Zend_Measure Proposal - Thomas Weidner

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Measure Component Proposal

Proposed Component Name	Zend_Measure
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Measure
Proposers	Thomas Weidner Gavin (Zend-liaison)
Revision	4.0 - 25 Jan 2007: API rework, simpler useage 3.1 - 23 July 2006: Corrections based on actual work 3.0 - 18 July 2006: Class redesign based on actual work and comments from Zend 2.0 - 26 June 2006: Complete redesign of class skeleton, added new functions 1.1 - 20 June 2006: Changed class to be static 1.0 - 19 June 2006: Initial release extracted from Zend_Locale Proposal V1.0. (wiki revision: 29)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_Measure is a locale aware Class which can convert measurement units.
Hint: The name Zend_Measure *was taken because* Zend_Convert *implements to convert between different file-types (Text->Excel, Windows->Unix,...) and not measurement units.*

2. References

- Conversion tables
- Conversion Sites

3. Component Requirements, Constraints, and Acceptance Criteria

- Lightweight and fast implementation

- Simple use for ZF-user
- Handling of different input formats

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Locale_Data

5. Theory of Operation

Zend_Measure can convert measurement units between different locales. The Conversion can be done for

- Length
- Weight
- Temperature
- Area
- Power
- Energy
- and much more...

6. Milestones / Tasks

zone: Missing {zone-data:milestones}

7. Class Index

- Zend_Measure_Exception
- Zend_Measure

8. Use Cases

Get output of Kelvin in actual locale

```
$unit = new Zend_Measure_Temperature('23,643°');
    echo $unit;
    // returns 23,643°K when en_EN is standard locale
```

Get formatted output in different locale

```
$unit = new Zend_Measure_Temperature('23,643', Zend_Measure_Temperature::FAHRENHEIT, 'de_DE');
    echo $unit->toString();
    // returns 23,643°F as de_DE is set (',' instead of '.')
```

Different Input Types which would be recognized

```
$input = new Zend_Measure_Length('0',Zend_Measure_Length::STANDARD);                 // = 0 meter
    $input = new Zend_Measure_Length('123456',Zend_Measure_Length::KILOMETER);           // =
123456000 meter
    $input = new Zend_Measure_Length('123456','KILOMETER');                              // =
123456000 meter
    $input = new Zend_Measure_Length('123,456',Zend_Measure_Length::STANDARD);           // = 123456
meter
    $input = new Zend_Measure_Length('123,456.7890',Zend_Measure_Length::STANDARD);      // =
123456.7890 meter
    $input = new Zend_Measure_Temperature('the earth is 14,350.5 kilometer and it has
24°',Zend_Measure_Temperature::STANDARD);
    // this would return 14350.5 celsius, as only the first number will be parsed
    $input = new Zend_Measure_Length('Die Sonne hat 14.350,5 km breite',
Zend_Measure_Length::KILOMETER,'de_DE');
    // value would be 14350500 meter as in german the group seperation is '.' and not ',' as in
english
    $input = new Zend_Measure_Length('14.350,6',Zend_Measure_Length::KILOMETER);
    // value would be 14350 meter when standard locale is en
    // but it would be 14350600 meter when the standard locale is de
```

Convert Fahrenheit to Celsius

```
$unit = new Zend_Measure_Temperature('86,7°F',Zend_Measure_Temperature::FAHRENHEIT); // measurement
unit value is 86.7
    echo $unit->convert(Zend_Measure_Temperature::CELSIUS);
     // output 23,7°C
```

Convert Kilo to Tonne with other locale

```
$unit = new Zend_Measure_Weight('25742.54 kg',Zend_Measure_Weight::KILOGRAM);
    echo $unit->convert(Zend_Measure_Weight::TONNE,'de_DE');
    // outputs 25,74T
```

Maths with different units

```
$unit = new Zend_Measure_Weight('23.5kg', Zend_Measure_Weight::KILOGRAM);
    $second = new Zend_Measure_Weight('10pd', Zend_Measure_Weight::POUND);
    echo $unit->add($second);
    // outputs 28.03kg
    echo $unit->sub($second);
    // outputs 23.5kg
    echo $unit->compare($second);
    // outputs 18.97kg
```

9. Class Skeletons

```


class Zend_Measure_Exception extends Zend_Exception {}

class Zend_Measure_Length extends Zend_Measure_Abstract
{
    const STANDARD = 'meter';
```

```
    const METER = 'meter';
    const KILOMETER = 'kilometer';

    private static $_UNITS = array (
        'meter' => array(1,'m'),
        'kilometer' => array(0.001,'km')
    );

    /**
     * Zend_Measure_Length provides an locale aware class for
     * conversion and formatting of length values
     *
     * Zend_Measure $input can be a locale based input string
     * or a value. $locale can be used to define that the
     * input is made in a different language than the actual one.
     *
     * @param mixed $value   - string, integer, real or float
     * @param type  $type    - OPTIONAL a Zend_Measure_Length Type
     * @param locale $locale - OPTIONAL a Zend_Locale Type
     * @throws Zend_Unit_Exception
     */
    public function __construct($value, $type, $locale) {}

    /**
     * Compares two units
     */
    public function equals( Object $length ) {}

    /**
     * Extracts the value from the input string
     *
     * @param mixed $value   - string, integer, real or float
     * @param type  $type    - OPTIONAL a Zend_Measure_Length Type
     * @param locale $locale - OPTIONAL a Zend_Locale Type
     * @throws Zend_Measure_Exception
     */
    public function setValue($value, $type, $locale) {}

    /**
     * Returns the actual value
     *
     * @return mixed (integer/float)
     */
    public function getValue() {}

    /**
     * Sets a new type for the unit
     *
     * @param type $type
     * @throws Zend_Measure_Exception
     */
    public function setType($type) {}
    /**
     * Returns the actual type
     *
     * @return string
     */
    public function getType() {}

    /**
     * Returns the locale formatted unit string (f.e. 1,234.567)
     *
     * @param $type   - OPTIONAL other type for output
     * @param $locale - OPTIONAL other locale for output (f.e. output in german locale 1.234,567)
     * @param $format - OPTIONAL other format for output (f.e. for rounding with precicion of 2
1,234,57)
     * @return string
     */
    public function toString() {}
    public function __toString() {}
```

```
    /**
     * Converts the standard unit to the expected unit
     * Alias for toString($type, OPTIONAL, OPTIONAL)
     *
     * @param $type
     * @param $locale - OPTIONAL locale for output
     * @param $format - OPTIONAL format for output
     * @return string
     */
    public function convert($type, $locale, $format) {}

    /**
     * Returns a plain value
     *
     * @param $type - OPTIONAL other type
     * @return mixed (integer or float)
     */
    public function toValue($type)

    /**
     * Adds a unit to an other
     *
     * @param $unit mixed - unit object
     * @return unit - return the new unit
     */
    public function add($unit) {}

    /**
     * Substracts a unit from an other
     *
     * @param $unit mixed - unit object
     * @return unit - return the new unit
     */
    public function sub($unit) {}

    /**
     * Compares two units, returning the difference
     *
     * @param $unit mixed - unit object
     * @return unit - return the new unit
     */
    public function compare($unit) {}
}


class Zend_Measure_Weight      extends Zend_Measure_Abstract {}
class Zend_Measure_Volume      extends Zend_Measure_Abstract {}
class Zend_Measure_Area        extends Zend_Measure_Abstract {}
class Zend_Measure_Speed       extends Zend_Measure_Abstract {}
class Zend_Measure_Temperature extends Zend_Measure_Abstract {}
class Zend_Measure_Time        extends Zend_Measure_Abstract {}
class Zend_Measure_Pressure    extends Zend_Measure_Abstract {}
class Zend_Measure_Energy      extends Zend_Measure_Abstract {}
class Zend_Measure_Power       extends Zend_Measure_Abstract {}
class Zend_Measure_Circular    extends Zend_Measure_Abstract {}
// and other types

/**
 * Defines an Abstract Class for all Types
 */
abstract class Zend_Measure_Abstract
{
    private $_value;    // internal standard measurement value

    private $_type;     // value type of input

    /**
     * Returns the standard value
     */
```

```
    protected function getValue() {}

    /**
     * Sets a new value
     */
    protected function setValue($value) {};

    /**
     * Returns the original type
     */
    protected function getType() {};

    /**
     * Sets the original type
     */
    protected function setType($type) {};

    /**
     * Serializes the object
     */
    public function serialize() {}

    /**
     * Unserialize the object
     */
    public function unserialize($serialized) {}

    /**
     * Throws an Exception, faster than implementing Zend_Measure_Exception every time
```

```
      */
     public function throwException($message) {}
 }
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>