Advanced Date/Time Handling

Derick Rethans

derick@php.net – @derickr

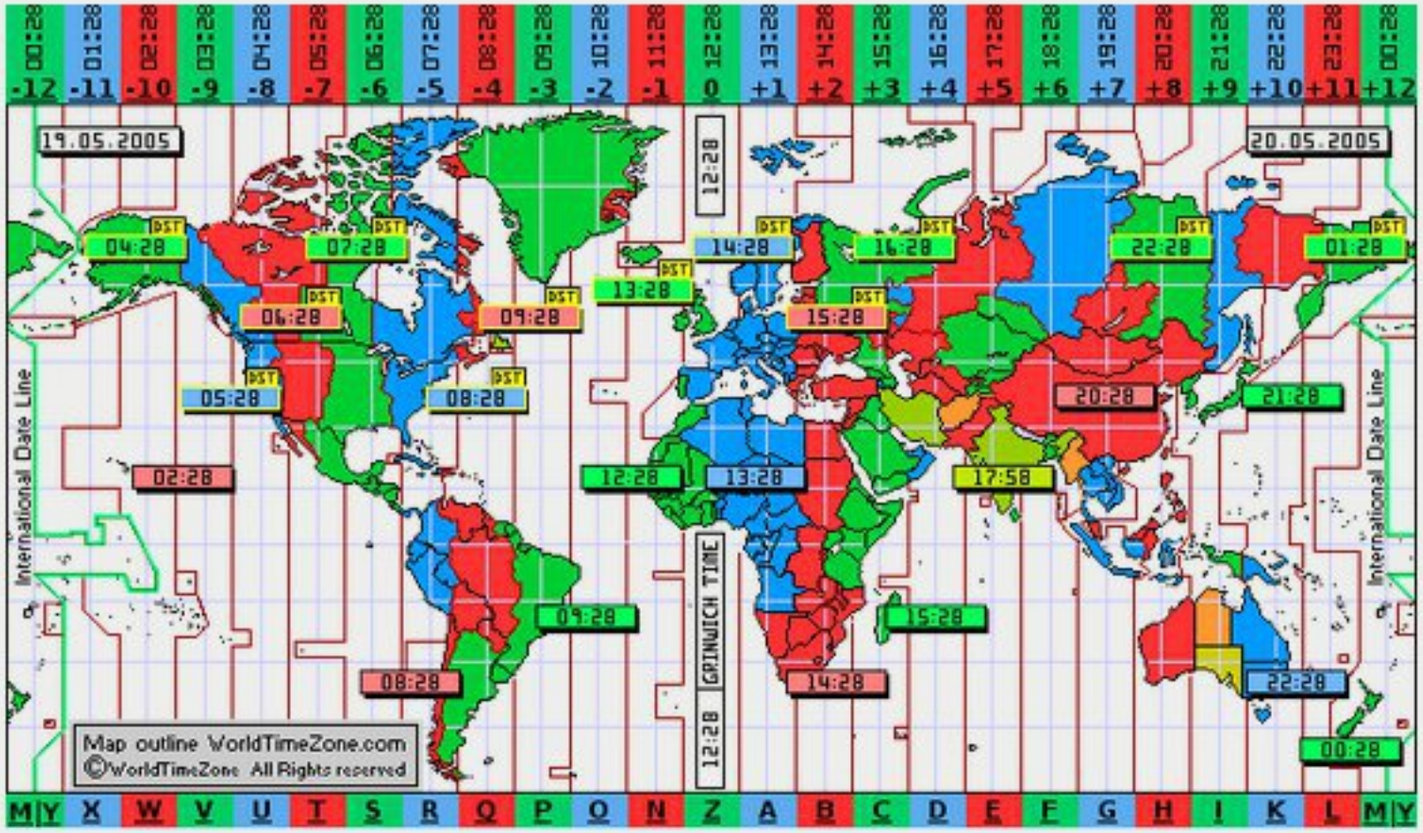
http://joind.in/10617

About Me

Derick Rethans

- Dutchman living in London
- One of the PHP MongoDB driver maintainers
- Author of the PHP debugger tool Xdebug, PHP's Date/Time/Timezone support, and a bunch of other PHP extensions
- I ♥ maps

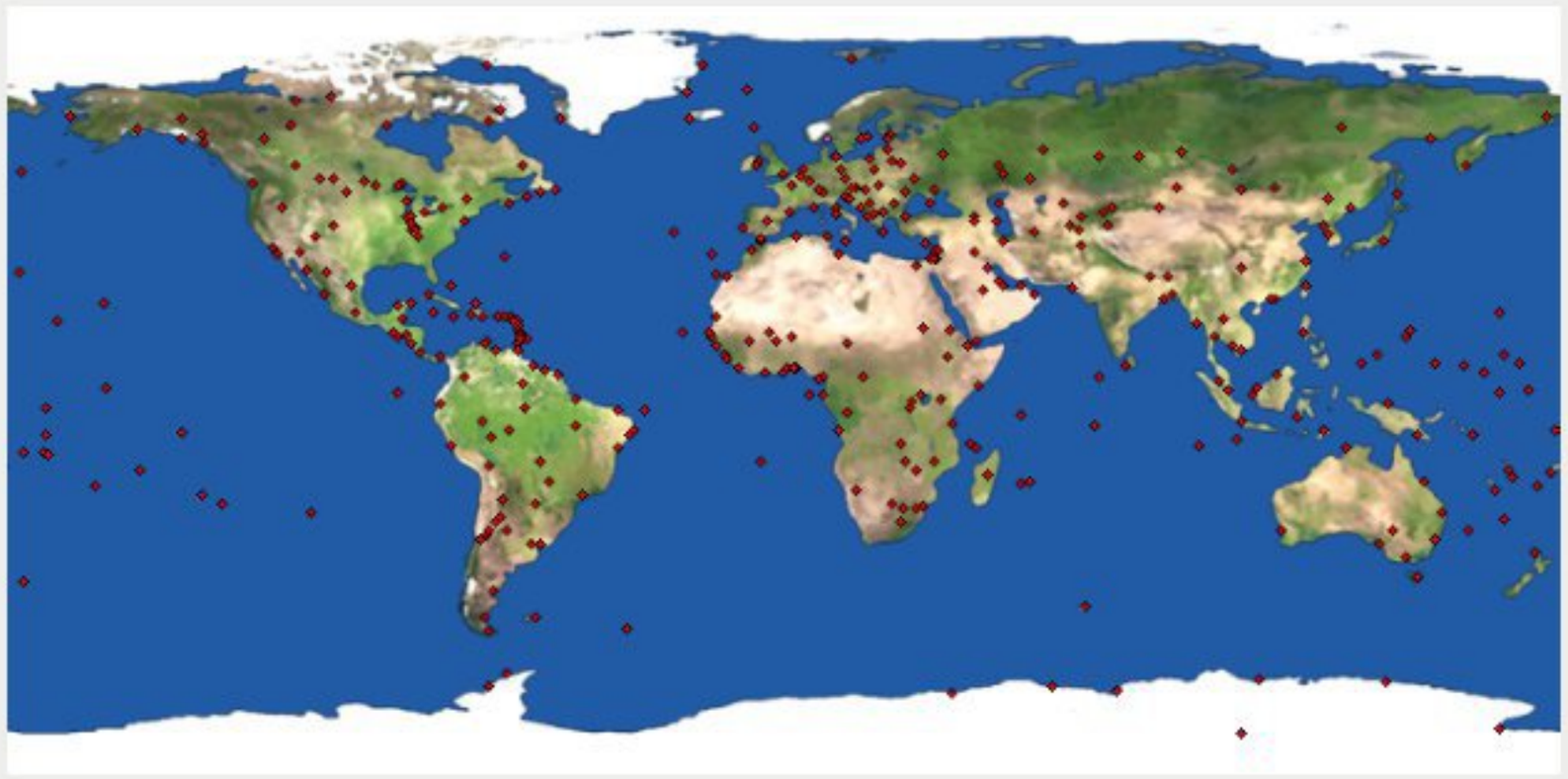
Timezones ... ?



- Most places have whole-hour timezone offsets
- Some places change timezones during the year
- One identifier can mean different zones:
  PST: **P**acific **S**tandard **T**ime, **P**akistan **S**tandard **T**ime
  EST: **E**astern **S**tandard **T**ime (USA), **E**astern **S**tandard **T**ime (Australia) and **E**astern Brazil **S**tandard **T**ime

Timezone Support

- There are more than 500 zones
- Do not dependend on timezone abbreviations
- Use timezone IDs like `Europe/Amsterdam` or `America/Indiana/Knox`



Changing Timezone Definitions

- An updated database is released about 20 times a year.
- Some of the changes are **very** sudden.
- PHP releases will therefore often have an outdated version.
- The PECL extension timezonedb provides a drop in replacement for the timezone database.
- `pecl install timezonedb`

Default Timezones

Setting a default timezone:

```
<?php
        date_default_timezone_set("Europe/Oslo");
        $ts = new DateTime("1978-12-22 09:15");
        echo $ts->format("e");
?>
```

Getting a default timezone:

```
<?php
        $default_identifier = date_default_timezone_get();
        echo $default_identifier;
?>
```

Default timezone is 'guessed' in the following order:

- *date_default_timezone_set()* value
- php.ini's `date.timezone` setting

Unix timestamps and timezones

```
<?php
  $date = strtotime( '2010-03-07 20:48:21 America/Toronto' );
  echo date( 'Y-m-d', $date ), "\n";
?>
```

Result:

```
2010-03-08
```

Explanation:

```
$d = strtotime('2010-03-07 20:48 America/Toronto');

$d = 1268012901; // string is turned into a number

echo date('Y-m-d', 1268012901), "\n";
```

Number is converted to a date/time string using the **default** timezone:

```
"2010-03-08" (01:48:21 Europe/London)
```

Parsing Dates

Parsing strings for date time information:

```
<?php
  $dt = new DateTime("2010-03-08 08:43:57");
?>
```

This function will **not** return the timestamp as an integer, but instead returns a DateTime object, which you can access (as string) through:

```
<?php
  $dt = new DateTime("2010-03-08 08:44:12");
  echo $dt->format( 'U' ), "\n";
?>
```

Result:

```
1268037852
```

The DateTime class is what you can do the really cool things with.

Parsing Dates

"22apr2006 8:36:14 # Europe/Oslo CEST"

Parsing strings with the *date_parse()* function:

```
<?php
$date = "22apr2006 8:36:14.43 # Europe/Oslo CEST";
$t = date_parse( $date );

echo $t['year'], '-', $t['month'], '-', $t['day'], " ";
echo $t['hour'], ':', $t['minute'], ':', $t['second'] + $t['fraction'], " ";
echo $t['tz_id'], "\n";

if ( $t['warning_count'] )
  echo "Warnings:\n";
  foreach( $t['warnings'] as $pos => $message )
    echo "- $message @$pos\n";
if ( $t['error_count'] )
  echo "Errors:\n";
  foreach( $t['errors'] as $pos => $message )
    echo "- $message @$pos\n";
```

Result:

```
2006-4-22 8:36:14.43 Europe/Oslo
Warnings:
- Double timezone specification @35
Errors:
- Unexpected character @21
```

Parsing Dates

08/03/10

Aug 3rd, 2010

(default)

Parsing Dates

Creation with the *date_create_from_format()* function:

```
<?php
$date = "06/08/04";
$obj = DateTime::createFromFormat( '!d/m/y', $date )->format( DateTime::ISO8601 );
?>
```

Parsing with the *date_parse_from_format()* function:

```
<?php
$date = "06/08/04";
print_r( date_parse_from_format( '!y*d*m', $date ) );
?>
```

Example:

Pattern	String	Y	M	D
!y*d*m	06/08/04	2006	04	08

Modifying Dates

```php
<?php
  date_default_timezone_set("Europe/London");
  $ts = new DateTime("2010-03-08 15:53:55");

  $ts->modify("+2 days");

  $ts->modify("Friday +3 weeks");

  $ts->modify("next friday");
?>
```

Result:

```
Mon, 08 Mar 2010 15:53:55 +0000

Wed, 10 Mar 2010 15:53:55 +0000

Fri, 02 Apr 2010 15:53:55 +0100

Fri, 09 Apr 2010 15:53:55 +0100
```

Immutable DateTime

Standard:

```
<?php
  $ts = new DateTime("2010-03-08 15:53:55");
  $new = $ts->modify("+2 days");
  echo $ts->format(DateTime::RFC822), "\n";
  echo $new->format(DateTime::RFC822), "\n";
?>
```

Result:

```
Wed, 10 Mar 10 15:53:55 +0000
Wed, 10 Mar 10 15:53:55 +0000
```

Immutable DateTime

Immutable (new in PHP 5.5):

```
<?php
  $ts = new DateTimeImmutable("2010-03-08 15:53:55");
  $new = $ts->modify("+2 days");
  echo $ts->format(DateTime::RFC822), "\n";
  echo $new->format(DateTime::RFC822), "\n";
?>
```

Result:

```
Mon, 08 Mar 10 15:53:55 +0000
Wed, 10 Mar 10 15:53:55 +0000
```

Using Timezones

Creating a timezone resource:

```
<?php
  $tz = new DateTimeZone("Asia/Singapore");
?>
```

Using the timezone when parsing a string:

```
<?php
  $tz = new DateTimeZone("Pacific/Honolulu");
  $ts = new DateTimeImmutable("1978-12-22 09:15", $tz);
?>
```

A passed timezone does not override a **parsed** timezone:

```
<?php
  $tz = new DateTimeZone("Pacific/Honolulu");
  $ts2 = new DateTimeImmutable("1978-12-22 09:15 Europe/London", $tz);
  echo $ts2->format( DateTime::RFC2822 );
?>
```

Result:

```
Fri, 22 Dec 1978 09:15:00 +0000
```

Updating the Timezone

Using the timezone when parsing a string with a date representation:

```
<?php
  $tz1 = new DateTimeZone("Pacific/Honolulu");
  $tz2 = new DateTimeZone("Australia/Melbourne");

  $ts = new DateTimeImmutable("1978-12-22 09:15", $tz1);
  echo $ts->getTimezone()->getName(), ':    ',
    $ts->format(DateTime::RFC2822), "<br/>";

  $melbourne = $ts->setTimezone($tz2);

  echo $melbourne->getTimezone()->getName(), ': ',
    $melbourne->format(DateTime::RFC2822), "<br/>";
?>
```

Result:

```
Pacific/Honolulu:    Fri, 22 Dec 1978 09:15:00 -1000
Australia/Melbourne: Sat, 23 Dec 1978 06:15:00 +1100
```

Timezones Utilities

Restricting listing by geographical location

Restrict by continent:

```
<?php
  $ids = timezone_identifiers_list( DateTimeZone::EUROPE );
  echo implode( ", ", $ids );
?>
```

Result:

```
Europe/Amsterdam, Europe/Andorra, Europe/Athens, Europe/Belgrade, Europe/Berlin, Europe/Bra
```

Restrict by country:

```
<?php
  $ids = timezone_identifiers_list( DateTimeZone::PER_COUNTRY, 'US' );
  echo implode( ", ", $ids );
?>
```

Result:

```
America/Adak, America/Anchorage, America/Boise, America/Chicago, America/Denver, America/De
```

Intervals

Creating an interval:

```
<?php
// full notation
$i = new DateInterval( 'P0003-01-15T06:12:30' );

// abbreviated notation
$i = new DateInterval( 'P3DT6H' );

// from a difference
$d1 = new DateTimeImmutable();
$d2 = new DateTimeImmutable( "2014-06-20 11:45 Europe/London" );
$i = $d1->diff( $d2 );

// displaying the difference
echo $i->format( '%a days, %h hours, %i minutes' ), "\n";
echo $i->format( '%m months, %d days, %h hours, %i minutes' ), "\n";
?>
```

Result:

```
29 days, 15 hours, 25 minutes
0 months, 29 days, 15 hours, 25 minutes
```

```
<?php
// create from string
$i = DateInterval::createFromDateString( "next weekday" );
?>
```

Intervals

Applying an interval

```
<?php
$d = new DateTimeImmutable( 'Aug 28th, 2009' );
echo $d->format( "l Y-m-d\n" );

$i = DateInterval::createFromDateString( "next weekday" );
echo $d->add( $i )->format( "l Y-m-d\n" );

$i = DateInterval::createFromDateString( "3 months 10 days" );
echo $d->sub( $i )->format( "l Y-m-d\n" );
?>
```

Result:

```
Friday 2009-08-28

Monday 2009-08-31

Thursday 2009-05-18
```

Relative Time

"previous month" / "next month"

```
<?php
  $date = new DateTimeImmutable( '2010-01-31 15:48:21' );
  echo $date->modify( 'next month' )->format( 'Y-m-d' ), "\n";
?>
```

Result:

```
2010-03-03
```

Explanation:

```
2010-03-03 15:48:21

February only has 28 days, so days are overflown
into the next month
```

Relative Time

"first day of next month"

```
<?php
  $date = new DateTimeImmutable( '2010-01-31 15:48:21' );
  echo $date->modify( 'first day of next month' )->format( 'Y-m-d' ), "\n";
?>
```

Result:

```
2010-02-01
```

Explanation:

```
2010-02-01 15:48:21

"first day of" resets the day number to 1
```

Periods

Iterating over a period

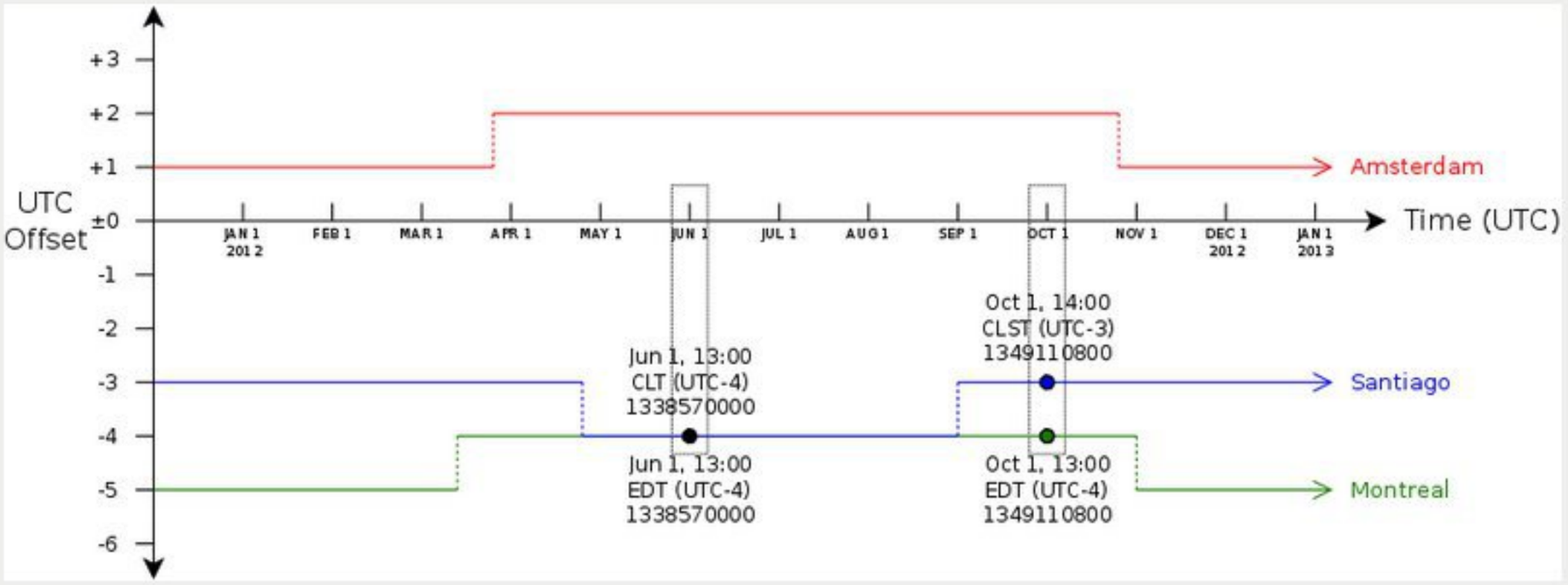
```
<?php
$db = new DateTimeImmutable( '2008-12-31' );
$de = new DateTimeImmutable( '2009-12-31' );
$di = DateInterval::createFromDateString(
        'third tuesday of next month'
);
$dp = new DatePeriod(
        $db, $di, $de, DatePeriod::EXCLUDE_START_DATE
);
foreach ( $dp as $dt )
{
   echo $dt->format( "F jS\n" );
}
?>
```

Result:

```
January 20th
February 17th
March 17th
April 21st
May 19th
June 16th
July 21st
August 18th
September 15th
October 20th
November 17th
December 15th
```

Storing date/time in a database

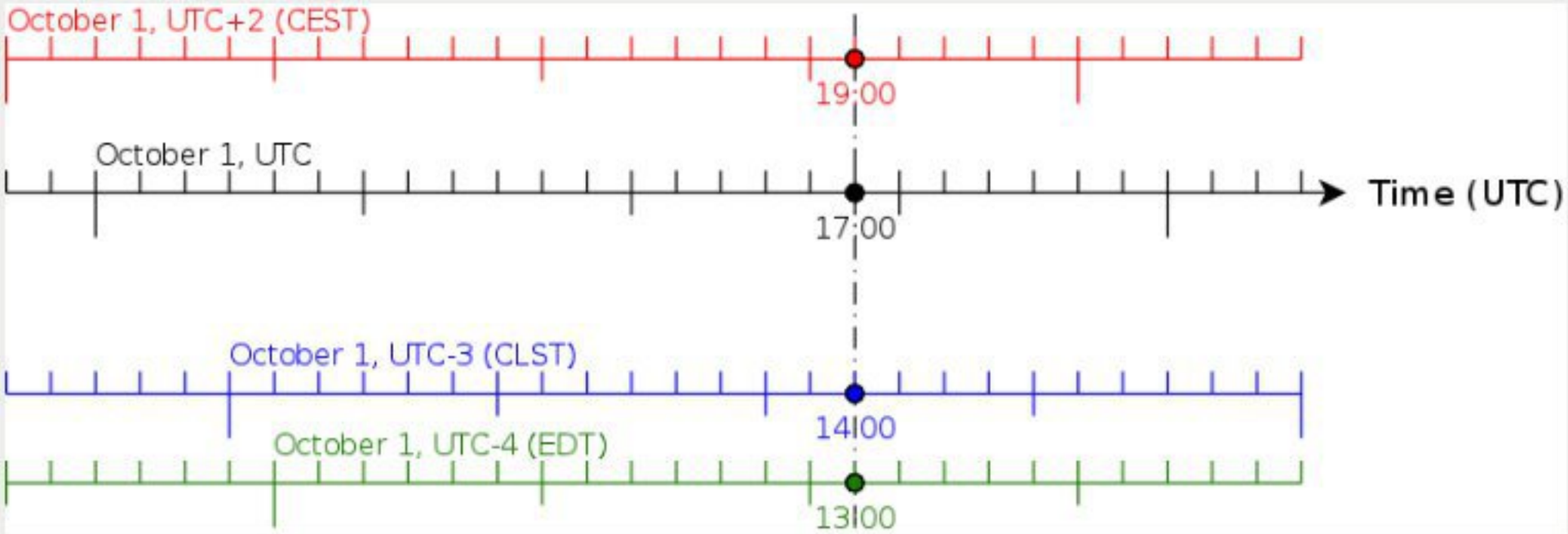
Timestamp and UTC offset for in the future



- **Goal**: Store date **exactly** 4 months in the future
- Store: *timestamp* and *UTC offset* (`UTC-4`)
- Current date/time: Jun 1st, 13:00 UTC-4 (`1338570000`)
- Future date/time: Oct 1st, 13:00 UTC-4 (`1349110800`)
- Correct for Montreal, where DST is still in effect.

Storing date/time in a database

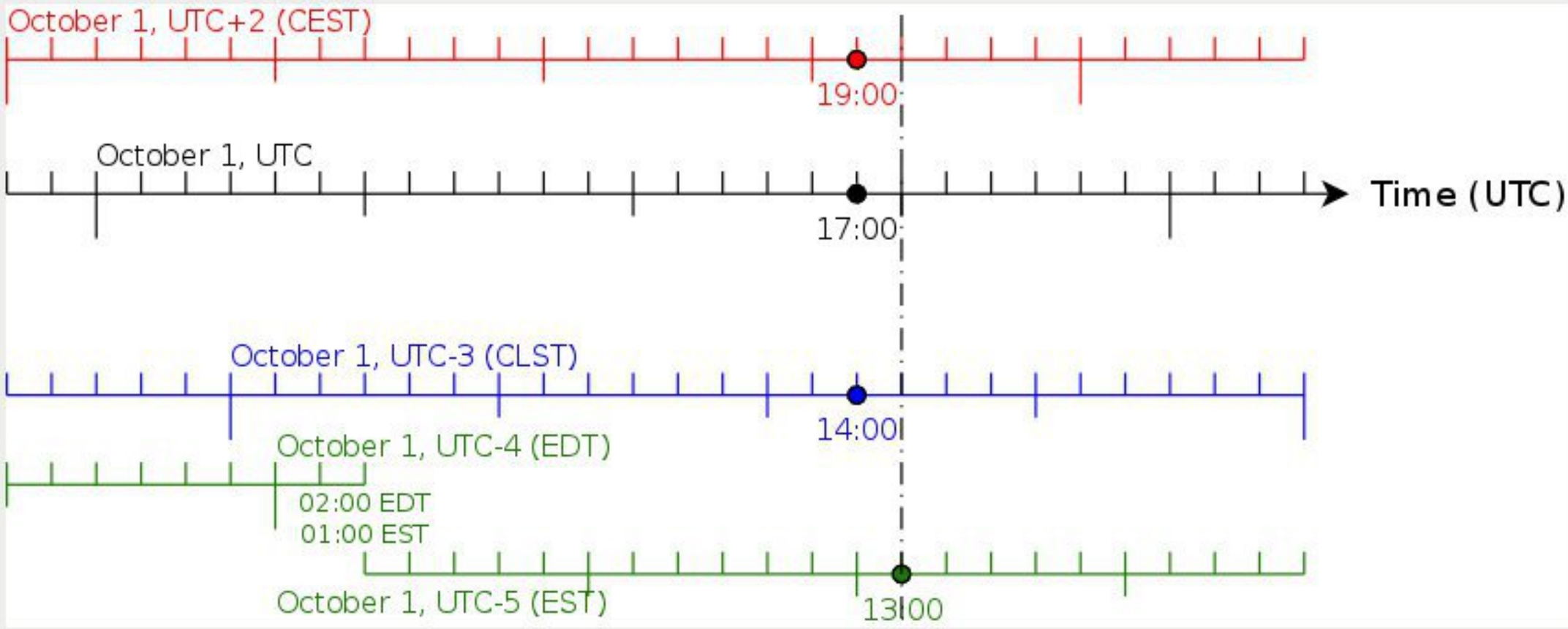
Timestamp and timezone identifier



- **Goal**: Store date **exactly** 4 months in the future
- Store: *timestamp* and *timezone identifier* (`America/Montreal`)
- Current date/time: Jun 1st, 13:00 EDT (`1338570000`)
- Future date/time: Oct 1st, 13:00 EDT (`1349110800`)
- Local time for Santiago can also be calculated with

Storing date/time in a database

Timestamp and timezone identifier?



You can store it as:

- `America/Montreal, 2012 10 1 13 0 0` → Oct 1st, 13:00 EDT
- That however, changes the original point in time…

Storing date/time in a database

Conclusion

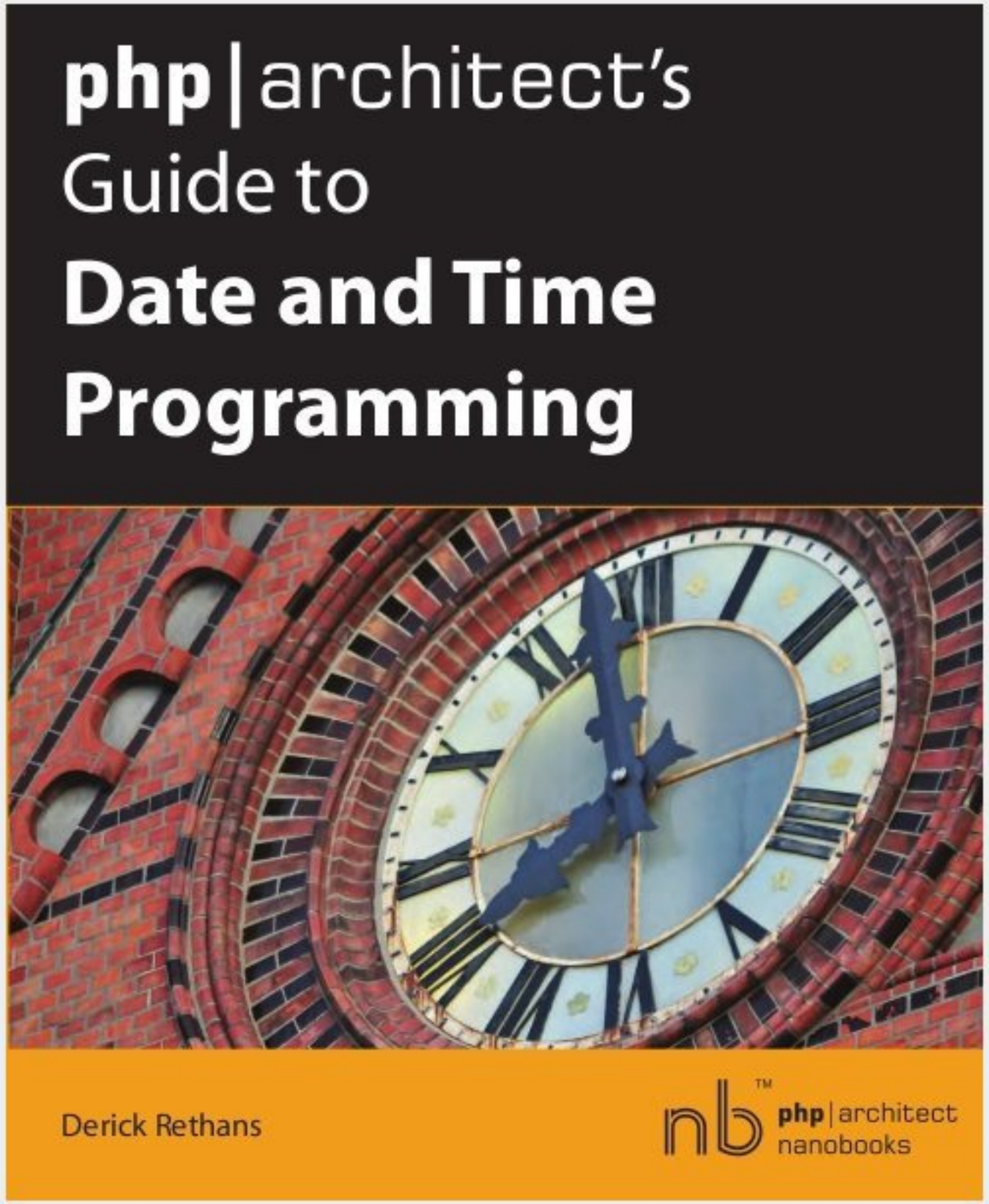
If you want to future proof a specific date and time (Oct 1st, 13:00)-ie, storing a time in **local time**:

```
{
        'title' : 'Giving a talk: Advanced Date/Time handling with PHP',
        'time' : {
                'tz': 'America/Montreal',
                'dt': '2012-10-01 13:00',
        }
}
```

If you want to store a time for an appointment for a group of people all in different timezones-ie, store a **specific point in time**:

```
{
        'title' : 'Meeting about getting rid of Daylight Saving Time',
        'time' : {
                'tz': 'America/Montreal',
                'ts': 1349110800,
        },
        'attendees' : [ 'Arthur David Olson', 'Paul Eggert' ],
}
```

php|architect's guide to Date/Time handling



http://joind.in/10617

derick@php.net — @derickr