Toodledo Repeat Parser

This document describes a library to be built for managing repeating tasks.

Toodledo has the capability to have tasks that repeat on a schedule. We support a variety of ways for people to specify how a task repeats (http://www.toodledo.com/info/help.php?sel=2). Currently, these values are stored as a string eg "every 2 months". We would like to improve this to provide for more flexibility in the future by standardizing on the iCalendar RRULE spec (http://tools.ietf.org/html/rfc2445).

Reference: https://lists.oasis-open.org/archives/obix-xml/200708/msg00001.html

Conversion to RRULE

The first thing that we will need is a way to convert all the old string values into RRULEs.

```
function convertToRRule($text,$fromcomp) {}
```

This function would take a string and convert it into the compatible RRULE. Examples

1) "Every 2 months" => "MONTHLY;INTERVAL=2"
2) "Weekday" => "WEEKLY;BYDAY=MO,TU,WE,TH,FR"

Some of this work has already been done and will be provided to you, but you will need to verify that it works for all cases and write tests. There are three special cases that we will need to handle that are not supported by RRULEs.

1) Subtasks can repeat based on their parent's repeat value and if this is set then the old string will be "With Parent". There is not a comparable RRULE for this, so we will need to translate this to a custom RRULE of "PARENT".
2) Tasks can be set to repeat from their due-date or their completion date. This is currently stored in a separate Boolean flag but we will want to combine it into the RRULE with a custom tag. If the task is set to repeat from the due-date, then don't add anything, this is the default value. If the task is set to repeat from the completion date, append ";FROMCOMP" to the RRULE.
3) We would like to support the ability for tasks to get fast forwarded to a future date if the rescheduled date would still be in the past. This functionality doesn't exist yet so you won't need to build this into the

conversion function, but we should support appending ";FASTFORWARD" to an RRULE.

Getting Next Occurrence

Once we have converted everything to an RRULE, we will need another function that takes a date and an RRULE and calculates the next occurrence.

```
function getNextDates($start,$due,$comp,$rrule) {
	…
	return array($newstart,$newdue,$rrule);
}
```

This function will take three dates as parameters, which will be a unix time stamps. It will also take the rule. You will need to calculate and return the new dates and new rule if necessary. This should work for any valid RRULE. There may be 3rd party libraries that can help (https://github.com/tplaner/When), so please leverage that work if it exists and works and has a compatible license.

A few features to keep in mind
1) It should be possible to have an end date ("UNTIL") or max occurrences ("COUNT") after which the task will no longer repeat
2) It should be possible to repeat the task on the last day of each month
3) It should work correctly for leap years in Feb.
4) It should have the ability to do Mother's day (second sunday of May)

Custom Rules and Considerations
1) For tasks that have an RRULE of "PARENT", the calling function will detect this and pass the parent's RRULE into the getNextDates() function, so you don't need any special logic for this case.
2) If the RRULE contains "FROMCOMP" then use $comp to calculate the next duedate instead of $due.
3) If the RRULE contains "FASTFORWARD" then apply the repeat rule as many times as necessary to get a future due-date.
4) The $start date should always be moved forward by the same amount of time as $due maintaining the same interval between $start and $due.
5) If $start is 0 then keep it at 0
6) If $due is 0 then keep it at 0 and reschedule $start instead.
7) If $rrule is empty, unparsable or does not have a next occurrence then return (-1,-1,'').
8) If the task should not repeat after this occurrence because the max occurrences or end date has been reached, return ($newstart,$newdue,'').

9) If $rrule contains “COUNT” then decrement the number in the count rule and return the new $rrule.

Testing

You should create unit tests to easily test your code. Tests should be written in PHPUnit (https://github.com/sebastianbergmann/phpunit/)

Project Requirements

The project will use Github to manage the code repository.

This project should be written in pure PHP. If you use any 3rd party libraries, please make sure that the license is compatible with our usage.

You will need to keep track of your hours worked on the project. Please round up or down to the nearest 15 minutes.