

WikiPlot Documentation

Contents

WikiPlot Userguide	1
WikiPlot Syntax Reference	3
Package WikiPlot Procedural Elements	8
cache.class.php	8
CleanupCache.php	9
WikiPlot.php	10
Function RenderWikiPlot	10
Function wfWikiPlotExtension	10
Function WikiPlotDeserializeBoolean	10
Function WikiPlotDeserializeColor	10
Function WikiPlotDeserializeInteger	11
Function WikiPlotDeserializeMixed	11
Function WikiPlotDeserializeString	12
WikiPlotSettings.php	14
Define WikiPlotCacheAge	14
Define WikiPlotCachePath	14
Define WikiPlotCacheURL	14
Define WikiPlotMaxUnusedAge	15
xml.class.php	16
Package WikiPlot Classes	17
Class Cache	17
Method CachePath	17
Method CleanupCache	18
Method CleanupMaxAge	18
Method CleanupUnused	18
Method FileExist	18
Method FileURL	19
Class XMLParser	19
Var \$Attributes	20
Var \$Input	20
Var \$Parser	21
Var \$Separator	21
Var \$Tag	22
Var \$TagData	22
Var \$Tags	22
Constructor XMLParser	23
Method CloseTag	23
Method CreateInputArray	24
Method CreateTagArray	24
Method ExplodeInputData	24
Method GetCharData	25
Method OpenTag	25

Method Parse	26
evalmath.class.php	27
graph.plot.class.php	28
plot.class.php	29
test.php	30
Class EvalMath	30
Var \$f	30
Var \$fb	31
Var \$last_error	31
Var \$suppress_errors	31
Var \$v	31
Var \$vb	31
Constructor EvalMath	31
Method e	31
Method evaluate	31
Method funcs	31
Method nfx	31
Method pfx	32
Method trigger	32
Method vars	32
Class EvalMathStack	32
Var \$count	32
Var \$stack	32
Method last	32
Method pop	32
Method push	32
Class Graph	33
Var \$Color	33
Var \$EnableLabel	33
Var \$Exp	34
Var \$Label	34
Var \$LabelFont	34
Method GetHashCode	35
Class Plot	35
Var \$BackgroundColor	36
Var \$Caption	36
Var \$CaptionFont	36
Var \$EnableAxis	37
Var \$EnableGrid	37
Var \$Graphs	37
Var \$GridColor	38
Var \$GridFont	38
Var \$Height	38
Var \$MaxX	39
Var \$MaxY	39
Var \$MinX	40
Var \$MinY	40
Var \$Width	41
Var \$XGridSpace	41

Var \$YGridSpace	42
Method DisplayPlot	42
Method DrawAxis	42
Method DrawCaption	43
Method DrawGrid	43
Method DrawPlots	44
Method DrawXGrid	44
Method DrawYGrid	45
Method GeneratePlot	46
Method GetCoordinatX	46
Method GetCoordinatY	47
Method GetHash	47
Method GetImageX	48
Method GetImageY	48
Method GetXGridSpace	49
Method GetYGridSpace	49
Method SaveAs	49
Method ShortNumber	50
Appendices	51
Appendix A - Class Trees	52
WikiPlot	52
Appendix C - Source Code	53
Package WikiPlot	54
source code: cache.class.php	55
source code: CleanupCache.php	58
source code: WikiPlot.php	59
source code: WikiPlotSettings.php	63
source code: xml.class.php	64
source code: evalmath.class.php	69
source code: graph.plot.class.php	75
source code: plot.class.php	77
source code: test.php	88

WikiPlot Userguide

Introduction to WikiPlot

Authors: Jonas F. Jensen.

WikiPlot@Jopsen.dk

WikiPlot Userguide

This is a simple guide to learn and become familiar with WikiPlot syntax, if the WikiPlot extension is installed on the wiki you wish to modify, you can add simply just type the WikiPlot syntax in your file. This chapter will not document all the features of WikiPlot but just the basics.

First things first, let's start with a short introduction to our terminology. We have one plot and one or more graphs (Note the words I just used). Where plot defines the coordinate space, width, height and axes of the final image. A plot contains one or more graphs, which is expressed with a mathematical expression (for instance: x^2+4x+5). If you think this is weird, hold on there, I will clarify in just a moment.

Now let's get dirty, following code will generate an image with 1 graph, from the expression $y(x)=x+4$.

```
<wikiplot>
  <graph>x+4</graph>
</wikiplot>
```

Okay that is possibly the shortest we can make it. I wouldn't be surprised if you would consider that a little too basic. So just to match some basic functionality, we are going to add another graph with the expression $y(x)=3*x-3$, and some optional parameters to modify the output image.

```
<wikiplot height="200" width="200" caption="Simple plot" xspan="-100;100" yspan="-100;100">
  <graph label="Graph 1." color="255,0,0">x+4</graph>
  <graph label="Graph 2.">3*x-3</graph>
</wikiplot>
```

Now this is different. This will result in an image with width and height of 200 pixels. It will have a caption saying **Simple plot**. The image will be a clip of a coordinate space, where minimum X will be -100 and maximum X will be 100, same goes for Y. The image contains 2 labels in the corner, one saying **Graph 1** another saying **Graph 2**, one of them will have the color `rgb(255,0,0)` which is red. Apart from that there will also be 2 graphs.

To simplify the example have been split and explained here:

```
<ul mark = "bullet">
```

- **height**

Height of the output image in pixels.

- **width**

Width of the output image in pixels.

- **caption**

Caption on the output image.

- **xspan and yspan**

Values representing minimum x and maximum x, in coordinate space. If you set `xspan="-50;75"` the lowest x values on your image will be -50 and the highest will be 75. This does not have anything with width to do, and is in no way related to pixels! This feature enables you to zoom in and out on the coordinate space, independent of image size. `xspan` and `yspan` are completely similar except for the fact that they change the y or x coordinate space respectively.

- **$x+4$ and $3*x-3$**

These are mathematical expressions defining the 2 graphs on the image.

- **label**

Labels that are placed in the corner of the image, displayed in the same color as the graphs they represent.

- **color**

Color of the graph, in an RGB (Red,Green,Blue) representation.

If you do not understand this, please feel free to contact us, or post your question at <http://groups.google.com/group/wikiplot> and we will hurry to help. We are well aware that our terminology is very bad, and that some of our syntax might confuse users, so please help us improve.

WikiPlot Syntax Reference

Complete WikiPlot syntax reference

Authors: Jonas F. Jensen.

WikiPlot@Jopsen.dk

WikiPlot Syntax Reference

This is a complete syntax reference for WikiPlot, if you are not familiar with xml or the most common WikiPlot syntax, it is recommended that you read the WikiPlot UserGuide first. Below you will find a documentation of the parameters and content for the wikiplot and graph tag, respectively. At the end of the article you will find a complete example of a plot with use of all parameters.

wikiplot parameters and content

The wikiplot tag contains one or more graph tags, the graph tags defines the different mathematical expressions to be plotted. The wikiplot tag defines the image/environment/coordinate-system these mathematical expressions are to be plotted upon. The wikiplot tag takes following parameters:

<ul mark = "bullet">

- **caption**

Defines the caption of the plot, is shown in the top centered on the final image. Leave empty or do not define this parameter, you do not want any caption on your image.

- **captionfont**

An integer representing font type of the caption, fonts 1-5 are builtin and represents different font sizes 1 being smallest and 5 biggest, defaults to 5.

- **height**

An integer, defining the height of the final image in pixels.

- **width**

An integer, defining the width of the final image in pixels.

- **xspan**

Two semicolon separated integers, defining the span of the x-axis. If xspan="-5;10" the minimum value on the x-axis will be -5 and the maximum value on the x-axis will be 10. This parameter is very important, because it

defines coordinate space to be viewed.

- **yspan**

Two semicolon separated integers, defining the span of the y-axis. If `yspan="-5;10"` the minimum value on the y-axis will be -5 and the maximum value on the y-axis will be 10. This parameter is very important, because it defines coordinate space to be viewed.

- **axis**

Enable or disable axis, whether or not to show axis $x=0$ and $y=0$. Defaults to true, valid values are: "true" or "false".

- **grid**

Enable or disable grid, whether or not to show grid, that makes it easier to read the plot. Defaults to true, valid values are: "true" or "false".

- **gridspace**

Two semicolon separated integers, defining the space between the lines of the grid. If this is not defined, WikiPlot will calculate some appropriate values, but these might not always look good. If `gridspace="10;20"` the distance between the gridlines on the x-axis will be 10 and the distance between the gridlines on the y-axis will be 20.

- **gridfont**

An integer representing font type of the labels at the grid, fonts 1-5 are builtin and represent different font sizes 1 being smallest and 5 biggest, defaults to 1.

- **gridcolor**

Three semicolon separated integers, defining the color of the gridlines, defaults to gray. This `gridcolor="240,240,240"` is an RGB (Red,Green,Blue) representation of a variant of the color gray.

graph parameters and content

The graph tags represent different mathematical expressions, that are to be plotted onto the coordinate-system defined by the surrounding/parent `wikiplot` tag. The graph tag contains the mathematical expression, it is representing. This mathematical expression may contain the variable x , and following mathematical functions:

- `sin()`
- `sinh()`
- `arcsin()`
- `asin()`
- `arcsinh()`
- `asinh()`
- `cos()`
- `cosh()`
- `arccos()`
- `acos()`

- arccosh()
- acosh()
- tan()
- tanh()
- arctan()
- atan()
- arctanh()
- atanh()
- sqrt()
- abs()
- ln()
- log()

Apart from these mathematical functions you may also use following constants:

- e
- pi

And last but not least, you may also use following mathematical operators:

- +
- -
- *
- /
- ^

If you have any questions regarding these mathematical expressions feel free to contact us, or take a look at the source found in `evalmath.class.php`. We haven't documented this class because we have not written it. The graph tag also takes certain parameters that allow you to affect the way it is represented on the plot. The graph tag take following parameters:

`<ul mark = "bullet">`

label

A label shown in the top left corner to identify the graph, this label will be printed in same color as the mathematical expression will be plotted. Leave empty or do not define this parameter, you do not want any label for your mathematical expression.

color

Three semicolon separated integers, defining the color of the label and plotted mathematical expression, defaults to black. This `color="0,0,0"` is an RGB (Red,Green,Blue) representation of the color black.

Complete Example

Following is an advanced example of how WikiPlot could be used. Normally you don't need to used all parameters, most basic es are covered in depth in the WikiPlot Userguide. This is a pretty extrem example of how to use all parameters:

```
ikiplot height="400" width="800" caption="Complete Example"
xspan="-100;100" yspan="-200;200" gridspace="10;20"
captionfont="4" gridfont="2" axis="true" grid="true">
```

```
<graph label="A red graph" color="255,0,0">x^2+4</graph>  
<graph label="A blue graph" color="0,0,255">3*x-3</graph>  
wikiplot>
```


Package WikiPlot Procedural Elements

cache.class.php

File used to control cache

This file provides functions to control the content of the cache. This file is made to make the software more maintain able, and as an interface to the cache for third party developers.

- **Package** WikiPlot
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

quire_once ["WikiPlotSettings.php"](#) *line 29*

Require local settings

This file is needed to control the cache correctly.

CleanupCache.php

File used to clear the cache

This file is supposed to be called as a cron script, to clear the cache on a regular basis.

- **Package** WikiPlot
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

quire_once "[cache.class.php](#)"[line 28]

Require cache class

This class is needed to control the cache.

WikiPlot.php

- **Package** WikiPlot
- **Filesource** [Source Code for this file](#)

function RenderWikiPlot(\$input, \$argv, [\$parser = null]) [*line 152*]

Function Parameters:

- **\$input**
- **\$argv**
- **\$parser**

function wfWikiPlotExtension() [*line 30*]

function WikiPlotDeserializeBoolean(\$value, &\$SetTo) [*line 45*]

Function Parameters:

- *string* **\$value** The string you wish to deserialize.
- *boolean* **&\$SetTo** The variable you want the values parsed to.

Deserialize boolean

Deserializes a boolean value from string, this function is used when you want to deserialize parameters given in the WikiML. If it is impossible to deserialize the value, the output object is not initialized at all.

- **Access** private

function WikiPlotDeserializeColor(\$value, &\$SetTo) [*line 132*]

Function Parameters:

- *string* **\$value** The string you wish to deserialize.
- *array* **&\$SetTo** The variable you want the values parsed to.

Deserialize Color

Deserializes an array representation of a rgb color from string, this function is used when you want to deserialize parameters given in the WikiML. This function can deserialize colors written as "255,255,255" (rgb) or "#000000" (hex). If it is impossible to deserialize the value, the output object is not initialized at all.

- **Access** private

function WikiPlotDeserializeInteger(\$value, &\$SetTo) [[line 110](#)]

Function Parameters:

- *string* **\$value** The string you wish to deserialize.
- *Integer* **&\$SetTo** The variable you want the values parsed to.

Deserialize Integer

Deserializes a integer value from string, this function is used when you want to deserialize parameters given in the WikiML. If it is impossible to deserialize the value, the output object is not initialized at all. Usually this function does nothing at all, just checks to see if the value can be parsed as an integer.

- **Access** private

function WikiPlotDeserializeMixed(\$value, &\$SetTo1, &\$SetTo2) [[line 87](#)]

Function Parameters:

- *string* **\$value** The string you wish to deserialize.
- *integer* **&\$SetTo1** The variable you want the values parsed to.
- *integer* **&\$SetTo2** The variable you want the values parsed to.

Deserialize Coordiante

Deserializes a 2 integers from string, this function is used when you want to deserialize parameters given in the WikiML. If it is impossible to deserialize the value, the output object is not initialized at all.

- **Access** private

function WikiPlotDeserializeString(\$value, &\$SetTo) [*line 67*]

Function Parameters:

- *string* **\$value** The string you wish to deserialize.
- *string* **&\$SetTo** The variable you want the values parsed to.

Deserialize String

Deserializes a string value from string, this function is used when you want to deserialize parameters given in the WikiML. If it is impossible to deserialize the value, the output object is not initialized at all. Usually this function does nothing.

- **Access** private

require_once "[xml.class.php](#)" [*line 19*]

Include xml.class.php

Requires XMLParser to parse xml to plot.

require_once "[cache.class.php](#)" [*line 26*]

Include cache.class.php

Requires Cache to control the cache.

require_once "[PlotClass/plot.class.php](#)" [*line 12*]

Include plot.class.php

Requires PlotClass to render plots.

WikiPlotSettings.php

File used to store settings

This file, is supposed to be manipulated by the user, it contains settings for WikiPlot. Primarily for the caching functionality.

- **Package** WikiPlot
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

ikiPlotCacheAge = 0 [*line 41*]

Max Cache Age

Maximum cache age in days. Delete a file older than... if 0 Cache never expires.

- **Var** Cache age in days.

ikiPlotCachePath = "./cache/" [*line 21*]

Path to the cache

Path to the cache, relative to the DOCUMENT_ROOT.

- **Var** Path relative to DOCUMENT_ROOT
- **See** \$CacheURL

ikiPlotCacheURL = "http://example.com/cache/" [*line 31*]

URL to cache

URL to cache directory define in \$CachePath.

- **Var** absolute url
- **See** \$CachePath

ikiPlotMaxUnusedAge = 14 [*line [50](#)*]

Max Unused Age

Maximun unused age before deletion.

- **Var** Age in days.

xml.class.php

The file contains XMLParser class

This file contains the XMLParser class which parses the XML data to a multidimensional array.

- **Package** WikiPlot
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

Package WikiPlot Classes

Class Cache

[line [41](#)]

Cache controlling class

Class used to control the cache.

- **Package** WikiPlot
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

ring function Cache::CachePath([\$FileName = null]) [line [164](#)]

Function Parameters:

- *string* **\$FileName** Filename you want the path to, shortcut to detecting if file exists.

Get cache Path

Get absolute path to the cache, returns false if FileName exists.

- **Uses** [Cache::FileExist\(\)](#)
- **Access** public

unction Cache::CleanupCache() [*line 52*]

Cleanup the cache

Cleans up the cache by removing old and unused files.

- **Uses** [Cache::CleanupUnused\(\)](#)
- **Uses** [Cache::CleanupMaxAge\(\)](#)
- **Access** public

unction Cache::CleanupMaxAge() [*line 65*]

Cleanup cache from old files

Removes old files from the cache, see LocalSettings.php for settings.

- **Usedby** [Cache::CleanupCache\(\)](#)
- **Access** public

unction Cache::CleanupUnused() [*line 97*]

Cleanup unused files from cache

Removes old unused files from the cache, see LocalSettings.php for settings. This functions identifies files as unused if they haven't been accessed for a long time.

- **Usedby** [Cache::CleanupCache\(\)](#)
- **Access** public

boolean function Cache::FileExist(\$FileName) [*line 129*]

Function Parameters:

- *string* **\$FileName** Filename relative to cache.

Does file exist in cache

Returns true or false depending on whether or not FileName Exist in cache.

- Usedby [Cache::CachePath\(\)](#)
- Usedby [Cache::FileURL\(\)](#)
- Access public

ing function `Cache::FileURL($FileName)` [[line 144](#)]

Function Parameters:

- *string* **\$FileName** Filename relative to cache.

Get file URL

Gets the URL og the given FileName, returns false if the files doesn't exist.

- Uses [Cache::FileExist\(\)](#)
- Access public

Class XMLParser

[[line 55](#)]

XMLParser class

This class parses a given XML data to a multidimensional array by using a user-defined tag. The default tag is <graph>. The exmple below explains how the class works.

```
<?php
$xml_data = "<root>
  <graph color='234,234,233' label='string'>x^2+5</graph>
  <another_tag name='tag'>This tag</another_tag>
  <graph>x^2+5</graph>
</root>"
;
```

```

$xml = new XMLParser($xml_data);
print_r($xml->CreateInputArray());
?>
OUTPUT:
Array
(
    [0] => Array
        (
            [0] => Array
                (
                    [COLOR] => 234,234,233
                    [LABEL] => string
                )
            [1] => x^2+5
        )
    [1] => Array
        (
            [0] => x^2+5
        )
)

```

- **Package** WikiPlot
- **Usedby** [XMLParser::CreateInputArray\(\)](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

XMLParser::\$Attributes

array = [line [112](#)]

Attributes of interested tag

The variable will always be an array whether the interested tag has any attributes or not. If the interested tag has any attribute the \$Attributes variable will be used otherwise it will be ignored.

- **Usedby** [XMLParser::OpenTag\(\)](#)
- **Usedby** [XMLParser::CreateTagArray\(\)](#)
- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Access** private

XMLParser::\$Input

string = [line [72](#)]

XML data given by user

Stores the XML data given by user as it is

- **Usedby** [XMLParser::ExplodeInputData\(\)](#)
- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Access** private

XMLParser::\$Parser

mixed = [line [63](#)]

Created XML Parser

Is a resource handle and referenced to be used by a thor XML functions

- **Usedby** [XMLParser::CloseTag\(\)](#)
- **Usedby** [XMLParser::GetCharData\(\)](#)
- **Usedby** [XMLParser::OpenTag\(\)](#)
- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Access** private

XMLParser::\$Separator

string = [line [147](#)]

The interested tag

The variable is our interested tag. It means the tag that we are interested to find in the given XML data. The way you should define your interested tag is as follows: If your interested tag is <Tag> then you should change the \$Separator variable to XMLParser::\$Separator = "<Tag" not "<Tag>" or something else!

- **Usedby** [XMLParser::ExplodeInputData\(\)](#)
- **Access** public

XMLParser::\$Tag

array = [line [101](#)]

An interested tag in given XML data

The variable stores attribute(s) and data of an interested tag not the tag itself <tag>. For example:

```
If this is an interested tag
< graph color='23,25,200' lable='string' > 2x^3+3x</ graph> the variable
Variable $Tag will look like this:
Array
(
    [0] => Array
        (
            [color] => 23,25,200
            [lable] => string
        )
    [1] => 2x^3+3x
)
```

As you can see the first element in the array is an array and it will always be an array if the interested tag has attribute(s). The second element in the array will be the data of the tag as string. One more thing to be noted is that the array can not contain more than two elements, while one element is possible.

- Used by [XMLParser::CreateTagArray\(\)](#)
- Access private

XMLParser::\$TagData

array = [line [123](#)]

Data of the tag

The variable will store the data of the tag. For example <tag> tag data </tag> \$TagData = "tag data";

- Used by [XMLParser::GetCharData\(\)](#)
- Used by [XMLParser::CreateTagArray\(\)](#)
- Access private

XMLParser::\$Tags

array = [line [133](#)]

All interested tags

The variable will store all the interested tags found in the given XML data.

- **Used by** [XMLParser::CreateInputArray\(\)](#)
- **Used by** [XMLParser::ExplodeInputData\(\)](#)
- **Used by** [XMLParser::XMLParser\(\)](#)
- **Access** private

Constructor *XMLParser* function *XMLParser::XMLParser(\$Data)* [line [171](#)]

Function Parameters:

- *string* **\$Data** XML Input Data from user

Constructor of XMLParser class

The function initializes the following variables: *\$Parser*, *\$Input*, *\$Tags*, *\$Attributes* and *\$Separator*. It makes it possible to use XML Parser within an object by using the function *xml_set_object*. Besides it uses also two more XML Parser Functions *xml_set_element_handler()*, *xml_set_character_data_handler()* and *xml_parser_free()*.

- **Uses** [XMLParser::ExplodeInputData\(\)](#)
- **Uses** [XMLParser::GetCharData\(\)](#)
- **Uses** [XMLParser::OpenTag\(\)](#)
- **Uses** [XMLParser::Parse\(\)](#)
- **Uses** *CloseTag()*
- **Uses** [XMLParser::\\$Tags](#)
- **Uses** [XMLParser::\\$Attributes](#)
- **Uses** [XMLParser::\\$Input](#)
- **Uses** [XMLParser::\\$Parser](#)
- **Access** private

Function *XMLParser::CloseTag(\$Parser, \$Tag)* [line [348](#)]

Function Parameters:

- *mixed* **\$Parser**
- *string* **\$Tag**

Handles end/closing tag

The function gets the end/closing tag using the \$Parser. It is used by xml_set_element_handler() function in the constructor.

- Uses [XMLParser::\\$Parser](#)
- Access private

Graph function XMLParser::CreateInputArray() [*line 367*]

Creates an array containing all parsed XML data

The function runs each and every tag in the \$Tags array through the XMLParser object. The parsed data is then stored in the \$Graph which is returned at the end of the proces.

- Uses [XMLParser](#)
- Uses [XMLParser::\\$Tags](#)

function XMLParser::CreateTagArray() [*line 237*]

Puts parsed data into an array

The function takes the variables \$Attributes and \$TagData and puts them into an array called \$Tag. The first element in the array will be Attribute(s) of the interested tag and the second element will be the data of the tag. If Attribute does not exist the first element will then be the data of the tag.

- Usedby [XMLParser::Parse\(\)](#)
- Uses [XMLParser::\\$TagData](#)
- Uses [XMLParser::\\$Tag](#)
- Uses [XMLParser::\\$Attributes](#)
- Access private

function XMLParser::ExplodeInputData() [*line 261*]

Findes the interested tag in XML Data

The function uses explode() function and the \$Separator to find the interested tag in the given XML Data. When the tags are found it puts them into array called \$Tags.

- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Uses** [XMLParser::\\$Tags](#)
- **Uses** [XMLParser::\\$Separator](#)
- **Uses** [XMLParser::\\$Input](#)
- **Access** private

function XMLParser::GetCharData(\$Parser, \$CharData) [*line 329*]

Function Parameters:

- *mixed* **\$Parser**
- *string* **\$CharData**

Gets data of the tag

The function gets the data of an interesting tag by using the \$Parser. It is used by xml_set_character_data_handler() function in the constructor.

- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Uses** [XMLParser::\\$TagData](#)
- **Uses** [XMLParser::\\$Parser](#)
- **Access** private

function XMLParser::OpenTag(\$Parser, \$Tag, \$Attributes) [*line 298*]

Function Parameters:

- *mixed* **\$Parser**
- *string* **\$Tag**
- *array* **\$Attributes**

Handles attribute(s) of a tag

The function gets the value of the attribute(s) of a tag using the \$Parser. It is used by

xml_set_element_handler() function in the constructor.

- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Uses** [XMLParser::\\$Parser](#)
- **Uses** [XMLParser::\\$Attributes](#)
- **Access** private

Function XMLParser::Parse(\$Data) [[line 214](#)]

Function Parameters:

- *string* **\$Data**

Parses the given XML data

The function uses xml_parse() function from XML Parser Functions in PHP and parses only the first tag in the given XML data and ignores everything else. So you can not use it for multitag XML data. The function also calls CreateTagArray() to generate tag attribute(s) and data to an array.

- **Usedby** [XMLParser::XMLParser\(\)](#)
- **Uses** [XMLParser::CreateTagArray\(\)](#)
- **Access** private

evalmath.class.php

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Filesource** [Source Code for this file](#)

graph.plot.class.php

File containing Graph representation

This file contains a class used as representation of a Graph in plot's. It cannot be used independently, it is a requirement of plot.class.php

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

plot.class.php

File use to draw plots

This file contains a class used to draw plot's. It's dependent on `graph.plot.class.php` and `evalmath.class.php`.

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

quire_once ['evalmath.class.php'](#)line [30](#)]

Includes EvalMath

EvalMath is used to evaluate mathematical expressions in a safe environment.

quire_once ['graph.plot.class.php'](#)line [37](#)]

Includes Graph representation class

Graph is used as a representation of a graph.

test.php

Example/Test

This is an example/test of how to use plot.class.php

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Filesource** [Source Code for this file](#)
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

clude ["plot.class.php"](#)*[line 21]*

Includes plot.class.php for testing

The file tests PlotClass, and must therefor depend on it.

Class EvalMath

[line 97]

Evaluation of expressions

Safe evaluation of mathematical expressions

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Usedby** [Plot::DrawPlots\(\)](#)

valMath::\$f

mixed = array() *[line 103]*

EvalMath::\$fb

```
mixed = array( // built-in functions
    'sin','sinh','arcsin','asin','arcsinh','asinh',
    'cos','cosh','arccos','acos','arccosh','acosh',
    'tan','tanh','arctan','atan','arctanh','atanh',
    'sqrt','abs','ln','log') [line 105]
```

EvalMath::\$last_error

```
mixed = null [line 100]
```

EvalMath::\$suppress_errors

```
mixed = false [line 99]
```

EvalMath::\$v

```
mixed = array('e'=>2.71,'pi'=>3.14) [line 102]
```

EvalMath::\$vb

```
mixed = array('e', 'pi') [line 104]
```

Constructor function EvalMath::EvalMath() [*line 111*]

Function EvalMath::e(\$expr) [*line 117*]

Function Parameters:

- **\$expr**

Function EvalMath::evaluate(\$expr) [*line 121*]

Function Parameters:

- **\$expr**

- **Usedby** [Plot::DrawPlots\(\)](#)

Function EvalMath::funcs() [*line 168*]

Function EvalMath::nfx(\$expr) [*line 178*]

Function Parameters:

- **\$expr**

function EvalMath::pfx(\$tokens, [\$vars = array()]) [[line 304](#)]

Function Parameters:

- **\$tokens**
- **\$vars**

function EvalMath::trigger(\$msg) [[line 366](#)]

Function Parameters:

- **\$msg**

function EvalMath::vars() [[line 161](#)]

Class EvalMathStack

[[line 374](#)]

- **Package** WikiPlot
- **Sub-Package** PlotClass

EvalMathStack::\$count

mixed = 0 [[line 377](#)]

EvalMathStack::\$stack

mixed = array() [[line 376](#)]

function EvalMathStack::last([\$n = 1]) [[line 392](#)]

Function Parameters:

- **\$n**

function EvalMathStack::pop() [[line 384](#)]

function EvalMathStack::push(\$val) [[line 379](#)]

Function Parameters:

- **\$val**

Class Graph

[line [36](#)]

Representation of a graph

Class used to represente graphs on a plot.

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

Graph::\$Color

array = array(0,0,0) [line [88](#)]

Color of the graph

Color of the graph and label, array of the RGB representation of the color. Example:
array(\$Red,\$Green,\$Blue);

- **Usedby** [Plot::DrawPlots\(\)](#)
- **Access** public

Graph::\$EnableLabel

boolean = true [line [66](#)]

Enable label

Enable label, defaults to true, draws label if true.

- **Usedby** [Plot::DrawPlots\(\)](#)
- **Access** public

Graph::\$Exp

string = [line [77](#)]

Expression

The mathematical expression representing the graph.

- **Access** public
- **See** [EvalMath::evaluate\(\)](#)

Graph::\$Label

string = [line [46](#)]

Label of graph

This is the label or legend of the graph and will be shown in the corner of the plot, i the graphs color.

- **Usedby** [Plot::DrawPlots\(\)](#)
- **Access** public

Graph::\$LabelFont

integer = 2 [line [56](#)]

Font of the label

This is the font of the label, defaults to 2, 1-5 are built-in and works as different fontsizes.

- **Usedby** [Plot::DrawPlots\(\)](#)
- **Access** public

ring function Graph::GetHash() [*line [98](#)*]

Get hash

Gets a hash of the graphs parameters. Actually is not a hashsum but just all parameter parsed as one string, this is done to reduce collision risk in Plot::GetHash().

- **Usedby** [Plot::GetHash\(\)](#)
- **Access** private

Class Plot

[*line [50](#)*]

Class used to draw plots

Class containing functions to draw plots to an image.

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** [GNU General Public License](#)

ot::\$BackgroundColor

```
array = array(255,255,255) [line 235]
```

Background color

Color of the background when using auto ImageResource created by GeneratePlot().

- **Usedby** [Plot::GeneratePlot\(\)](#)
- **Access** public

Plot::\$Caption

```
string = null [line 73]
```

Caption of the plot

Caption of the plot, will be shown as text centered on the final plot. Leave this variable as null if no Caption is wanted.

- **Usedby** [Plot::DrawCaption\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **See** [Plot::DrawCaption\(\)](#)
- **Access** public

Plot::\$CaptionFont

```
integer = 5 [line 84]
```

Caption font

Font of the Caption, the fonts 1-5 is built in, and behaves as different sizes.

- **Usedby** [Plot::DrawCaption\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **See** [Plot::DrawCaption\(\)](#)
- **Access** public

Plot::\$EnableAxis

boolean = true [[line 170](#)]

Enable Axis

Defaults to true and draws 2 axis.

- **Usedby** [Plot::GeneratePlot\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **See** [Plot::DrawAxis\(\)](#)
- **Access** public

Plot::\$EnableGrid

boolean = true [[line 180](#)]

Enable Grid

Defaults to true and draws a grid.

- **Usedby** [Plot::GeneratePlot\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **See** [Plot::DrawGrid\(\)](#)
- **Access** public

Plot::\$Graphs

array = array() [[line 61](#)]

Graphs to plot

Array containing list of Graphs to plot.

- **Usedby** [Plot::DrawPlots\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)

- See [Graph](#)
- Access public

ot::\$GridColor

array = array(240,240,240) [[line 196](#)]

Grid color

Defaults to gray, and determines the color of the grid. This is an array of three integers, one for red, green and blue. Where integers have values between 0 and 255.

```
var $Red = 240;
var $Green = 240;
var $Blue = 240;
$this-> GridColor = array($Red,$Green,$Blue);
```

- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawXGrid\(\)](#)
- Usedby [Plot::GetHash\(\)](#)
- See [Plot::DrawGrid\(\)](#)
- Access public

ot::\$GridFont

integer = 1 [[line 206](#)]

Grid font

Font of the grids labels, the fonts 1-5 is built in, and behaves as different sizes.

- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawXGrid\(\)](#)
- Usedby [Plot::GetHash\(\)](#)
- See [Plot::DrawGrid\(\)](#)
- Access public

ot::\$Height

integer = 100 [[line 106](#)]

Height of output image

The width of the output image, in pixels.

- **Usedby** [Plot::GetCoordinatY\(\)](#)
- **Usedby** [Plot::GetImageY\(\)](#)
- **Usedby** [Plot::DrawXGrid\(\)](#)
- **Usedby** [Plot::GeneratePlot\(\)](#)
- **See** [Plot::DrawPlots\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **Access** public

Plot::\$MaxX

integer = 100 [*line* [133](#)]

Maximum X

Maximum X in coordinate space. Together with MinX this variable defines width of the plot in coordinate space. This width may differ from width of the image, the coordinate will be scaled correctly.

- **Usedby** [Plot::DrawAxis\(\)](#)
- **Usedby** [Plot::GetCoordinatX\(\)](#)
- **Usedby** [Plot::GetImageX\(\)](#)
- **Usedby** [Plot::DrawYGrid\(\)](#)
- **Usedby** [Plot::DrawXGrid\(\)](#)
- **See** [Plot::DrawPlots\(\)](#)
- **See** [Plot::\\$MinX](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **Access** public

Plot::\$MaxY

integer = 100 [*line* [159](#)]

Maximum Y

Maximum Y in coordinate space. Together with MinY this variable defines height of the plot in coordinate space. This height may differ from height of the image, the coordinate will be scaled correctly.

- Usedby [Plot::DrawAxis\(\)](#)
- Usedby [Plot::GetCoordinateY\(\)](#)
- Usedby [Plot::GetImageY\(\)](#)
- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawXGrid\(\)](#)
- See [Plot::DrawPlots\(\)](#)
- See [Plot::\\$MinY](#)
- Usedby [Plot::GetHash\(\)](#)
- Access public

Plot::\$MinX

integer = -10 [[line 120](#)]

Minimum X

Minimum X in coordinate space. Together with MaxX this variable defines width of the plot in coordinate space. This width may differ from width of the image, the coordinate will be scaled correctly.

- Usedby [Plot::DrawAxis\(\)](#)
- Usedby [Plot::GetCoordinateX\(\)](#)
- Usedby [Plot::GetImageX\(\)](#)
- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawXGrid\(\)](#)
- See [Plot::DrawPlots\(\)](#)
- See [Plot::\\$MaxX](#)
- Usedby [Plot::GetHash\(\)](#)
- Access public

Plot::\$MinY

integer = -10 [[line 146](#)]

Minimum Y

Minimum Y in coordinate space. Together with MaxY this variable defines height of the plot in coordinate space. This height may differ from height of the image, the coordinate will be scaled correctly.

- Usedby [Plot::DrawAxis\(\)](#)

- Usedby [Plot::GetCoordinatY\(\)](#)
- Usedby [Plot::GetImageY\(\)](#)
- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawXGrid\(\)](#)
- See [Plot::DrawPlots\(\)](#)
- See [Plot::\\$MaxY](#)
- Usedby [Plot::GetHash\(\)](#)
- Access public

Plot::\$Width

integer = 100 [*line 95*]

Width of output image

The width of the output image, in pixels.

- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::GetCoordinatX\(\)](#)
- Usedby [Plot::GetImageX\(\)](#)
- Usedby [Plot::DrawCaption\(\)](#)
- Usedby [Plot::DrawPlots\(\)](#)
- See [Plot::DrawPlots\(\)](#)
- Usedby [Plot::GetHash\(\)](#)
- Usedby [Plot::GeneratePlot\(\)](#)
- Access public

Plot::\$XGridSpace

integer = null [*line 216*]

X grid space

Distance between grids on the x axis in coordinate space. Defaults to null, leave it null, if you want autogenerated gridspace.

- Usedby [Plot::GetXGridSpace\(\)](#)
- Usedby [Plot::GetHash\(\)](#)
- See [Plot::GetXGridSpace\(\)](#)
- Access public

Plot::\$YGridSpace

integer = null [[line 226](#)]

Y grid space

Distance between grids on the y axis in coordinate space. Defaults to null, leave it null, if you want autogenerated gridspace.

- **Usedby** [Plot::GetYGridSpace\(\)](#)
- **Usedby** [Plot::GetHash\(\)](#)
- **See** [Plot::GetYGridSpace\(\)](#)
- **Access** public

Function Plot::DisplayPlot([\$DisplayType = "png"], [\$ImageResource = null], [\$ChangeSize = false]) [[line 721](#)]
Function Parameters:

- *string* **\$DisplayType** Type of image to view (png|jpeg|gif).
- *ImageResource* **\$ImageResource** Defaults to null, will generate empty ImageResource.
- *Boolean* **\$ChangeSize** May we change the size of the plot to fit given ImageResource?

Display plot as image

Displays plot as image on the page. This makes current http-request return an image. You can set the DisplayType to png, gif or jpeg. Defaults to png, gif not recommended. Note: this changes the current http-request mimetype to the respective image mimetype.

- **Uses** [Plot::GeneratePlot\(\)](#)
- **Access** public

Function Plot::DrawAxis(&\$ImageResource) [[line 690](#)]
Function Parameters:

- *ImageResource* **&\$ImageResource** ImageResource representation of the plot.

Draw axis

Draw both x and y axis to the plot.

- Uses [Plot::GetImageX\(\)](#)
- Uses [Plot::GetImageY\(\)](#)
- Usedby [Plot::GeneratePlot\(\)](#)
- Uses [Plot::\\$MinY](#)
- Uses [Plot::\\$MinX](#)
- Uses [Plot::\\$MaxX](#)
- Uses [Plot::\\$MaxY](#)
- Access private

Function `Plot::DrawCaption(&$ImageResource)` [[line 439](#)]

Function Parameters:

- *ImageResource* **&\$ImageResource** ImageResource representation of the plot.

Draw caption to ImageResource

Draws the caption to an ImageResource representation of the plot.

- Usedby [Plot::GeneratePlot\(\)](#)
- Uses [Plot::\\$Width](#)
- Uses [Plot::\\$CaptionFont](#)
- Uses [Plot::\\$Caption](#)
- Access private

Function `Plot::DrawGrid(&$ImageResource)` [[line 540](#)]

Function Parameters:

- *ImageResource* **&\$ImageResource** ImageResource representation of the plot.

Draw grids

Draws both x and y grid, using DrawXGrid() and DrawYGrid().

- **Usedby** [Plot::GeneratePlot\(\)](#)
- **Uses** [Plot::DrawYGrid\(\)](#)
- **Uses** [Plot::DrawXGrid\(\)](#)
- **Access** private

Function Plot::DrawPlots(&\$ImageResource) [[line 372](#)]

Function Parameters:

- *ImageResource* **&\$ImageResource** ImageResource representation of the plot.

Get ImageResource of the plot

Generates ImageResource representation of the plot.

- **Uses** [Graph::\\$EnableLabel](#)
- **Uses** [Graph::\\$Color](#)
- **Uses** [Graph::\\$Label](#)
- **Uses** [Graph::\\$LabelFont](#)
- **Usedby** [Plot::GeneratePlot\(\)](#)
- **Uses** [Plot::GetImageY\(\)](#)
- **Uses** [Plot::GetImageX\(\)](#)
- **Uses** [Plot::\\$Width](#)
- **Uses** [Plot::\\$Graphs](#)
- **Uses** [EvalMath](#)
- **Uses** [EvalMath::evaluate\(\)](#)
- **Uses** [Plot::GetCoordinateX\(\)](#)
- **Access** private

Function Plot::DrawXGrid(&\$ImageResource) [[line 565](#)]

Function Parameters:

- *ImageResource* **&\$ImageResource** ImageResource representation of the plot.

Draws x-grid

Drawing X grid on the plot.

- Uses [Plot::GetImageY\(\)](#)
- Uses [Plot::GetImageX\(\)](#)
- Uses [Plot::GetXGridSpace\(\)](#)
- Uses [Plot::ShortNumber\(\)](#)
- Usedby [Plot::DrawGrid\(\)](#)
- Uses [Plot::\\$MinY](#)
- Uses [Plot::\\$MinX](#)
- Uses [Plot::\\$GridFont](#)
- Uses [Plot::\\$GridColor](#)
- Uses [Plot::\\$Height](#)
- Uses [Plot::\\$MaxX](#)
- Uses [Plot::\\$MaxY](#)
- Access private

Function `Plot::DrawYGrid(&$ImageResource)` [line [630](#)]

Function Parameters:

- *ImageResource* **&\$ImageResource** ImageResource representation of the plot.

Draws y-grid

Drawing y grid on the plot.

- Uses [Plot::GetImageY\(\)](#)
- Uses [Plot::GetImageX\(\)](#)
- Uses [Plot::GetYGridSpace\(\)](#)
- Uses [Plot::ShortNumber\(\)](#)
- Usedby [Plot::DrawGrid\(\)](#)
- Uses [Plot::\\$Width](#)
- Uses [Plot::\\$MinY](#)
- Uses [Plot::\\$GridFont](#)
- Uses [Plot::\\$GridColor](#)
- Uses [Plot::\\$MaxX](#)
- Uses [Plot::\\$MaxY](#)
- Uses [Plot::\\$MinX](#)
- Access private

ImageResource function `Plot::GeneratePlot([$ImageResource = null], [$ChangeSize = false])` [*line* [298](#)]

Function Parameters:

- *ImageResource* **\$ImageResource** Defaults to null, will generate empty ImageResource.
- *Boolean* **\$ChangeSize** May we change the size of the plot to fit given ImageResource?

Get ImageResource of the plot

Generates ImageResource representation of the plot.

- Uses [Plot::DrawPlots\(\)](#)
- Uses [Plot::DrawGrid\(\)](#)
- Uses [Plot::\\$EnableGrid](#)
- Usedby [Plot::DisplayPlot\(\)](#)
- Usedby [Plot::SaveAs\(\)](#)
- Uses [Plot::DrawCaption\(\)](#)
- Uses [Plot::DrawAxis\(\)](#)
- Uses [Plot::\\$BackgroundColor](#)
- Uses [Plot::\\$EnableAxis](#)
- Uses [Plot::\\$Height](#)
- Uses [Plot::\\$Width](#)
- Access public

integer function `Plot::GetCoordinatX($x)` [*line* [780](#)]

Function Parameters:

- *integer* **\$x** X image coordinat to be converted.

Convert to coordinate space

Converts an x image position to x coordinate position. Coordinate space may differ from Image space, if `Width!= (MaxX-MinX)`.

- Usedby [Plot::DrawPlots\(\)](#)
- Uses [Plot::\\$Width](#)

- Uses [Plot::\\$MinX](#)
- Uses [Plot::\\$MaxX](#)
- Access private

integer function Plot::GetCoordinatY(\$y) [*line* [797](#)]

Function Parameters:

- integer \$y Y image coordinat to be converted.

Convert to coordinate space

Converts an y image position to y coordinate position. Coordinate space may differ from Image space, if Height!= (MaxY-MinY).

- Uses [Plot::\\$MinY](#)
- Uses [Plot::\\$MaxY](#)
- Uses [Plot::\\$Height](#)
- Access private

string function Plot::GetHash() [*line* [260](#)]

Generate hash

Generates a unique hashsum (md5) for the plot, generated from all parameters.

- Uses [Plot::\\$MinY](#)
- Uses [Plot::\\$MinX](#)
- Uses [Plot::\\$MaxY](#)
- Uses [Plot::\\$Width](#)
- Uses [Plot::\\$XGridSpace](#)
- Uses [Graph::GetHash\(\)](#)
- Uses [Plot::\\$YGridSpace](#)
- Uses [Plot::\\$MaxX](#)
- Uses [Plot::\\$Height](#)
- Uses [Plot::\\$EnableAxis](#)
- Uses [Plot::\\$CaptionFont](#)
- Uses [Plot::\\$EnableGrid](#)
- Uses [Plot::\\$Graphs](#)
- Uses [Plot::\\$GridFont](#)
- Uses [Plot::\\$GridColor](#)

- Uses [Plot::\\$Caption](#)

integer function Plot::GetImageX(\$x) [*line [814](#)*]

Function Parameters:

- integer **\$x** X coordinat to be converted.

Convert to image space

Converts an x in coordinate space to x image position. Coordinate space may differ from Image space, if Width!= (MaxX-MinX).

- Usedby [Plot::DrawXGrid\(\)](#)
- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawAxis\(\)](#)
- Usedby [Plot::DrawPlots\(\)](#)
- Uses [Plot::\\$Width](#)
- Uses [Plot::\\$MaxX](#)
- Uses [Plot::\\$MinX](#)
- Access private

integer function Plot::GetImageY(\$y) [*line [831](#)*]

Function Parameters:

- integer **\$y** Y coordinat to be converted.

Convert to image space

Converts an y in coordinate space to y image position. Coordinate space may differ from Image space, if Height!= (MaxY-MinY).

- Usedby [Plot::DrawXGrid\(\)](#)
- Usedby [Plot::DrawYGrid\(\)](#)
- Usedby [Plot::DrawAxis\(\)](#)
- Usedby [Plot::DrawPlots\(\)](#)

- Uses [Plot::\\$MinY](#)
- Uses [Plot::\\$Height](#)
- Uses [Plot::\\$MaxY](#)
- Access private

integer function Plot::GetXGridSpace() [*line [490](#)*]

Get X grid space

Returns X grid space, either calculated or from given value if given one.

- Usedby [Plot::DrawXGrid\(\)](#)
- Uses [Plot::\\$XGridSpace](#)
- Access private

integer function Plot::GetYGridSpace() [*line [515](#)*]

Get Y grid space

Returns Y grid space, either calculated or from given value if given one. If it is to be calculated, it is calculated the same way as x axes!

- Usedby [Plot::DrawYGrid\(\)](#)
- Uses [Plot::\\$YGridSpace](#)
- Access private

function Plot::SaveAs(\$Path, [\$SaveAs = "png"], [\$ImageResource = null], [\$ChangeSize = false]) [*line [752](#)*]

Function Parameters:

- *string* **\$Path** Path of file to save.
- *string* **\$SaveAs** Filetype definition (png|jpeg|gif).
- *ImageResource* **\$ImageResource** Defaults to null, will generate empty ImageResource.
- *Boolean* **\$ChangeSize** May we change the size of the plot to fit given ImageResource?

Save plot to image

Saves the plot to an image. You can set the SaveAs to a file type: png, gif or jpeg, defaults to png.

- **Uses** [Plot::GeneratePlot\(\)](#)
- **Access** public

ing function `Plot::ShortNumber($Number, [$MaxLen = 7])` [[line 465](#)]

Function Parameters:

- *integer* **\$Number** The number you wish to shorten.
- *integer* **\$MaxLen** The maximum length of the output default to 7.

Generates short numbers

Rewrites numbers into scientific notation, with a certain maximum length.

Example: `ShortNumber(501000000) == 5.01e8`

- **Usedby** [Plot::DrawYGrid\(\)](#)
- **Usedby** [Plot::DrawXGrid\(\)](#)
- **Access** private

Appendices

Appendix A - Class Trees

Package WikiPlot

Cache

[Cache](#)

EvalMath

[EvalMath](#)

EvalMathStack

[EvalMathStack](#)

Graph

[Graph](#)

Plot

[Plot](#)

XMLParser

[XMLParser](#)

Appendix C - Source Code

Package WikiPlot

File Source for cache.class.php

Documentation for this file is available at [cache.class.php](http://www.phpdoc.org/projects/phpdocu/cache.class.php)

```
<?php
/*
Copyright (C) 2006 by the WikiPlot project authors (See http://code.google.com/p/WikiPlot).

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

/**
 * File used to control cache
 *
 * This file provides functions to control the content of the cache.
 * This file is made to make the software more maintain able, and as an interface to the cache for third party
 developers.
 */
 * @package WikiPlot
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */

/**
 * Require local settings
 *
 * This file is needed to control the cache correctly.
 */
require_once("WikiPlotSettings.php"                );

/**
 * Cache controlling class
 *
 * Class used to control the cache.
 */
 * @package WikiPlot
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */
class Cache
{
    /**
     *Cleanup the cache
     *
     *Cleans up the cache by removing old and unused files.
     */
    * @access public
    * @uses CleanupMaxAge()
    * @uses CleanupUnused()
    */
    function CleanupCache()
    {
        $this-> CleanupMaxAge();
        $this-> CleanupUnused();
    }

    /**
     * Cleanup cache from old files
     *
     * Removes old files from the cache, see LocalSettings.php for settings.
     */
}
```

```

*
* @access public
*/
function CleanupMaxAge()
{
    $CachePath = $_SERVER["DOCUMENT_ROOT"] . WikiPlotCachePath;
    if ($cache = opendir($CachePath))
    {
        $MaxFileAge = time() - (WikiPlotCacheAge * 24 * 60 * 60);
        while (false !== ($file = readdir($cache)))
        {
            $FileAge = filetime($CachePath . "/" . $file);
            if($FileAge > $MaxFileAge)
            {
                if(!unlink($CachePath . "/" . $file))
                {
                    //TODO: throw some error!
                }
            }
        }
        closedir($cache);
    }else{
        //TODO: throw some error!
    }
}

/**
 * Cleanup unused files from cache
 *
 * Removes old unused files from the cache, see LocalSettings.php for settings.
 * This functions indentifies files as unused if they havn't been accessed for a long time.
 *
 * @access public
*/
function CleanupUnused()
{
    $CachePath = $_SERVER["DOCUMENT_ROOT"] . WikiPlotCachePath;
    if ($cache = opendir($CachePath))
    {
        $MaxFileAge = time() - (WikiPlotMaxUnusedAge * 24 * 60 * 60);
        while (false !== ($file = readdir($cache)))
        {
            $FileAge = filetime($CachePath . "/" . $file);
            if($FileAge > $MaxFileAge)
            {
                if(!unlink($CachePath . "/" . $file))
                {
                    //TODO: throw some error!
                }
            }
        }
        closedir($cache);
    }else{
        //TODO: throw some error!
    }
}

/**
 * Does file exist in cache
 *
 * Returns true or false depending on whether or not FileName Exist in cache.
 *
 * @access public
 * @param string $FileName Filename relative to cache.
 * @return boolean Whether or not FileName exist.
*/
function FileExist($FileName)
{
    return file_exists($_SERVER["DOCUMENT_ROOT"] . WikiPlotCachePath . $FileName);
}

/**
 * Get file URL
 *
 * Gets the URL og the given FileName, returns false if the files doen't exist.
 *
 * @access public
 * @uses FileExist()
 * @param string $FileName Filename relative to cache.

```

```

2  * @return string Returns the URL of the file.
3  */
4  function FileURL($FileName)
5  {
6      if($this-> FileExist($FileName))
7      {
8          return WikiPlotCacheURL . "/" . $FileName;
9      }else{
10         return false;
11     }
12 }
13
14 /**
15  * Get cache Path
16  *
17  * Get absolute path to the cache, returns false if FileName exists.
18  *
19  * @access public
20  * @uses FileExist()
21  * @param string $FileName Filename you want the path to, shortcut to detecting if file exists.
22  * @return string Path to the cache, false if FileName exists.
23  */
24 function CachePath($FileName = null)
25 {
26     if(!is_null($FileName))
27     {
28         if($this-> FileExist($FileName))
29         {
30             return false;
31         }else{
32             return $_SERVER["DOCUMENT_ROOT"] . WikiPlotCachePath . $FileName;
33         }
34     }else{
35         return $_SERVER["DOCUMENT_ROOT"] . WikiPlotCachePath;
36     }
37 }
38 }
39
40 ?>

```

File Source for CleanupCache.php

Documentation for this file is available at [CleanupCache.php](#)

```
<?php
/*
Copyright (C) 2006 by the WikiPlot project authors (See http://code.google.com/p/WikiPlot).

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

/**
 * File used to clear the cache
 *
 * This file is supposed to be called as a cron script, to clear the cache on a regular basis.
 *
 * @package WikiPlot
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */

/**
 * Require cache class
 *
 * This class is needed to control the cache.
 */
require_once("cache.class.php"           );

//Create instance of cache
$Cache = new Cache;

//Cleanup cache
$Cache-> CleanupCache();
?>
```

File Source for WikiPlot.php

Documentation for this file is available at [WikiPlot.php](http://www.phpdoc.org/projects/phpdocu/WikiPlot.php)

```
<?php
/**
 * @package WikiPlot
 */

/**
 *Include plot.class.php
 *
 *Requires PlotClass to render plots.
 */
require_once("PlotClass/plot.class.php"           );

/**
 *Include xml.class.php
 *
 *Requires XMLParser to parse xml to plot.
 */
require_once("xml.class.php"                      );

/**
 *Include cache.class.php
 *
 *Requires Cache to control the cache.
 */
require_once("cache.class.php"                   );

$wgExtensionFunctions[] = "wfWikiPlotExtension"   ;

function wfWikiPlotExtension() {
    global $wgParser;
    $wgParser->    setHook( "wikipLOT"           , "RenderWikiPlot"           );
}

/**
 *Deserialize boolean
 *
 *Deserializes a boolean value from string, this function is used when you want to deserialize parameters given
the WikiML.
 *If it is impossible to deserialize the value, the output object is not initialized at all.
 *
 *@access private
 *@param string $value The string you wish to deserialize.
 *@param boolean &$amp;$SetTo The variable you want the values parsed to.
 */
function WikiPlotDeserializeBoolean($value,&    $SetTo)
{
    if($value == "true"           )
    {
        $SetTo = true;
    }
    elseif($value == "false"           )
    {
        $SetTo = false;
    }
}

/**
 *Deserialize String
 *
 *Deserializes a string value from string, this function is used when you want to deserialize parameters given in
e WikiML.
 *If it is impossible to deserialize the value, the output object is not initialized at all. Usually this function
es nothing.
 *
 *@access private
 *@param string $value The string you wish to deserialize.
```

```

*@param string &$SetTo The variable you want the values parsed to.
*/
function WikiPlotDeserializeString($value,& $SetTo)
{
    if(is_string($value))
    {
        $SetTo = $value;
    }
}

/**
*Deserialize Coordiante
*
*Deserializes a 2 integers from string, this function is used when you want to deserialize parameters given in
e WikiML.
*If it is impossible to deserialize the value, the output object is not initialized at all.
*
*@access private
*@param string $value The string you wish to deserialize.
*@param integer &$SetTo1 The variable you want the values parsed to.
*@param integer &$SetTo2 The variable you want the values parsed to.
*/
function WikiPlotDeserializeMixed($value,& $SetTo1,& $SetTo2)
{
    if(!is_null($value))
    {
        $values = explode(";", $value, 2);
        if(is_numeric($values[0])&& is_numeric($values[1]))
        {
            $SetTo1 = $values[0];
            $SetTo2 = $values[1];
        }
    }
}

/**
*Deserialize Integer
*
*Deserializes a integer value from string, this function is used when you want to deserialize parameters given
the WikiML.
*If it is impossible to deserialize the value, the output object is not initialized at all. Usually this function
es nothing at all, just checks to see if the value can be parsed as an integer.
*
*@access private
*@param string $value The string you wish to deserialize.
*@param Integer &$SetTo The variable you want the values parsed to.
*/
function WikiPlotDeserializeInteger($value,& $SetTo)
{
    if(!is_null($value))
    {
        if(is_numeric($value))
        {
            $SetTo = $value;
        }
    }
}

/**
*Deserialize Color
*
*Deserializes an array representation of a rgb color from string, this function is used when you want to
erialize parameters given in the WikiML.
*This function can deserialize colors written as "255,255,255" (rgb) or "#000000" (hex).
*If it is impossible to deserialize the value, the output object is not initialized at all.
*
*@access private
*@param string $value The string you wish to deserialize.
*@param array &$SetTo The variable you want the values parsed to.
*/
function WikiPlotDeserializeColor($value,& $SetTo)
{
    if(!is_null($value))
    {
        $values = explode(";", $value, 3);
        if(is_numeric($values[0])&& is_numeric($values[1])&& is_numeric($values[2]))
        {
            $SetTo = array($values[0], $values[1], $values[2]);
        }
    }
}

```



```

elseif(strpos($value,"#"
    ))
{
    $red = hexdec(substr($val, 1, 2));
    $green = hexdec(substr($val, 3, 2));
    $blue = hexdec(substr($val, 5, 2));
    $SetTo = array($red,$green,$blue);
}
}
}

# The callback function for rendering plot
function RenderWikiPlot( $input, $argv, $parser = null ) {
    if (!$parser) $parser =& $GLOBALS['wgParser'];
    # $argv is an array containing any arguments passed to the
    # extension like <example argument="foo" bar>..
    # Put this on the sandbox page: (works in MediaWiki 1.5.5)
    # <example argument="foo" argument2="bar">Testing text **example** in between
    new tags</example>

    $Plot = new Plot();

    WikiPlotDeserializeBoolean($argv["grid"]            ,,$Plot-> EnableGrid);
    WikiPlotDeserializeBoolean($argv["axis"]            ,,$Plot-> EnableAxis);

    WikiPlotDeserializeString($argv["caption"]          ,,$Plot-> Caption);

    WikiPlotDeserializeMixed($argv["xspan"]             ,,$Plot-> MinX,$Plot-> MaxX);
    WikiPlotDeserializeMixed($argv["yspan"]             ,,$Plot-> MinY,$Plot-> MaxY);
    WikiPlotDeserializeMixed($argv["gridspace"]         ,,$Plot-> XGridSpace,$Plot-> YGridSpace);

    WikiPlotDeserializeInteger($argv["height"]          ,,$Plot-> Height);
    WikiPlotDeserializeInteger($argv["width"]           ,,$Plot-> Width);
    WikiPlotDeserializeInteger($argv["captionfont"]     ,,$Plot-> CaptionFont);
    WikiPlotDeserializeInteger($argv["gridfont"]        ,,$Plot-> GridFont);

    WikiPlotDeserializeColor($argv["gridcolor"]         ,,$Plot-> GridColor);

    /*
    //TODO: remeber to use width and height as x- and yspan if x-/yspan isn't provided.
    WikiML specification
    <wikiplot grid="true" caption="Caption text" axis="true" xspan="-
    ;10" yspan="-10;10" height="20" width="20" gridspace="x;y"
    ptionfont="5" gridfont="1" gridcolor="200,200,200">
    <graph label="Graph 1" color="0,0,255" font="3">5x^3</graph>
    <graph label="Graph 2" color="#449933" font="3">2x^2</graph>
    </wikiplot>
    */

    //Parsing Xml
    $XmlParser = new XMLParser($input);
    $Graphs = $XmlParser-> CreateInputArray();

    foreach($Graphs as $Graph)
    {
        $G = new Graph;
        if(!is_array($Graph[1]))
        {
            $G-> Exp = $Graph[1];
            WikiPlotDeserializeString($Graph[0]["label"] ,,$G-> Label);
            WikiPlotDeserializeColor($Graph[0]["color"] ,,$G-> Color);
        }else{
            $G-> Exp = $Graph[0];
        }
        array_push($Plot-> Graphs,$G);
    }

    //Render the plot

    //Get instance of cache
    $cache = new cache();

    //Url of the current plot
    $PlotURL = "" ;

    $PlotFileName = $Plot-> GetHash() . ".png" ;
    if(!$cache-> FileExist($PlotFileName))
    {
        $Plot-> SaveAs($cache-> CachePath($PlotFileName));
    }else{
        $PlotURL = $cache-> FileURL($PlotFileName);
    }
}

```

```

8     }
9
0     $output = "      <      a href='$PlotURL' class='image' title='See the plot'><      img
c='$PlotURL'></      a>      "      ;
1
2     /*
3     //Render as wikitext:
4     //To use external images this must be enabled: $wgAllowExternalImages = true; remeber to inform user.
5     $localParser = new Parser();
6     $output = $localParser->parse("<Test:This is rendered wikitext", $parser->mTitle, $parser-
Options); //Once we test this, remember to check if adding parameters true, false OR false, true OR false OR
ue... see http://meta.wikimedia.org/wiki/MediaWiki_extensions_FAQ for more information on wikitext rendering in
tensions
7     $text = $output->getText();
8     */
9     return $output;
0 }
1
2 ?>

```

File Source for WikiPlotSettings.php

Documentation for this file is available at [WikiPlotSettings.php](#)

```
<?php
/**
 * File used to store settings
 *
 * This file, is supposed to be manipulated by the user, it contains settings for WikiPlot. Primarily for the
 * caching functionality.
 *
 * @package WikiPlot
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */

/**
 * Path to the cache
 *
 * Path to the cache, relative to the DOCUMENT_ROOT.
 *
 * @see $CacheURL
 * @var string Path relative to DOCUMENT_ROOT
 */
define("WikiPlotCachePath"          , "./cache/"          );

/**
 * URL to cache
 *
 * URL to cache directory define in $CachePath.
 *
 * @see $CachePath
 * @var string absolute url
 */
define("WikiPlotCacheURL"            , "http://example.com/cache/"            );

/**
 * Max Cache Age
 *
 * Maximum cache age in days. Delete a file older than...
 * if 0 Cache never expires.
 *
 * @var Integer Cache age in days.
 */
define("WikiPlotCacheAge"            , 0);

/**
 * Max Unused Age
 *
 * Maximun unused age before deletion.
 *
 * @var Integer Age in days.
 */
define("WikiPlotMaxUnusedAge"        , 14);
?>
```

File Source for xml.class.php

Documentation for this file is available at [xml.class.php](http://www.phpdoc.org/projects/phpdoc/xml.class.php)

```
<?php
/**
 * The file contains XMLParser class
 *
 * This file contains the XMLParser class which parses the XML data to
 * a multidimensional array.
 *
 * @package WikiPlot
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */

/**
 * XMLParser class
 *
 * This class parses a given XML data to a multidimensional array by using
 * a user-defined tag. The default tag is <graph>. The example below explains
 * how the class works.
 * <code>
 * <?php
 * $xml_data = "<root>
 *             <graph color='234,234,233' label='string'>x^2+5</graph>
 *             <another_tag name='tag'>This tag</another_tag>
 *             <graph>x^2+5</graph>
 *             </root>";
 *
 * $xml = new XMLParser($xml_data);
 * print_r($xml->CreateInputArray());
 * ?>
 * OUTPUT:
 * Array
 * (
 *     [0] => Array
 *         (
 *             [0] => Array
 *                 (
 *                     [COLOR] => 234,234,233
 *                     [LABEL] => string
 *                 )
 *             [1] => x^2+5
 *         )
 *     [1] => Array
 *         (
 *             [0] => x^2+5
 *         )
 * )
 * </code>
 *
 * @package WikiPlot
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */
class XMLParser {

    /**
     * Created XML Parser
     *
     * Is a resource handle and referenced to be used by athor XML functions
     * @access private
     */
    var $Parser;

    /**
     * XML data given by user
     *
     * Stores the XML data given by user as it is
     */
}
```

```

*
* @var string
* @access private
*/
var $Input;
/**
 * An interested tag in given XML data
 *
 * The variable stores attribute(s) and data of an interested tag not
 * the tag it selv <tag>. For example:
 * <code>
 * If this is an interested tag
 * <graph color='23,25,200' lable='string'>2x^3+3x</graph> the variable
 * Variable $Tag will look like this:
 * Array
 * (
 *     [0] => Array
 *         (
 *             [color] => 23,25,200
 *             [lable] => string
 *         )
 *     [1] => 2x^3+3x
 * )
 * </code>
 * As you can see the first element in the array is an array and it
 * will always be an array if the interested tag has attribute(s). The second
 * element in the array will be the data of the tag as string. One more thing
 * to be notes is that the array can not contain more then two elements, while one
 * element is possible.
 *
 * @var array
 * @access private
*/
var $Tag;
/**
 * Attributes of interested tag
 *
 * The variable will always be an array whether the interested tag has any
 * attributes or not. If the interested tag has any attribute the $Attributes
 * variable will be used otherwise it will be ignored.
 *
 * @var array
 * @access private
*/
var $Attributes;
/**
 * Data of the tag
 *
 * The variable will store the data of the tag. For example
 * <tag> tag data </tag>
 * $TagData = "tag data";
 *
 * @var array
 * @access private
*/
var $TagData;
/**
 * All interested tags
 *
 * The variable will store alle the interested tags found in the
 * given XML data.
 *
 * @var array
 * @access private
*/
var $Tags;
/**
 * The interested tag
 *
 * The variable is our iterested tag. It means the tag that we are
 * interested to finde in the given XML data.
 * The way you should definde your interested tag is as follows:
 * If your interested tag is <Tag> than you should change the
 * $Separator variable to XMLParser::Separator = "<Tag" not "<Tag>"
 * or something else!
 *
 * @var string
 * @access public
*/
var $Separator;

```

```

8  /**
9  * Constructor of XMLParser class
10 *
11 * The function initializes the following variables:
12 * $Parser, $Input, $Tags, $Attributes and $Separator.
13 * It makes it possible to use XML Parser within an object
14 * by using the function xml_set_object. Besides it uses also
15 * two more XML Parser Functions xml_set_element_handler(),
16 * xml_set_character_data_handler() and xml_parser_free().
17 *
18 * @access private
19 * @param string $Data XML Input Data from user
20 * @return XMLParser
21 * @uses $Parser
22 * @uses $Input
23 * @uses $Tags
24 * @uses $Attributes
25 * @uses ExplodeInputData()
26 * @uses Parse()
27 * @uses OpenTag()
28 * @uses CloseTag()
29 * @uses GetCharData()
30 */
31 function XMLParser($Data)
32 {
33     //Initialize $Parser and creat an XML Parser to use later on
34     $this-> Parser = xml_parser_create();
35     //Initialize $Input and set it equal to $Data (XML from user)
36     $this-> Input = $Data;
37     //Initialize $Tags to be an array
38     $this-> Tags = array();
39     //Initialize $Attributes to be an array
40     $this-> Attributes = array();
41     //Initialize $Separator to be an array
42     $this-> Separator = "<graph" ;
43
44     //Set XML Parser to use it within object
45     xml_set_object($this-> Parser, $this);
46     //Set up start and end element handlers for the parser
47     xml_set_element_handler($this-> Parser, "OpenTag" , "CloseTag" );
48     //Set up character data handler for the parser
49     xml_set_character_data_handler($this-> Parser, "GetCharData" );
50
51     //Call ExplodeInputData() to get the interested tags
52     $this-> ExplodeInputData();
53
54     //Call Parse() to parse the $Input
55     $this-> Parse($this-> Input);
56
57     //Free the XML parser to later use
58     xml_parser_free($this-> Parser);
59 }
60
61 /**
62 * Parses the given XML data
63 *
64 * The function uses xml_parse() function from XML Parser Functions in PHP
65 * and parses only the first tag in the given XML data and ignores
66 * everything else. So you can not use it for multitag XML data.
67 * The function also calls CreateTagArray() to generate tag attribute(s)
68 * and data to an array.
69 *
70 * @access private
71 * @param string $Data
72 * @uses CreateTagArray()
73 */
74 function Parse($Data)
75 {
76     //Parse XML Data using the $Parser
77     xml_parse($this-> Parser, $Data);
78     //Put returned values (Attribute(s) and TagData)
79     //from XML praser into an array called $Tag
80     $this-> CreateTagArray();
81 }
82
83 /**
84 * Puts parsed data into an array
85 *
86 * The function takes the variables $Attributes and $TagData and
87 * puts them into an array called $Tag. The first element in the

```

```

8  * array will be Attribute(s) of the interested tag and the second
9  * element will be the data of the tag. If Attribute does not exist
0  * the first element will then be the data of the tag.
1  *
2  * @access private
3  * @uses $Attributes
4  * @uses $TagData
5  * @uses $Tag
6  */
7  function CreateTagArray()
8  {
9      if (!empty($this-> Attributes) && !empty($this-> TagData))
10     {
11         $this-> Tag = array($this-> Attributes, $this-> TagData);
12     }
13     else
14     {
15         $this-> Tag = array($this-> TagData);
16     }
17 }
18
19 /**
20  * Findes the interested tag in XML Data
21  *
22  * The function uses explode() function and the $Separator to finde
23  * the interested tag in the given XML Data. When the tags are found
24  * it puts them into array called $Tags.
25  *
26  * @access private
27  * @uses $Separator
28  * @uses $Input
29  * @uses $Tags
30  */
31 function ExplodeInputData()
32 {
33     //Split the given XML data by using $Separator
34     $InterestedTags = explode($this-> Separator, $this-> Input);
35
36     //Go through the array containing the interesting tags
37     //NOTICE: $i must be = 1 because the array contains
38     //nothing on 0 position
39     //NOTICE: $ must be < lenght of the array and not <= because
40     //the last element in the array is not interesting.
41     for ($i=1; $i < count($InterestedTags); $i++)
42     {
43         //Put the $Separator into the tag
44         //(the separator vanishes when exploding the data)
45         //fx. If the separator is <tag>. The following will take place.
46         //<tag>Hello</tag> will be exploded by <tag> and
47         //returned as >Hello</tag>. To complete the tag
48         //we put the separator back on place. <tag + >Hello</tag>
49         //this will return the complete tag = <tag>Hello</tag>
50         array_push($this-> Tags, $this-> Separator . $InterestedTags[$i]);
51     }
52 }
53
54 /**
55  * Handles attribute(s) of a tag
56  *
57  * The function gets the value of the attribute(s) of a tag using the
58  * $Parser. It is used by xml_set_element_handler() function in the
59  * constructor.
60  *
61  * @access private
62  * @param mixed $Parser
63  * @param string $Tag
64  * @param array $Attributes
65  * @uses $Parser
66  * @uses $Attributes
67  */
68 function OpenTag($Parser, $Tag, $Attributes)
69 {
70     //Check whether $Attributes is an array and is not an empty array
71     if (is_array($Attributes) && count($Attributes) > 0)
72     {
73         //Put $this->Attributes equal to $Attributes while changing the
74         //case of its key(s) to lowercase. The case of the key(s) is
75         //important due to avoid error later on.
76         $this-> Attributes = array_change_key_case($Attributes, CASE_LOWER);
77     }
78 }

```

```

8         else
9         {
0             // $this->Attributes will be an empty array() which is ignored
1             // when adding it to the general array which is returned by the
2             // class!
3         }
4     }
5
6 /**
7  * Gets data of the tag
8  *
9  * The function gets the data of an interesting tag by using the
0  * $Parser. It is used by xml_set_character_data_handler() function
1  * in the constructor.
2  *
3  * @access private
4  * @param mixed $Parser
5  * @param string $CharData
6  * @uses $Parser
7  * @uses $TagData
8  */
9 function GetCharData($Parser, $CharData)
0 {
1     // Set $this->TagData equal to $CharData
2     // for later use.
3     $this-> TagData = $CharData;
4 }
5
6 /**
7  * Handles end/closing tag
8  *
9  * The function gets the end/closing tag using the $Parser.
0  * It is used by xml_set_element_handler() function in the
1  * constructor.
2  *
3  * @access private
4  * @param mixed $Parser
5  * @param string $Tag
6  * @uses $Parser
7  */
8 function CloseTag($Parser, $Tag)
9 {
0     // Have nothing to do! :(
1     // But must be present.
2 }
3
4 /**
5  * Creates an array containing all parsed XML data
6  *
7  * The function runs each and every tag in the $Tags array
8  * through the XMLParser object. The parsed data is then
9  * stored in the $Graph which is returned at the end of the
0  * process.
1  *
2  * @access public
3  * @return $Graph
4  * @uses $Tags
5  * @uses XMLParser
6  */
7 function CreateInputArray()
8 {
9     // Create an array to store the parsed XML data in it
0     // and then return it at the end of the process.
1     $Graph = array();
2
3     // Get each interested tag from $Tags
4     foreach( $this-> Tags as $Tag )
5     {
6         // Create instance of XMLParser and parse the
7         // single tag to it
8         $XMLParser = new XMLParser($Tag);
9         // Store the data parsed by the XMLParser in the $Graph
0         array_push($Graph, $XMLParser-> Tag);
1     }
2
3     // Return the $Graph to user
4     return $Graph;
5 }
6 }
7 ?>

```


File Source for evalmath.class.php

Documentation for this file is available at [evalmath.class.php](http://www.phpdoc.org/projects/phpdoc/evalmath.class.php)

```
<?
/**
 * Evaluation of expressions
 *
 * Safe evaluation of mathematical expressions
 *
 * @package WikiPlot
 * @subpackage PlotClass
 */

/*
=====

EvalMath - PHP Class to safely evaluate math expressions
Copyright (C) 2005 Miles Kaufmann <http://www.twmagic.com/>

=====

NAME
    EvalMath - safely evaluate math expressions

SYNOPSIS
    <?
        include('evalmath.class.php');
        $m = new EvalMath;
        // basic evaluation:
        $result = $m->evaluate('2+2');
        // supports: order of operation; parentheses; negation; built-in functions
        $result = $m->evaluate('-8(5/2)^2*(1-sqrt(4))-8');
        // create your own variables
        $m->evaluate('a = e^(ln(pi))');
        // or functions
        $m->evaluate('f(x,y) = x^2 + y^2 - 2x*y + 1');
        // and then use them
        $result = $m->evaluate('3*f(42,a)');
    ?>

DESCRIPTION
    Use the EvalMath class when you want to evaluate mathematical expressions
    from untrusted sources. You can define your own variables and functions,
    which are stored in the object. Try it, it's fun!

METHODS
    $m->evalute($expr)
        Evaluates the expression and returns the result. If an error occurs,
        prints a warning and returns false. If $expr is a function assignment,
        returns true on success.

    $m->e($expr)
        A synonym for $m->evaluate().

    $m->vars()
        Returns an associative array of all user-defined variables and values.

    $m->funcs()
        Returns an array of all user-defined functions.

PARAMETERS
    $m->suppress_errors
        Set to true to turn off warnings when evaluating expressions

    $m->last_error
        If the last evaluation failed, contains a string describing the error.
        (Useful when suppress_errors is on).

AUTHOR INFORMATION
    Copyright 2005, Miles Kaufmann.
```

```

LICENSE
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote
products derived from this software without specific prior written
permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.

*/

class EvalMath {

    var $suppress_errors = false;
    var $last_error = null;

    var $v = array('e'=> 2.71,'pi'=> 3.14); // variables (and constants)

    var $f = array(); // user-defined functions

    var $vb = array('e', 'pi'); // constants

    var $fb = array( // built-in functions

        'sin','sinh','arcsin','asin','arcsinh','asinh',
        'cos','cosh','arccos','acos','arccosh','acosh',
        'tan','tanh','arctan','atan','arctanh','atanh',
        'sqrt','abs','ln','log');

    function EvalMath() {
        // make the variables a little more accurate
        $this-> v['pi'] = pi();
        $this-> v['e'] = exp(1);
    }

    function e($expr) {
        return $this-> evaluate($expr);
    }

    function evaluate($expr) {
        $this-> last_error = null;
        $expr = trim($expr);
        if (substr($expr, -1, 1) == ';') $expr = substr($expr, 0, strlen($expr)-1); // strip semicolons at the
        //=====
        // is it a variable assignment?
        if (preg_match('/^\s*([a-z]\w*)\s*=\s*(.+)$/i', $expr, $matches)) {
            if (in_array($matches[1], $this-> vb)) { // make sure we're not assigning to a constant
                return $this-> trigger(" cannot assign to constant '$matches[1]'" );
            }
            if (($tmp = $this-> pfx($this-> nfx($matches[2])) === false) return false; // get the result
            // make sure it's good
            $this-> v[$matches[1]] = $tmp; // if so, stick it in the variable array
            return $this-> v[$matches[1]]; // and return the resulting value
        }
        //=====
        // is it a function assignment?
        if (preg_match('/^\s*([a-z]\w*)\s*\(\s*([a-z]\w*(?:\s*\s*([a-z]\w*)*)\s*)\s*\s*=\s*(.+)$/i', $expr,
            $matches)) {
            $fnn = $matches[1]; // get the function name
            if (in_array($matches[1], $this-> fb)) { // make sure it isn't built in
                return $this-> trigger(" cannot redefine built-in function '$matches[1]'" );
            }
        }
    }
}

```

```

1      $args = explode(" ", preg_replace("/\s+/", " ", $matches[2])); // get
e arguments
2      if (($stack = $this-> nfx($matches[3])) == false) return false; // see if it can be converted to
stfix
3      for ($i = 0; $i < count($stack); $i++) { // freeze the state of the non-argument variables
4          $token = $stack[$i];
5          if (preg_match('/^[a-z]\w*$/i', $token) and !in_array($token, $args)) {
6              if (array_key_exists($token, $this-> v)) {
7                  $stack[$i] = $this-> v[$token];
8              } else {
9                  return $this-> trigger(" undefined variable '$token' in function definition" );
0              }
1          }
2      }
3      $this-> f[$fnn] = array('args'=> $args, 'func'=> $stack);
4      return true;
5      //=====
6      } else {
7          return $this-> pfx($this-> nfx($expr)); // straight up evaluation, woo
8      }
9  }
0
1  function vars() {
2      $output = $this-> v;
3      unset($output['pi']);
4      unset($output['e']);
5      return $output;
6  }
7
8  function funcs() {
9      $output = array();
0      foreach ($this-> f as $fnn=> $dat)
1          $output[] = $fnn . '(' . implode(',', $dat['args']) . ')';
2      return $output;
3  }
4
5  //===== HERE BE INTERNAL METHODS =====\\
6
7  // Convert infix to postfix notation
8
9  function nfx($expr) {
0
1      $index = 0;
2      $stack = new EvalMathStack;
3      $output = array(); // postfix form of expression, to be passed to pfx()
4      $expr = trim(strtolower($expr));
5
6      $ops = array('+', '-', '*', '/', '^', '_');
7      $ops_r = array('+=> 0, '-=> 0, '*=> 0, '/=> 0, '^=> 1); // right-associative operator?
8      $ops_p = array('+=> 0, '-=> 0, '*=> 1, '/=> 1, '^=> 2); // operator precedence
9
0      $expecting_op = false; // we use this in syntax-checking the expression
1      // and determining when a - is a negation
2
3      if (preg_match("/[^\w\s+^\\(\\)\\.\\-|]/", $expr, $matches)) { // make sure the characters are all
4          return $this-> trigger(" illegal character '{$matches[0]}'" );
5      }
6
7      while(1) { // 1 Infinite Loop ;}
8          $op = substr($expr, $index, 1); // get the first character at the current index
9          // find out if we're currently at the beginning of a number/variable/function/parenthesis/operand
0          $sex = preg_match('/^([a-z]\w*|(?|\d+(?:\.\d*)?)|\.|\d+|\\(|\\))', substr($expr, $index), $match);
1          //=====
2          if ($op == '-' and !$expecting_op) { // is it a negation instead of a minus?
3              $stack-> push('_'); // put a negation on the stack
4              $index++;
5          } elseif ($op == '_') { // we have to explicitly deny this, because it's legal on the stack
6              return $this-> trigger("illegal character '_'"); // but not in the input expression
7          } //=====
8          elseif ((in_array($op, $ops) or $sex) and $expecting_op) { // are we putting an operator on the
9              if ($sex) { // are we expecting an operator but have a number/variable/function/opening
0                  $op = '*'; $index--; // it's an implicit multiplication
1              }
2              // heart of the algorithm:
3              while($stack-> count > 0 and ($o2 = $stack-> last()) and in_array($o2, $ops) and
4                  $ops_r[$op] < $ops_p[$o2] : $ops_p[$op] <= $ops_p[$o2])) {
5                  $output[] = $stack-> pop(); // pop stuff off the stack into the output
6              }
7          }
8      }
9  }
0
1  }
2
3  }

```

```

4         }
5         // many thanks: http://en.wikipedia.org/wiki/Reverse_Polish_notation#The_algorithm_in_detail
6         $stack-> push($op); // finally put OUR operator onto the stack
7         $index++;
8         $expecting_op = false;
9         //=====
0     } elseif ($op == ')') and $expecting_op { // ready to close a parenthesis?
1         while (($o2 = $stack-> pop()) != '(') { // pop off the stack back to the last (
2             if (is_null($o2)) return $this-> trigger("unexpected ')" );
3             else $output[] = $o2;
4         }
5         if (preg_match("/^([a-z]\w*)\($/" , $stack-> last(2), $matches)) { // did we just
6             $fnn = $matches[1]; // get the function name
7             $arg_count = $stack-> pop(); // see how many arguments there were (cleverly stored on the
8             $output[] = $stack-> pop(); // pop the function and push onto the output
9             if (in_array($fnn, $this-> fb)) { // check the argument count
0                 if($arg_count > 1)
1                     return $this-> trigger(" too many arguments ($arg_count given, 1
2                     expected)" );
3                 } elseif (array_key_exists($fnn, $this-> f)) {
4                     if ($arg_count != count($this-> f[$fnn]['args']))
5                         return $this-> trigger(" wrong number of arguments ($arg_count given, "
6                         unt($this-> f[$fnn]['args']) . " expected)" );
7                     } else { // did we somehow push a non-function on the stack? this should never happen
8                         return $this-> trigger("internal error" );
9                     }
0                 }
1                 $index++;
2                 //=====
3             } elseif ($op == ',' and $expecting_op) { // did we just finish a function argument?
4                 while (($o2 = $stack-> pop()) != '(') {
5                     if (is_null($o2)) return $this-> trigger("unexpected ','" ); // oops, never had a
6                     else $output[] = $o2; // pop the argument expression stuff and push onto the output
7                 }
8                 // make sure there was a function
9                 if (!preg_match("/^([a-z]\w*)\($/" , $stack-> last(2), $matches))
0                     return $this-> trigger("unexpected ','" );
1                 $stack-> push($stack-> pop()+1); // increment the argument count
2                 $stack-> push('('); // put the ( back on, we'll need to pop back to it again
3                 $index++;
4                 $expecting_op = false;
5                 //=====
6             } elseif ($op == '(' and !$expecting_op) {
7                 $stack-> push('('); // that was easy
8                 $index++;
9                 $allow_neg = true;
0                 //=====
1             } elseif ($ex and !$expecting_op) { // do we now have a function/variable/number?
2                 $expecting_op = true;
3                 $val = $match[1];
4                 if (preg_match("/^([a-z]\w*)\($/" , $val, $matches)) { // may be func, or variable w/
5                     // implicit multiplication against parentheses...
6                     if (in_array($matches[1], $this-> fb) or array_key_exists($matches[1], $this-> f)) { //
7                         // it's a func
8                         $stack-> push($val);
9                         $stack-> push(1);
0                         $stack-> push('(');
1                         $expecting_op = false;
2                     } else { // it's a var w/ implicit multiplication
3                         $val = $matches[1];
4                         $output[] = $val;
5                     }
6                 } else { // it's a plain old var or num
7                     $output[] = $val;
8                 }
9                 $index += strlen($val);
0                 //=====
1             } elseif ($op == ')') { // miscellaneous error checking
2                 return $this-> trigger("unexpected ')" );
3             } elseif (in_array($op, $ops) and !$expecting_op) {
4                 return $this-> trigger(" unexpected operator '$op'" );
5             } else { // I don't even want to know what you did to get here
6                 return $this-> trigger("an unexpected error occurred" );
7             }
8         }
9         if ($index == strlen($expr)) {
0             if (in_array($op, $ops)) { // did we end with an operator? bad.
1                 return $this-> trigger(" operator '$op' lacks operand" );
2             }
3         }

```

```

7         } else {
8             break;
9         }
10    }
11    while (substr($expr, $index, 1) == ' ') { // step the index past whitespace (pretty much turns
12        $index++; // into implicit multiplication if no operator is there)
13    }
14    }
15    while (!is_null($op = $stack-> pop())) { // pop everything off the stack and push onto output
16        if ($op == '(') return $this-> trigger("expecting ')" ); // if there are (s on the
17        $output[] = $op;
18    }
19    return $output;
20 }
21
22 // evaluate postfix notation
23
24 function pfx($tokens, $vars = array()) {
25     if ($tokens == false) return false;
26
27     $stack = new EvalMathStack;
28
29     foreach ($tokens as $token) { // nice and easy
30         // if the token is a binary operator, pop two values off the stack, do the operation, and push the
31         // result back on
32         if (in_array($token, array('+', '-', '*', '/', '^'))) {
33             if (is_null($op2 = $stack-> pop())) return $this-> trigger("internal error" );
34             if (is_null($op1 = $stack-> pop())) return $this-> trigger("internal error" );
35             switch ($token) {
36                 case '+':
37                     $stack-> push($op1+$op2); break;
38                 case '-':
39                     $stack-> push($op1-$op2); break;
40                 case '*':
41                     $stack-> push($op1*$op2); break;
42                 case '/':
43                     if ($op2 == 0) return $this-> trigger("division by zero" );
44                     $stack-> push($op1/$op2); break;
45                 case '^':
46                     $stack-> push(pow($op1, $op2)); break;
47             }
48             // if the token is a unary operator, pop one value off the stack, do the operation, and push it back
49         } elseif ($token == "-" ) {
50             $stack-> push(-1*$stack-> pop());
51         } // if the token is a function, pop arguments off the stack, hand them to the function, and push the
52         // result back on
53         elseif (preg_match("/^([a-z]\w*)\$/", $token, $matches)) { // it's a function!
54             $fnn = $matches[1];
55             if (in_array($fnn, $this-> fb)) { // built-in function:
56                 if (is_null($op1 = $stack-> pop())) return $this-> trigger("internal error" );
57                 $fnn = preg_replace("/^arc/", "a", $fnn); // for the 'arc' trig synonyms
58                 if ($fnn == 'ln') $fnn = 'log';
59                 eval('$stack->push(' . $fnn . '($op1));'); // perfectly safe eval()
60             } elseif (array_key_exists($fnn, $this-> f)) { // user function
61                 // get args
62                 $args = array();
63                 for ($i = count($this-> f[$fnn]['args'])-1; $i >= 0; $i--) {
64                     if (is_null($args[$this-> f[$fnn]['args'][$i]] = $stack-> pop())) return $this->
65                     trigger("internal error" );
66                 }
67                 $stack-> push($this-> pfx($this-> f[$fnn]['func'], $args)); // yay... recursion!!!!
68             }
69         } // if the token is a number or variable, push it on the stack
70         else {
71             if (is_numeric($token)) {
72                 $stack-> push($token);
73             } elseif (array_key_exists($token, $this-> v)) {
74                 $stack-> push($this-> v[$token]);
75             } elseif (array_key_exists($token, $vars)) {
76                 $stack-> push($vars[$token]);
77             } else {
78                 return $this-> trigger(" undefined variable '$token'" );
79             }
80         }
81     }
82 }

```

```

0 // when we're out of tokens, the stack should have a single element, the final result
1 if ($stack-> count != 1) return $this-> trigger("internal error" );
2 return $stack-> pop();
3 }
4
5 // trigger an error, but nicely, if need be
6 function trigger($msg) {
7     $this-> last_error = $msg;
8     if (!$this-> suppress_errors) trigger_error($msg, E_USER_WARNING);
9     return false;
0 }
1 }
2
3 // for internal use
4 class EvalMathStack {
5
6     var $stack = array();
7     var $count = 0;
8
9     function push($val) {
10         $this-> stack[$this-> count] = $val;
11         $this-> count++;
12     }
13
14     function pop() {
15         if ($this-> count > 0) {
16             $this-> count--;
17             return $this-> stack[$this-> count];
18         }
19         return null;
20     }
21
22     function last($n=1) {
23         return $this-> stack[$this-> count-$n];
24     }
25 }

```

File Source for graph.plot.class.php

Documentation for this file is available at graph.plot.class.php

```
<?php
/*
Copyright (C) 2006 by the WikiPlot project authors (See http://code.google.com/p/WikiPlot).

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

/**
 * File containing Graph representation
 *
 * This file contains a class used as representation of a Graph in plot's. It cannot be used independently, it is
a requirement of plot.class.php
 */
 * @package WikiPlot
 * @subpackage PlotClass
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */

/**
 * Representation of a graph
 *
 * Class used to represente graphs on a plot.
 */
 * @package WikiPlot
 * @subpackage PlotClass
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */
class Graph
{
    /**
     * Label of graph
     *
     * This is the label or legend of the graph and will be shown in the corner of the plot, i the graphs color.
     */
    * @access public
    * @var string
    */
    var $Label;

    /**
     * Font of the label
     *
     * This is the font of the label, defaults to 2, 1-5 are built-in and works as different fontsizes.
     */
    * @access public
    * @var integer
    */
    var $LabelFont = 2;

    /**
     * Enable label
     *
     * Enable label, defaults to true, draws label if true.
     */
}
```

```

*
**@access public
**@var boolean
**/
var $EnableLabel = true;

/**
 *Expression
 *
 *The mathematical expression representing the graph.
 *
 **@see EvalMath::evaluate()
 **@access public
 **@var string
 **/
var $Exp;

/**
 * Color of the graph
 *
 * Color of the graph and label, array of the RGB representation of the color.
 * Example: array($Red,$Green,$Blue);
 *
 **@access public
 **@var array
 **/
var $Color = array(0,0,0);

/**
 *Get hash
 *
 *Gets a hash of the graphs parameters. Actually is not a hashsum but just all parameter parsed as one
string, this is done to reduce collision risk in Plot::GetHash().
 *
 **@access private
 **@return string Hash of all parameters.
 **/
function GetHash()
{
    return $this-> Label . "_" . $this-> LabelFont . "_" . $this-> Exp . "_"
his-> Color[0] . "_" . $this-> Color[1] . "_" . $this-> Color[2] . "_" . $this-
EnableLabel;
}
}
?>

```


File Source for plot.class.php

Documentation for this file is available at plot.class.php

```
<?php
/*
Copyright (C) 2006 by the WikiPlot project authors (See http://code.google.com/p/WikiPlot).

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General
Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

/**
 * File use to draw plots
 *
 * This file contains a class used to draw plot's. It's dependent on graph.plot.class.php and evalmath.class.php.
 *
 * @package WikiPlot
 * @subpackage PlotClass
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */

/**
 *Includes EvalMath
 *
 *EvalMath is used to evaluate mathematical expressions in a safe environment.
 */
require_once('evalmath.class.php');

/**
 *Includes Graph representation class
 *
 *Graph is used as a representation of a graph.
 */
require_once('graph.plot.class.php');

/**
 * Class used to draw plots
 *
 * Class containing functions to draw plots to an image.
 *
 * @package WikiPlot
 * @subpackage PlotClass
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */
class Plot
{
    /**
     * Graphs to plot
     *
     * Array containing list of Graphs to plot.
     *
     * @var array
     * @access public
     * @see Graph
     */
    var $Graphs = array();
}
```

```

/**
 * Caption of the plot
 *
 * Caption of the plot, will be shown as text centered on the final plot.
 * Leave this variable as null if no Caption is wanted.
 *
 * @var string
 * @access public
 * @see DrawCaption
 */
var $Caption = null;

/**
 * Caption font
 *
 * Font of the Caption, the fonts 1-5 is built in, and behaves as different sizes.
 *
 * @var integer
 * @access public
 * @see DrawCaption
 */
var $CaptionFont = 5;

/**
 * Width of output image
 *
 * The width of the output image, in pixels.
 *
 * @var integer
 * @access public
 * @see DrawPlots
 */
var $Width = 100;

/**
 * Height of output image
 *
 * The width of the output image, in pixels.
 *
 * @var integer
 * @access public
 * @see DrawPlots
 */
var $Height = 100;

/**
 * Minimum X
 *
 * Minimum X in coordinate space.
 * Together with MaxX this variable defines width of the plot in coordinate space.
 * This width may differ from width of the image, the coordinate will be scaled correctly.
 *
 * @var integer
 * @access public
 * @see DrawPlots
 * @see MaxX
 */
var $MinX = -10;

/**
 * Maximum X
 *
 * Maximum X in coordinate space.
 * Together with MinX this variable defines width of the plot in coordinate space.
 * This width may differ from width of the image, the coordinate will be scaled correctly.
 *
 * @var integer
 * @access public
 * @see DrawPlots
 * @see MinX
 */
var $MaxX = 100;

/**
 * Minimum Y
 *
 * Minimum Y in coordinate space.
 * Together with MaxY this variable defines height of the plot in coordinate space.
 * This height may differ from height of the image, the coordinate will be scaled correctly.
 *
 * @var integer
 * @access public

```

```

3  * @see DrawPlots
4  * @see MaxY
5  */
6  var $MinY = -10;
7  /**
8   * Maximum Y
9   *
10  * Maximum Y in coordinate space.
11  * Together with MinY this variable defines height of the plot in coordinate space.
12  * This height may differ from height of the image, the coordinate will be scaled correctly.
13  *
14  * @var integer
15  * @access public
16  * @see DrawPlots
17  * @see MinY
18  */
19  var $MaxY = 100;
20
21  /**
22  * Enable Axis
23  *
24  * Defaults to true and draws 2 axis.
25  *
26  * @var boolean
27  * @access public
28  * @see DrawAxis
29  */
30  var $EnableAxis = true;
31  /**
32  * Enable Grid
33  *
34  * Defaults to true and draws a grid.
35  *
36  * @var boolean
37  * @access public
38  * @see DrawGrid
39  */
40  var $EnableGrid = true;
41  /**
42  * Grid color
43  *
44  * Defaults to gray, and determines the color of the grid. This is an array of three integers, one for red,
45  * green and blue. Where integer has values between 0 and 255.
46  * <code>
47  * var $Red = 240;
48  * var $Green = 240;
49  * var $Blue = 240;
50  * $this->GridColor = array($Red,$Green,$Blue);
51  * </code>
52  *
53  * @var array
54  * @access public
55  * @see DrawGrid
56  */
57  var $GridColor = array(240,240,240);
58  /**
59  * Grid font
60  *
61  * Font of the grids labels, the fonts 1-5 is built in, and behaves as different sizes.
62  *
63  * @var integer
64  * @access public
65  * @see DrawGrid
66  */
67  var $GridFont = 1;
68  /**
69  * X grid space
70  *
71  * Distance between grids on the x axis in coordinate space. Defaults to null, leave it null, if you want
72  * to generate grid space.
73  *
74  * @var integer
75  * @access public
76  * @see GetXGridSpace
77  */
78  var $XGridSpace = null;
79  /**
80  * Y grid space
81  *
82  * Distance between grids on the y axis in coordinate space. Defaults to null, leave it null, if you want

```

```

togenerated gridspace.
1
2 * @var integer
3 * @access public
4 * @see GetYGridSpace
5 */
6 var $YGridSpace = null;
7 /**
8  * Background color
9  *
10  * Color of the background when using auto ImageResource created by GeneratePlot().
11  *
12  * @var array
13  * @access public
14  */
15 var $BackgroundColor = array(255,255,255);
16
17 /**
18  *Generate hash
19  *
20  *Generates a unigue hashsum (md5) for the plot, generated from all parameters.
21  *
22  *@uses $Caption
23  *@uses $CaptionFont
24  *@uses $Width
25  *@uses $Height
26  *@uses $MinX
27  *@uses $MaxX
28  *@uses $MinY
29  *@uses $MaxY
30  *@uses $EnableGrid
31  *@uses $GridColor
32  *@uses $GridFont
33  *@uses $EnableAxis
34  *@uses $XGridSpace
35  *@uses $YGridSpace
36  *@uses $Graphs
37  *@uses Graph::GetHash()
38  *@return string Hash representation of the object.
39  */
40 function GetHash()
41 {
42     $Hash = "C:" . $this->Caption;
43     $Hash .= "F:" . $this->CaptionFont;
44     $Hash .= "W:" . $this->Width;
45     $Hash .= "H:" . $this->Height;
46     $Hash .= "X:" . $this->MinX . "_" . $this->MaxX;
47     $Hash .= "Y:" . $this->MinY . "_" . $this->MaxY;
48     $Hash .= "A:" . $this->EnableAxis;
49     $Hash .= "G:" . $this->EnableGrid . "_" . $this->GridColor . "_" . $this->GridFont;
his-> $Hash .= "S:" . $this->XGridSpace . "_" . $this->YGridSpace;
50     $Hash .= "V:" . $LastChangedRevision: 63 $ ;
51     foreach($this->Graphs as $key => $S)
52     {
53         $Hash .= "G:" . $key . "_" . $S->GetHash();
54     }
55     return md5($Hash);
56 }
57
58 /**
59  *Get ImageResource of the plot
60  *
61  *Generates ImageResource representation of the plot.
62  *
63  *@access public
64  *@uses EnableGrid
65  *@uses DrawGrid()
66  *@uses $Width
67  *@uses $Height
68  *@uses $EnableAxis
69  *@uses DrawAxis()
70  *@uses DrawCaption()
71  *@uses DrawPlots()
72  *@uses $BackgroundColor
73  *@param ImageResource $ImageResource Defaults to null, will generate empty ImageResource.
74  *@param Boolean $ChangeSize May we change the size of the plot to fit given ImageResource?
75  *@return ImageResource ImageResource representation of the plot.
76  */
77 function GeneratePlot($ImageResource = null, $ChangeSize = false)

```

```

9 {
10     //If ImageResource is null
11     if(is_null($ImageResource))
12     {
13         //Get ImageResource
14         $ImageResource = imagecreatetruecolor($this-> Height,$this-> Width);
15
16         //AntiAlias ON
17         imageantialias($ImageResource,true);
18
19         //Fill the image with white
20         imagefill($ImageResource,0,0,imagecolorexact($ImageResource,$this-> BackgroundColor[0],$this-
21 BackgroundColor[1],$this-> BackgroundColor[2]));
22
23     }//If ImageResource doesn't fit image and we may not change size
24     elseif($ChangeSize==false&&( imagesx($ImageResource) !=$this-
25 Width | imagesy($ImageResource) !=$this-> Height))
26     {
27         //Get ImageResource
28         $ImageResource = imagecreatetruecolor($this-> Height,$this-> Width);
29
30         //AntiAlias ON
31         imageantialias($ImageResource,true);
32
33         //Fill the image with white
34         imagefill($ImageResource,0,0,imagecolorexact($ImageResource,$this-> BackgroundColor[0],$this-
35 BackgroundColor[1],$this-> BackgroundColor[2]));
36     }//If we may change the size of the plot
37     elseif($ChangeSize)
38     {
39         //Changing size of the plot.
40         $this-> Width = imagesx($ImageResource);
41         $this-> Height = imagesy($ImageResource);
42     }
43
44     //If grid is enabled
45     if($this-> EnableGrid)
46     {
47         $this-> DrawGrid($ImageResource);
48     }
49
50     //If axis is enabled
51     if($this-> EnableAxis)
52     {
53         $this-> DrawAxis($ImageResource);
54     }
55
56     //Draw caption
57     $this-> DrawCaption($ImageResource);
58
59     //Draw plots
60     $this-> DrawPlots($ImageResource);
61
62     //Return ImageResource
63     return $ImageResource;
64 }
65
66 /**
67 *Get ImageResource of the plot
68 *
69 *Generates ImageResource representation of the plot.
70 *
71 *@access private
72 *@uses $Width
73 *@uses EvalMath
74 *@uses EvalMath::evaluate()
75 *@uses GetCoordinatX()
76 *@uses GetImageX()
77 *@uses GetImageY()
78 *@uses $Graphs
79 *@uses Graph::$Color
80 *@uses Graph::$LabelFont
81 *@uses Graph::$EnableLabel
82 *@uses Graph::$Label
83 *@param ImageResource &$ImageResource ImageResource representation of the plot.
84 */
85 function DrawPlots(& $ImageResource)
86 {
87     //Get a black Color
88     $Black = imagecolorexact($ImageResource,0,0,0);

```

```

6 //Y position for Labels relative to Image
7 $LabelY = 5;
8
9 //Plot all graphs
10 foreach($this-> Graphs as $key => $S)
11 {
12     //Get Color
13     $Color = imagecolorexact($ImageResource,$S-> Color[0],$S-> Color[1],$S-> Color[3]);
14
15     //Set Expression
16     $m = new EvalMath;
17     $m-> evaluate("f(x) = " . $S-> Exp);
18
19     //Set OldCoordinat*, don't start with a line from 0,0
20     $OldCoordinatX = $this-> GetCoordinatX(0);
21     $OldCoordinatY = $m-> evaluate("f(" . $OldCoordinatX . ") " );
22
23     //Plot the graph
24     for($ImageX=0;$ImageX< $this-> Width;$ImageX++)
25     {
26         //Get some NewCoordinat*
27         $NewCoordinatX = $this-> GetCoordinatX($ImageX);
28         $NewCoordinatY = $m-> evaluate("f(" . $NewCoordinatX . ") " );
29
30         //Draw a line from OldCoordinat*
31         imageline(
32             $ImageResource,
33             $this-> GetImageX($OldCoordinatX),
34             $this-> GetImageY($OldCoordinatY),
35             $this-> GetImageX($NewCoordinatX),
36             $this-> GetImageY($NewCoordinatY),
37             $Color);
38
39         //Get some OldCoordinat*
40         $OldCoordinatX = $NewCoordinatX;
41         $OldCoordinatY = $NewCoordinatY;
42     }
43
44     //Draw label if it is enabled
45     if($S-> EnableLabel)
46     {
47         //Draw label
48         imagestring($ImageResource,$S-> LabelFont,5,$LabelY,"- " . $S-> Label,$Color);
49
50         //Add Label height to next Label X position
51         $LabelY += imagefontheight($S-> LabelFont);
52     }
53 }
54 }
55
56 /**
57 *Draw caption to ImageResource
58 *
59 *Draws the caption to an ImageResource representation of the plot.
60 *
61 *@access private
62 *@uses $Width
63 *@uses $Caption
64 *@uses $CaptionFont
65 *@param ImageResource &$ImageResource ImageResource representation of the plot.
66 */
67 function DrawCaption(& $ImageResource)
68 {
69     //Get a black color for caption
70     $Black = imagecolorexact($ImageResource,0,0,0);
71
72     //width of the caption
73     $CaptionWidth = strlen($this-> Caption)*imagefontwidth($this-> CaptionFont);
74
75     //X position of the caption, making it centered
76     $X = ($this-> Width-$CaptionWidth)/2;
77
78     //Draw the caption
79     imagestring($ImageResource,$this-> CaptionFont,$X,0,$this-> Caption,$Black);
80 }
81
82 /**
83 *Generates short numbers

```

```

6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

6  *@uses DrawXGrid()
7  *@uses DrawYGrid()
8  *@param ImageResource &$ImageResource ImageResource representation of the plot.
9  */
10 function DrawGrid(& $ImageResource)
11 {
12     $this-> DrawXGrid($ImageResource);
13     $this-> DrawYGrid($ImageResource);
14 }
15
16 /**
17  *Draws x-grid
18  *
19  *Drawing X grid on the plot.
20  *
21  *@access private
22  *@uses GetXGridSpace()
23  *@uses $GridColor
24  *@uses $MinX
25  *@uses $MaxX
26  *@uses $MinY
27  *@uses $MaxY
28  *@uses GetImageX()
29  *@uses GetImageY()
30  *@uses $GridFont
31  *@uses $Height
32  *@uses ShortNumber()
33  *@param ImageResource &$ImageResource ImageResource representation of the plot.
34  */
35 function DrawXGrid(& $ImageResource)
36 {
37     //Get grid width
38     $XGridSpace = $this-> GetXGridSpace();
39
40     //Get color to draw with
41     $Color = imagecolorexact($ImageResource,$this-> GridColor[0],$this-> GridColor[1],$this->
GridColor[2]);
42     //Get text color
43     $Black = imagecolorexact($ImageResource,0,0,0);
44
45     //Calculate start and end coordinats of the grid
46     $XGridStart = ($this-> MinX-fmod($this-> MinX,$XGridSpace));
47     $XGridEnd = $this-> MaxX-fmod(($this-> MaxX-$this-> MinX),$XGridSpace);
48
49     //Draw the grid
50     for($XCoordinate=$XGridStart;$XCoordinate< $XGridEnd;$XCoordinate+=$XGridSpace)
51     {
52         imageline(
53             $ImageResource,
54             $this-> GetImageX($XCoordinate),
55             $this-> GetImageY($this-> MinY),
56             $this-> GetImageX($XCoordinate),
57             $this-> GetImageY($this-> MaxY),
58             $Color);
59
60         //If Y axes is not on the image (working in ImageSpace not CoordinatSpace)
61         $Y = $this-> GetImageY(0);
62         if($Y > ( $this-> Height-imagefontheight($this-> GridFont)))
63         {
64             $Y = $this-> Height-(imagefontheight($this-> GridFont)+2);
65         }else{
66             if($Y< 0)
67             {
68                 $Y = 0;
69             }
70         }
71         imagestring(
72             $ImageResource,
73             $this-> GridFont,
74             $this-> GetImageX($XCoordinate)+2,
75             $Y+2,
76             $this-> ShortNumber($XCoordinate),
77             $Black);
78     }
79 }
80
81 /**
82  *Draws y-grid
83  *
84  *Drawing y grid on the plot.

```



```

5 *
6 *@access private
7 *@uses GetYGridSpace()
8 *@uses $GridColor
9 *@uses $MinX
10 *@uses $MaxX
11 *@uses $MinY
12 *@uses $MaxY
13 *@uses GetImageX()
14 *@uses GetImageY()
15 *@uses $GridFont
16 *@uses $Width
17 *@uses ShortNumber()
18 *@param ImageResource &$ImageResource ImageResource representation of the plot.
19 */
20 function DrawYGrid(& $ImageResource)
21 {
22     //Get grid width
23     $YGridSpace = $this-> GetYGridSpace();
24
25     //Get color to draw with
26     $Color = imagecolorexact($ImageResource,$this-> GridColor[0],$this-> GridColor[1],$this->
GridColor[2]);
27     //Get text color
28     $Black = imagecolorexact($ImageResource,0,0,0);
29
30     //Calculate start and end coordinats of the grid
31     $YGridStart = ($this-> MinY-fmod($this-> MinY,$YGridSpace));
32     $YGridEnd = $this-> MaxY-fmod(($this-> MaxY-$this-> MinY),$YGridSpace);
33
34     //Draw the grid
35     for($YCoordinate=$YGridStart;$YCoordinate< $YGridEnd;$YCoordinate+=$YGridSpace)
36     {
37         imageline(
38             $ImageResource,
39             $this-> GetImageX($this-> MinX),
40             $this-> GetImageY($YCoordinate),
41             $this-> GetImageX($this-> MaxX),
42             $this-> GetImageY($YCoordinate),
43             $Color);
44
45         //If X axes is not on the image (working in ImageSpace not CoordinatSpace)
46         $X = $this-> GetImageX(0);
47         if($X > ( $this-> Width-(imagefontwidth($this-> GridFont)*7)))
48         {
49             $X = $this-> Width-(imagefontwidth($this-> GridFont)*7+2);
50         }else{
51             if($X< 0)
52             {
53                 $X = 0;
54             }
55         }
56         imagestring(
57             $ImageResource,
58             $this-> GridFont,
59             $X+2,
60             $this-> GetImageY($YCoordinate)+2,
61             $this-> ShortNumber($YCoordinate),
62             $Black);
63     }
64 }
65
66 /**
67 * Draw axis
68 *
69 * Draw both x and y axis to the plot.
70 *
71 *@access private
72 *@uses $MinX
73 *@uses $MaxX
74 *@uses $MinY
75 *@uses $MaxY
76 *@uses GetImageX()
77 *@uses GetImageY()
78 *@param ImageResource &$ImageResource ImageResource representation of the plot.
79 */
80 function DrawAxis(& $ImageResource)
81 {
82     $Black = imagecolorexact($ImageResource,0,0,0);
83     //Draw X-axis

```

```

4     imageline(
5         $ImageResource,
6         $this-> GetImageX(0),
7         $this-> GetImageY($this-> MinY),
8         $this-> GetImageX(0),
9         $this-> GetImageY($this-> MaxY),
10        $Black);
11    //Draw Y-axis
12    imageline($ImageResource,
13        $this-> GetImageX($this-> MinX),
14        $this-> GetImageY(0),
15        $this-> GetImageX($this-> MaxX),
16        $this-> GetImageY(0),
17        $Black);
18    }
19
20    /**
21     *Display plot as image
22     *
23     *Displays plot as image on the page. This makes current http-request return an image. You can set the
24     *displayType to png, gif or jpeg. Defaults to png, gif not recommended. Note: this changes the current http-request
25     *mimetype to the respective image mimetype.
26     *
27     *@access public
28     *@uses GeneratePlot()
29     *@param string $DisplayType Type of image to view (png/jpeg/gif).
30     *@param ImageResource $ImageResource Defaults to null, will generate empty ImageResource.
31     *@param Boolean $ChangeSize May we change the size of the plot to fit given ImageResource?
32     */
33    function DisplayPlot($DisplayType = "png" , $ImageResource = null, $ChangeSize = false)
34    {
35        if($DisplayType == "png" )
36        {
37            header("Content-type: image/png" );
38            imagepng($this-> GeneratePlot($ImageResource, $ChangeSize));
39        }
40        elseif($DisplayType == "gif" )
41        {
42            header("Content-type: image/gif" );
43            imagegif($this-> GeneratePlot($ImageResource, $ChangeSize));
44        }
45        else
46        {
47            header("Content-type: image/jpeg" );
48            imagejpeg($this-> GeneratePlot($ImageResource, $ChangeSize));
49        }
50    }
51
52    /**
53     *Save plot to image
54     *
55     *Saves the plot to an image. You can set the SaveAs to a file type: png, gif or jpeg, defaults to png.
56     *
57     *@access public
58     *@uses GeneratePlot()
59     *@param string $Path Path of file to save.
60     *@param string $SaveAs Filetype definition (png/jpeg/gif).
61     *@param ImageResource $ImageResource Defaults to null, will generate empty ImageResource.
62     *@param Boolean $ChangeSize May we change the size of the plot to fit given ImageResource?
63     */
64    function SaveAs($Path,$SaveAs = "png" , $ImageResource = null, $ChangeSize = false)
65    {
66        if($SaveAs == "png" )
67        {
68            imagepng($this-> GeneratePlot($ImageResource, $ChangeSize),$Path);
69        }
70        elseif($SaveAs == "gif" )
71        {
72            imagegif($this-> GeneratePlot($ImageResource, $ChangeSize),$Path);
73        }
74        else
75        {
76            imagejpeg($this-> GeneratePlot($ImageResource, $ChangeSize),$Path);
77        }
78    }
79
80    /**
81     * Convert to coordinate space
82     *
83     * Converts an x image position to x coordinate position. Coordinate space may differ from Image space, if

```

```

1 dth!= (MaxX-MinX).
2 *
3 *@access private
4 *@uses $MaxX
5 *@uses $MinX
6 *@uses $Width
7 *@param integer $x X image coordinat to be converted.
8 *@return integer Coordiante space representation given parameter.
9 */
10 function GetCoordinatX($x)
11 {
12     return (($this-> MaxX-$this-> MinX)/$this-> Width)*$x+$this-> MinX;
13 }
14
15 /**
16 * Convert to coordinate space
17 *
18 * Converts an y image position to y coordinate position. Coordinate space may differ from Image space, if
19 ight!= (MaxY-MinY).
20 *
21 *@access private
22 *@uses $MaxY
23 *@uses $MinY
24 *@uses $Height
25 *@param integer $y Y image coordinat to be converted.
26 *@return integer Coordiante space representation given parameter.
27 */
28 function GetCoordinatY($y)
29 {
30     return (($this-> MaxY-$this-> MinY)/$this-> Height)*($this-> Height-$y)+$this-> MinY;
31 }
32
33 /**
34 * Convert to image space
35 *
36 * Converts an x in coordinate space to x image position. Coordinate space may differ from Image space, if
37 dth!= (MaxX-MinX).
38 *
39 *@access private
40 *@uses $MaxX
41 *@uses $MinX
42 *@uses $Width
43 *@param integer $x X coordinat to be converted.
44 *@return integer Image position representation given parameter.
45 */
46 function GetImageX($x)
47 {
48     return ($x-$this-> MinX)*($this-> Width/($this-> MaxX-$this-> MinX));
49 }
50
51 /**
52 * Convert to image space
53 *
54 * Converts an y in coordinate space to y image position. Coordinate space may differ from Image space, if
55 ight!= (MaxY-MinY).
56 *
57 *@access private
58 *@uses $MaxY
59 *@uses $MinY
60 *@uses $Height
61 *@param integer $y Y coordinat to be converted.
62 *@return integer Image position representation given parameter.
63 */
64 function GetImageY($y)
65 {
66     return $this-> Height-($y-$this-> MinY)*($this-> Height/($this-> MaxY-$this-> MinY));
67 }
68 }
69 ?>

```

File Source for test.php

Documentation for this file is available at [test.php](#)

```
<?php
/**
 * Example/Test
 *
 * This is an example/test of how to use plot.class.php
 *
 * @package WikiPlot
 * @subpackage PlotClass
 * @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
 * @author WikiPlot development team.
 * @copyright Copyright 2006, WikiPlot development team.
 */
header("Content-type: image/png"          );

/**
 * Includes plot.class.php for testing
 *
 * The file tests PlotClass, and must therefor depend on it.
 */
include("plot.class.php"                  );

$Plot = new Plot;

$G = new Graph;
$G-> Exp = "0.002x^3+2x+5"                ;
$G-> Color = array(0,0,255);
$G-> Label = "test"                       ;

$G1 = new Graph;
$G1-> Exp = "sin(x*0.3)*50+0.00005x^3+0.001x^2"      ;
$G1-> Color = array(0,255,0);
$G1-> Label = "test1"                            ;

$G2 = new Graph;
// $G2->Exp = "sin(x*0.3)*50+0.05x^2+100";
$G2-> Exp = "tan(x/4)*5"                      ;
$G2-> Color = array(255,0,0);
$G2-> Label = "Tan(x)"                        ;

$Plot-> Graphs = array($G,$G1,$G2);

$Plot-> Caption = "Test Graph"                 ;

$Plot-> Width = 500;
$Plot-> Height = 500;

$Plot-> MinX = -250;
$Plot-> MaxX = 250;
$Plot-> MinY = -250;
$Plot-> MaxY = 250;
$Plot-> DisplayPlot();

?>
```

Index

Constructor XMLParser::XMLParser()	23
<i>Constructor of XMLParser class</i>	
Cache::FileURL()	19
<i>Get file URL</i>	
Constructor EvalMath::EvalMath()	31
Cache.class.php	55
<i>Source code</i>	
CleanupCache.php	58
<i>Source code</i>	
Cache::FileExist()	18
<i>Does file exist in cache</i>	
Cache::CleanupUnused()	18
<i>Cleanup unused files from cache</i>	
Cache	17
<i>Cache controlling class</i>	
CleanupCache.php	9
<i>File used to clear the cache</i>	
Cache::CachePath()	17
<i>Get cache Path</i>	
Cache::CleanupCache()	18
<i>Cleanup the cache</i>	
Cache::CleanupMaxAge()	18
<i>Cleanup cache from old files</i>	
Cache.class.php	8
<i>File used to control cache</i>	

EvalMathStack	32
EvalMath::vars()	32
EvalMath::trigger()	32
EvalMath::pfx()	32
EvalMathStack::\$count	32
EvalMathStack::\$stack	32
Evalmath.class.php	69
<i>Source code</i>	
EvalMathStack::push()	32
EvalMathStack::pop()	32
EvalMathStack::last()	32
EvalMath::nfx()	31
EvalMath::funcs()	31
EvalMath::\$last_error	31
EvalMath::\$fb	31

valMath::\$f	30
valMath	30
<i>Evaluation of expressions</i>	
valMath::\$suppress_errors	31
valMath::\$v	31
valMath::evaluate()	31
valMath::e()	31
valMath::\$vb	31
valmath.class.php	27

Graph::\$LabelFont	34
<i>Font of the label</i>	
Graph::GetHash()	35
<i>Get hash</i>	
graph.plot.class.php	75
<i>Source code</i>	
Graph::\$Label	34
<i>Label of graph</i>	
Graph::\$Exp	34
<i>Expression</i>	
Graph	33
<i>Representation of a graph</i>	
Graph::\$Color	33
<i>Color of the graph</i>	
Graph::\$EnableLabel	33
<i>Enable label</i>	
graph.plot.class.php	28
<i>File containing Graph representation</i>	

Plot::DrawYGrid()	45
<i>Draws y-grid</i>	
Plot::GeneratePlot()	46
<i>Get ImageResource of the plot</i>	
Plot::GetCoordinatX()	46
<i>Convert to coordinate space</i>	
Plot::DrawXGrid()	44
<i>Draws x-grid</i>	
Plot::DrawPlots()	44
<i>Get ImageResource of the plot</i>	
Plot::DrawAxis()	42
<i>Draw axis</i>	
Plot::DrawCaption()	43
<i>Draw caption to ImageResource</i>	
Plot::DrawGrid()	43
<i>Draw grids</i>	
Plot::GetCoordinatY()	47
<i>Convert to coordinate space</i>	

Plot::GetHash()	47
Generate hash	
Plot::SaveAs()	49
Save plot to image	
Plot::ShortNumber()	50
Generates short numbers	
Plot.class.php	77
Source code	
Plot::GetYGridSpace()	49
Get Y grid space	
Plot::GetXGridSpace()	49
Get X grid space	
Plot::GetImageX()	48
Convert to image space	
Plot::GetImageY()	48
Convert to image space	
Plot::DisplayPlot()	42
Display plot as image	
Plot::\$YGridSpace	42
Y grid space	
Plot::\$EnableAxis	37
Enable Axis	
Plot::\$EnableGrid	37
Enable Grid	
Plot::\$Graphs	37
Graphs to plot	
Plot::\$CaptionFont	36
Caption font	
Plot::\$Caption	36
Caption of the plot	
Plot	35
Class used to draw plots	
Plot::\$BackgroundColor	36
Background color	
Plot::\$GridColor	38
Grid color	
Plot::\$GridFont	38
Grid font	
Plot::\$MinY	40
Minimum Y	
Plot::\$Width	41
Width of output image	
Plot::\$XGridSpace	41
X grid space	
Plot::\$MinX	40
Minimum X	
Plot::\$MaxY	39
Maximum Y	
Plot::\$Height	38
Height of output image	
Plot::\$MaxX	39
Maximum X	
Plot.class.php	29

enderWikiPlot()	10
---------------------------------	----

st.php	88
Source code	
st.php	30
Example/Test	

WikiPlotCachePath	14
Path to the cache	
WikiPlotCacheAge	14
Max Cache Age	
WikiPlotSettings.php	14
File used to store settings	
WikiPlotCacheURL	14
URL to cache	
WikiPlotMaxUnusedAge	15
Max Unused Age	
WikiPlotSettings.php	63
Source code	
WikiPlot.php	59
Source code	
WikiPlotDeserializeString()	12
Deserialize String	
WikiPlotDeserializeMixed()	11
Deserialize Coordiante	
WikiPlot.php	10
WikiPlot Syntax Reference	3
WikiPlotExtension()	10
WikiPlotDeserializeBoolean()	10
Deserialize boolean	
WikiPlotDeserializeInteger()	11
Deserialize Integer	
WikiPlotDeserializeColor()	10
Deserialize Color	
WikiPlot Userguide	1

MLParser::ExplodeInputData()	24
Findes the interested tag in XML Data	
MLParser::CreateTagArray()	24

<i>Puts parsed data into an array</i>	24
XMLParser::CreateInputArray()	24
<i>Creates an array containing all parsed XML data</i>	
XMLParser::GetCharData()	25
<i>Gets data of the tag</i>	
XMLParser::OpenTag()	25
<i>Handles attribute(s) of a tag</i>	
xml.class.php	64
<i>Source code</i>	
XMLParser::Parse()	26
<i>Parses the given XML data</i>	
XMLParser::CloseTag()	23
<i>Handles end/closing tag</i>	
XMLParser::\$Tags	22
<i>All interested tags</i>	
XMLParser::\$Input	20
<i>XML data given by user</i>	
XMLParser::\$Attributes	20
<i>Attributes of interested tag</i>	
XMLParser	19
<i>XMLParser class</i>	
XMLParser::\$Parser	21
<i>Created XML Parser</i>	
XMLParser::\$Separator	21
<i>The interested tag</i>	
XMLParser::\$TagData	22
<i>Data of the tag</i>	
XMLParser::\$Tag	22
<i>An interested tag in given XML data</i>	
xml.class.php	16
<i>The file contains XMLParser class</i>	