database
$connection = mysql_connect($server, $user, $pass);

// formulate the SQL query − same as above
$query = "SELECT * from calendar WHERE date='$this_date' ORDER
BY time";

// run the query on the database
$result = mysql_db_query($db,$query,$connection);
?>
```

If you take a look at the table structure above, you'll see that the date and time fields require entry in a specific format − so the first order of business is to take the three variables passed to "day.view.php" and format them to match that format with sprintf(). So

```
<?
$currYear = 2001;
$currMonth = 1;
$currDay = 25;
?>
```

becomes

```
2001-01-25
```

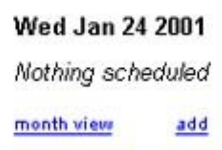
Next, I've opened up a database connection and executed a query to find out if any appointments have been scheduled for that date. Depending on the result, I'll either display a list of appointments, or a message with the words "Nothing scheduled".

```
<?
// run the query on the database
$result = mysql_db_query($db,$query,$connection);

// if result
if(mysql_num_rows($result) > 0)
{
?>
<tr>
<td align=center>
<font face=Arial size=-1>
<i>Time</i>
</font>
</td>
<td align=left>
<font face=Arial size=-1>
<i>Description</i>
</font>
</td>
</tr>
<?
// get the list of appointments
while($row = mysql_fetch_array($result))
{
$this_time = $row["time"];
$comment = $row["comment"];
$id = $row["id"];
?>
<tr>
<td align=center valign=top><font face=Arial size=−1><? echo
substr($this_time,0,5); ?></font> </td>
<td align=left valign=top width=200><font face=Arial
size=−1><? echo
$comment; ?></font>  <font face=Arial size=−2><a
href="edit.php?id=<? echo $id; ?>? echo $currYear;
?>? echo $currMonth; ?>? echo $currDay;
?>">edit</a></font>  <font face=Arial size=−2><a
href="delete.php?id=<? echo $id; ?>? echo $currYear;
?>? echo $currMonth; ?>? echo $currDay;
?>">delete</a></font></td>
</tr>
<?
}
// close connection
mysql_close($connection);
}
else
{
?>
<tr>
<td align=center colspan=2>
<font face=Arial size=−1>
<i>Nothing scheduled</i>
</font>
</td>
</tr>
<?
}
?>
<tr>
<td align=left>
<a href="month.view.php?currYear=<? echo $currYear; ?>? echo
$currMonth; ?>? echo $currDay; ?>"><font face=Arial
size=−2>month
view</font></a>
</td>
<td align=right>
<a href="add.php?currYear=<? echo $currYear; ?>? echo
$currMonth; ?>? echo $currDay; ?>"><font face=Arial
size=−2>add</font></a>
</td>
</tr>
```

```
</table>
?>
```

Each entry (if there is one) is displayed with an "edit" and "delete" link next to it − these point to the "edit.php" and "delete.php" files respectively. Once the appointment list has been displayed, I've added two links at the bottom − one takes you back to "month view", while the other allows you to add a new appointment.

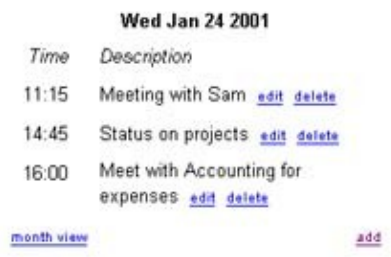
Here's what the result looks like.



Wed Jan 24 2001

Nothing scheduled

month view add



Wed Jan 24 2001

Time	*Description*
11:15	Meeting with Sam edit delete
14:45	Status on projects edit delete
16:00	Meet with Accounting for expenses edit delete

month view add

Adding a new appointment is accomplished with "add.php", which again receives the date, month and year as GET parameters. Let's take a closer look at it next.

Adding, Editing, Deleting...

The file "add.php" is actually split into two sections, one to display the initial form and the second to process form data. The $submit variable is used to identify which section to display at a given time.

```
<?
// form not yet submitted
if (!$submit)
{
// format date
$this_date = $currYear . "-" . sprintf("%02d", $currMonth) .
"-" .
sprintf("%02d", $currDay);
?>
<html>
<head>
</head>

<body>
<table border="0" cellpadding="2" cellspacing="2">
<form action="<? echo $PHP_SELF; ?>" method="get">

<tr>
<td colspan=2 align=center>
<font face=Arial size=-1>
<b><? echo date("D M d Y",
mktime(0,0,0,$currMonth,$currDay,$currYear));
?></b>
</font>
</td>
</tr>

<tr>
<td>
<font face=Arial size=-1>
<i>Time</i>
</font>
</td>
<td>
<select name="hh">
<?
// drop-downs for the date and time
for ($x=0; $x<=23; $x++)
{
echo "<option value=\"" . sprintf("%02d", $x) . "\">" .
sprintf("%02d",
$x) . "</option>\n";
}
?>
</select>

<select name="mm">
<?
for ($x=0; $x<=59; $x++)
{
echo "<option value=\"" . sprintf("%02d", $x) . "\">" .
sprintf("%02d",
$x) . "</option>\n";
}
?>
</select>

</td>
</tr>
<tr>
<td>
<font face=Arial size=-1>
<i>Description</i>
</font>
</td>
<td><input type="Text" name="comment" size="15"></td>
</tr>
<input type=hidden name=this_date value=<? echo $this_date;
?>>
<input type=hidden name=currYear value=<? echo $currYear; ?>>
<input type=hidden name=currMonth value=<? echo $currMonth;
?>>
<input type=hidden name=currDay value=<? echo $currDay; ?>>
<tr>
<td colspan=2 align=center>
<input type="Submit" name="submit" value="Add Entry">
</td>
</tr>
</form>
</table>

</body>
</html>
<?
}
else
{
// form data is processed here
}
```

```
?>
```

There's nothing very challenging here. If the $submit variable doesn't exist, a basic form is displayed with drop−down lists for the appointment time, and a text field for an appointment description. In order to simplify the form processing code, a number of hidden values are also passed with the visible data. PHP code is used to generate the drop−down lists, since it's faster to use a "for" loop than to create sixty <OPTION> tags (remember, you're dealing with an incredibly lazy programmer here!)

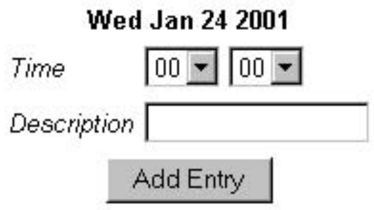
Once the user submits the form, the script is called again, but this time, the second half of the code is executed.

```
<?
// form not yet submitted
if (!$submit)
{
// form
}
else
{
include("config.php");
// format time
$this_time = $hh . ":" . $mm . ":00";

// set up default description
if ($comment == "") { $comment = "Not available"; }

// open a connection to the database
$connection = mysql_connect($server, $user, $pass);

// formulate the SQL query − same as above
$query = "INSERT into calendar VALUES (NULL, '$this_date',
'$this_time',
'$comment')";

// run the query on the database
// assume the database is named "php101"
$result = mysql_db_query($db,$query ,$connection);

// close connection
mysql_close($connection);

header("Location: day.view.php?currYear=" . $currYear . ".
$currMonth . ". $currDay);
}
?>
```

Once the form is submitted, the data entered into it is formatted as per the data structures in the database, a query is generated, and the data is added to the database (note how "config.php" is include()d before connecting to the database to set up the necessary variables).

In case the comment field is empty, the words "Not available" are used as a default comment string. Finally, the header() function is used to redirect the browser back to the "day view" page, which displays an updated appointment list.

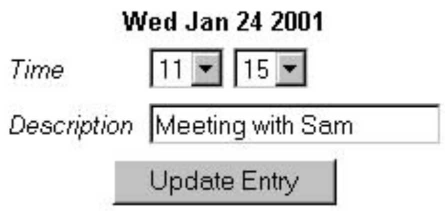
Here's what the result looks like.



The files "edit.php" and "delete.php" are similar − the primary difference lies in the query strings.

```
<?
// from edit.php
$query = "UPDATE calendar SET time='$this_time',
comment='$comment' WHERE
id='$id'";

// from delete.php
$query = "DELETE FROM calendar WHERE id='$id'";
?>
```

The "edit.php" script also contains some extra code, used to pre−fill the form with the details of the appointment. If you take a look at the script, you'll see that, before the form is displayed, a query is generated to obtain the appointment time and description, and this information is then used to pre−fill the text box and pre−select the hour and minute from the drop−down lists.



Wed Jan 24 2001

Time	11 15
Description	Meeting with Sam

Here's the code to accomplish this:

```
<?
// form not submitted
if (!$submit)
{
?>
<html>
<head>
</head>

<body>
<?
// get data for this appointment to pre-fill form

include("config.php");

// open a connection to the database
$connection = mysql_connect($server, $user, $pass);

// formulate the SQL query - same as above
$query = "SELECT * from calendar where id='$id'";

// run the query on the database
$result = mysql_db_query($db,$query ,$connection);

while($row = mysql_fetch_array($result))
{
$this_time = explode(":", $row["time"]);
$this_date = $row["date"];
$comment = $row["comment"];
$hh = $this_time[0];
$mm = $this_time[1];
}
// close connection
mysql_close($connection);
?>
<!-- table code - snipped out à

<tr>
<td><font face=Arial size=-1><i>Time</i></font></td>
<td>
<select name="hh">
<?
// drop-down boxes for hour and time
for ($x=0; $x<=23; $x++)
{
$output = "<option value=\"" . sprintf("%02d", $x) . "\"";

// pre-select as per data
if (sprintf("%02d", $x) == $hh)
{
$output .= " selected";
}
$output .= ">" . sprintf("%02d", $x) . "</option>\n";
echo $output;
}
?>
</select>

<select name="mm">
<?
// and similar code to pre-select the minute
?>
</select>

<!-- more table code - snipped -->

</body>
</html>
<?
}
else
{
// what happens when the form is submitted
}
?>
```

The Final Touch

There's one more thing you can do to make this calendar a little more useful − display an indicator in "month view" to identify which days already have appointments scheduled, and which days are completely free of appointments. In order to do this, go back to "month.view.php" and add the following lines of code to it, somewhere near the beginning of the script (but after you've defined $currDay, $currMonth, and $currYear):

```
<?
include("config.php");

// open a connection to the database
$connection = mysql_connect($server, $user, $pass);

// formulate the SQL query − same as above
$query = "SELECT DISTINCT date from calendar where date >= '"
. $currYear .
"−" . sprintf("%02d", $currMonth) . "−01' and date <= '" .
$currYear . "−"
. sprintf("%02d", $currMonth) . "−" . $totalDays[$currMonth] .
"'";

// run the query on the database
$result = mysql_db_query($db,$query ,$connection);

$x=0;
$dateList=array();
if(mysql_num_rows($result) > 0)
{
while($row = mysql_fetch_array($result))
{
$dates = explode("−", $row["date"]);
$dateList[$x] = $dates[2];
$x++;
}
}
// close connection
mysql_close($connection);
?>
```

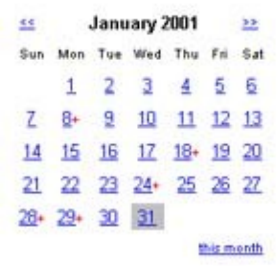
What have I done here? I've used the three variables to formulate a query which returns a list of all the days in $currMonth which already have appointments scheduled (the DISTINCT keyword helps to eliminate duplicate entries). Then I've taken each of those date strings (in the form YYYY−MM−DD), split them into separate entities, and created an array called $dateList which contains a list of all the days on which appointments are scheduled.

The plan is to add an additional check to the sections of code responsible for generating the dates in the month, such that dates which match the elements in the $dateList array have an additional identifier to indicate that something is already scheduled for that day.

You saw earlier that I had separated the date and time into separate columns when creating the database table. One of the primary reasons behind this was to simplify the task of obtaining a list of dates which had one or more appointments scheduled. If the date and time had been combined into a single column, the DISTINCT keyword would have failed to eliminate duplicate entries, and I would have had to write a lot more code to weed out the duplicates. And we already know how lazy I am...

```
<?
// counter to track the current date
$dayCount=1;
while ($dayCount <= $totalDays[$currMonth])
{
// use this to find out when the 7−day block is complete and
display a new
row
if ($rowCount % 7 == 0)
{
echo "</tr>\n<tr>\n";
}

// if today, display in different colour

// print date
if ($dayCount == date("j") &$currYear == date("Y") &$currMonth
==
date("n"))
{
echo "<td align=center bgcolor=Silver><font face=Arial
size=−1><a
href=day.view.php?currYear=" . $currYear . ". $currMonth .
". $dayCount . ">" . $dayCount. "</a></font>";
}
else
{
echo "<td align=center><font face=Arial size=−1><a
href=day.view.php?currYear=" . $currYear . ". $currMonth .
". $dayCount . ">" . $dayCount . "</a></font>";
}

// newly−added code to find out is appointment is already
scheduled
// and print indicator if so
for ($y=0; $y<sizeof($dateList); $y++)
{
//echo $dateList[$y];
if ($dateList[$y] == $dayCount)
{
echo "<font face=Arial color=red size=-4>+</font>";
}
}

echo "</td>\n";
// increment counters
$dayCount++;
$rowCount++;
}
?>
```

And here's what the finished product looks like:



<<			January 2001			>>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8+	9	10	11	12	13
14	15	16	17	18+	19	20
21	22	23	24+	25	26	27
28+	29+	30	31			
						this month

And that's about it. You can now begin using this calendar, as is, for keeping track of your life; modify it as per your requirements; or use it as an entry point to other applications. And if you're new to PHP, this tutorial should hopefully have offered you some insight into how Web applications are developed, and maybe even sparked some ideas of your own. If so, let me know...and till next time, stay healthy!