Using PEAR Date, converting between time zones in PHP

Introduction

PHP comes with an extensive catalog of date and time functions, all designed to let you easily retrieve temporal information, massage it into a format you require, and either use it in a calculation or display it to the user. However, if you'd like to do something more complicated, things get much, much messed up.

A simple example of this involves displaying the time on a Web page. With PHP, you can easily use the date() function to read the server's clock and display the required information in a specific format. But what if you'd like to display the time in a different location - for example, if your company is located in a different country from your server and you want to see "home" time instead of local time? Well, then you have to figure out the difference between the two places and perform some date arithmetic to adjust for the different time zones. If the time difference is significant, you need to take account of whether the new time is on the day before or after, worry about daylight savings time, and keep track of end-of-the-month and leap year constraints.

As you can imagine, the math to perform such time zone conversions can quickly get very complicated if you do it manually. To be fair, PHP has built-in time zone functions to help with this, but these aren't particularly intuitive and require a fair amount of time to get used to. A quicker alternative is to use the PEAR Date class, which comes with built-in support for time zones and is, by far, the simplest way to perform these conversions.

This article will teach you how to convert temporal values between time zones with the PEAR Date class. It assumes that you have a working Apache and PHP installation and that the PEAR Date class has been correctly installed.

Note: You can install the PEAR Date package directly from the Web, either by downloading it or by using the instructions provided.

Getting started

Let's begin with the basics - initialising and using a Date object. Create a PHP script with the following lines of code:

Code: PHP

```php
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("1951-09-13 16:55:11");

// retrieve date
echo $d->getDate();
?>
```

This is fairly simple - include the class code, initialise a Date() object with a date/time string, and then use the getDate() method to display the value you just inserted. Here's the output:

```
1951-09-13 16:55:11
```

What if you want the date in a different format? If the format is a standard one, such as the ISO format, simply pass getDate() a modifier indicating this:

Code: PHP

```php
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("1951-09-13 16:55:11");
```

```
// retrieve date as timestamp
echo $d->getDate(DATE_FORMAT_ISO_BASIC);
?>
```

The output in this case conforms to the standard ISO format.

```
19510913T165511
```

If you'd like a custom format, you can do that too, with the format() method. Like PHP's native date() function, this method accepts a series of format specifiers that indicate how each component of the date is to be formatted. Below is an example (look in the class documentation for a complete list of modifiers):

Code: PHP

```
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("1951-09-13 16:55:11");

// retrieve date as formatted string
echo $d->format("%A, %d %B %Y %T");
?>
```

And here's the output:

```
Thursday, 13 September 1951 16:55:11
```

Converting between time zones

Now that you've got the basics, let's talk about time zones. Once you have a Date() object initialised, converting from one time zone to another is a simple two-step process:

1. Tell the Date class which time zone you're converting from, with the setTZByID() method.
2. Then, tell the Date class which time zone you wish to convert to, with the convertTZByID() method.

Code: PHP

```php
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("1951-09-13 10:36:27");

// set local time zone
$d->setTZByID("GMT");

// convert to foreign time zone
$d->convertTZByID("IST");

// retrieve converted date/time
echo $d->format("%A, %d %B %Y %T");
?>
```

In this case, I'm attempting to convert from Greenwich Mean Time (GMT) to Indian Standard Time (IST). India is about 5.5 hours ahead of Greenwich, which is why the output of the script is:

```
Thursday, 13 September 1951 16:06:27
```

Simple, isn't it? Here's another example, this one demonstrating how the class handles leap years and month end values.

Code: PHP

```php
<?php
// include class
```

```
include ("Date.php");

// initialize object
$d = new Date("2008-03-01 06:36:27");

// set local time zone
$d->setTZByID("GMT");

// print local time
echo "Local time is " . $d->format("%A, %d %B %Y %T") . "\n";

// convert to foreign time zone
$d->convertTZByID("PST");

// retrieve converted date/time
echo "Destination time is " . $d->format("%A, %d %B %Y %T");
?>
```

And the output is:

Code:

```
Local time is Saturday, 01 March 2008 06:36:27
Destination time is Friday, 29 February 2008 22:36:27
```

Note: In case you're wondering where the time zone IDs come from, you can find a complete list within the class documentation.

Calculating GMT offsets

Another piece of information that's sometimes useful when working with time zones is the GMT offset -- that is, the difference between the specified time zone and standard GMT. The PEAR Date class lets you get this information easily, via its getRawOffset() method. Here's an example:

Code: PHP

```php
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("2006-06-21 10:36:27");

// set local time zone
$d->setTZByID("PST");

// get raw offset from GMT, in msec
echo $d->tz->getRawOffset();
?>
```

Here, the getRawOffset() method calculates the time difference between the local time and GMT. Here's the output:

```
-28800000
```

Note that this offset value is expressed in milliseconds, so you will need to divide it by 3600000 (the number of milliseconds in one hour) to calculate the time zone difference in hours.

Tip: You can use the inDaylightTime() method to see if the destination is currently observing daylight savings time. Look in the class documentation for details on this method.

Adding and subtracting timespans

The Date class also lets you perform sophisticated date arithmetic on temporal values, adding or subtracting durations to a date/time value. These durations (or timespans) are expressed as a string containing day, hour, minute and/or second components.

Code: PHP

```php
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("1951-09-13 16:55:11");

// add 01:20 to it
$d->addSpan(new Date_Span("0,1,20,0"));

// retrieve date as formatted string
echo $d->format("%A, %d %B %Y %T");
?>
```

In this case, I've added an hour and twenty minutes to the initial timestamp, by calling the Date class' addSpan() method and supplying it with a Date_Span() object initialised to that duration. The output is fairly easy to guess:

```
Thursday, 13 September 1951 18:15:11
```

Just as you can add timespans, so too can you subtract them. That, in fact, is the purpose of the subtractSpan() method, which is illustrated below.

Code: PHP

```php
<?php
// include class
include ("Date.php");

// initialize object
$d = new Date("1951-09-13 16:55:11");

// add 01:20 to it
$d->addSpan(new Date_Span("0,1,20,0"));

// subtract 00:05 from it
```

```
$d->subtractSpan(new Date_Span("0,0,5,0"));

// retrieve date as formatted string
echo $d->format("%A, %d %B %Y %T");
?>
```

Here, I've first added an hour and twenty minutes, and then subtracted a further five minutes. The net effect is an addition of an hour and fifteen minutes, and the output reflects this:

```
Thursday, 13 September 1951 18:10:11
```

As the examples above illustrate, the PEAR Date class provides methods to easily and efficiently perform fairly complex date math. If you're looking for a stress-free way to convert timestamps between different locations, I'd recommend it to you.

Source: http://www.go4expert.com/articles/using-pear-date-converting-time-zones-t3494/