Dates

The Date class

To help working with dates the framework provides the HubzeroUtilityDate class. Since that can be a bit much to type every time when instantiating a new instance, a global Date facade can be used instead. To get a Date object that represents the current date and time do the following:

```
$now = Date::getRoot();
```

The first thing to note is that we do NOT use the =& assignment operator. The static getRoot() method does not return references to globally accessible instances of Date. This means each time Date is used it is retrieving a new object.

It is also possible to specify the date and time we want the Date object to represent. A likely source for this would be a DATETIME field extracted from the database.

```
$created = Date::of($row->created);
```

Since HubzeroUtilityDate extends PHP's DateTime object, the method used to parse date and time values is relatively robust. Formats other than the MySQL DATETIME representation of YYYY-MM-DD HH:MM:SS can be used. The table below describes the acceptable formats.

Format	Example	Notes
Timestamp	1254497100	Seconds since the Unix Epoch
RFC 2822	Fri, 2 Oct 2009 15:25:00 +0000	Name of day and UTC offset is optional. Date does not support all of the obsolete RFC 822 time zone identifiers. Date support numeric time zone identifiers, UT, GMT, and military time zones.
RFC 3339	2009-10-02 T15:25:00+00:00	RFC 3339 time zone offset can be expressed numerically or as the time zone alpha identifier Z (Zulu, UTC+0). RFC 3339 *is also known as* ISO 8601.
US English date format	2 October 2009	For more information about US English date formats refer to

Format	Example	Notes
		http://php.net/strtotime.

In the table above both the RFC 2822 and RFC 3339 examples include a UTC offset in the value. In the examples the offset is 0. Date always internally represents the date and time in the UTC+0 time zone. Had the offsets in the examples been non zero values, and had we used these to create new Date objects, we would have found that the date and time within the Date objects would have been adjusted to represent a timezone of UTC+0.

Outputting Dates

The Date class includes five handy methods for retrieving formatted date and time strings. The most versatile of these methods is format(). This method allows for explicitly defining the format in which the date and time are to be described. The format can be defined in the same way as when using the PHP function strftime().

```
$string = $myDate->format('%Y-%m-%d');
```

The remaining four methods to retrieve formatted date and time strings are used to extract specific representations of the date and time. These representations are RFC 2822 (successor to RFC 822), ISO 8601 (also known as RFC 3339), Unix timestamp, and SQL (determined by the specific database connector used).

```
// D, d M Y H:i:s
// Tuesday, 06 October 2009 12:54:37+0000
$rfc2822 = $myDate->toRFC822();
```

```
// Y-m-dTH:i:s
// 2009-10-06T12:54:37Z
$iso8601 = $myDate->toISO8601();
```

```
// Unix timestamp
// 1254833677
$unix = $myDate->toUnix();
```

```
// The format is determined by the databse being used. The following e
xample is for MySQL.
// Y-m-d H:i:s
// 2009-10-06 12:54:37
$mysql = $myDate->toSql();
```

Outputting dates in different time zones

As mentioned above, Date internally stores dates and times in the UTC time zone. In conjunction with that it is good practice to store dates and times in the database in the UTC time zone. For end users however, this is not necessarily easy to read. To aid with this, Date can output dates and times in different time zones.

In addition to the date and time that a Date object represents, a Date object can also record a time zone in which to output formatted dates. This value can be set and retrieved with the setTimezone and getTimezone methods, respectively.

The timezone being discussed in this section is separate from the timezone specified when *creating* a new Date object.