WikiPlot Documentation

Contents

WikiPlot Userguide

Introduction to WikiPlot

Authors:*Jonas F. Jensen.*

[*WikiPlot@Jopsen.dk*](mailto:WikiPlot@Jopsen.dk)

WikiPlot Userguide

This is s 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 you 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 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 graph, which is expressed with a mathematical expression (for instance: x^2+4x+5). If you thing this is wired, 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 supprised if you would consider that a little too basic. So just to match some basic functionallity, we are going to add an other graph with the expression y(x)=3*x-3, and some optional parameters to modifly 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 heigth 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 contain 2 lables 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 splittede 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 minimun x and maximun x, in coordinate space. It you set xspan="-50;75" the lowest x values on you image will be -50 and the heighest 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 completly similar except for the fact that they change the y or x coordiante 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 you 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 confues 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 note familiar with xml or the most common WikiPlot syntax, it is recommened that you read the WikiPlot UserGuide first. Below you will find a documentation of the parameters and content for the wikiplot and graph tag, respektively. 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 i 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 the height of the final image in pixels.
- **width**

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

  Two semicolon seperated integers, defining the span of the x-axis. If xspan="-5;10" the minium value on the x-axis will be -5 and the maximun value on the x-axis will be 10. This parameter is very importent, because it defines coordinat space to be viewed.

- **yspan**

  Two semicolon seperated integers, defining the span of the y-axis. If yspan="-5;10" the minium value on the y-axis will be -5 and the maximun value on the y-axis will be 10. This parameter is very importent, because it defines coordinat 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 seperated integers, defining the space between the line 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 x-axis will be 20.
- **gridfont**

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

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

graph parameters and content

The graph tags represents 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 havn't documentated 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 seperated 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

llowing 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

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

Function Parameters:

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

unction wfWikiPlotExtension() *[line 30]*

unction 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

unction 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

nction 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. Usualy this function does nothing at all, just checks to see if the value can be parsed as an integer.

- **Access** private

nction 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

nction 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. Usualy this function does nothing.

- **Access** private

quire_once **"xml.class.php"** *[line 19]*

Include xml.class.php
Requires XMLParser to parse xml to plot.

quire_once **"cache.class.php"** *[line 26]*

Include cache.class.php
Requires Cache to control the cache.

quire_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 functionallity.

- **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

nction 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

nction 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

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

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.

- **Usedby** Cache::CleanupCache()
- **Access** public

olean 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

ring 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 doen'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

MLParser::$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

MLParser::$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

MLParser::$Parser

mixed = [line 63]

Created XML Parser

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

- **Usedby** XMLParser::CloseTag()
- **Usedby** XMLParser::GetCharData()
- **Usedby** XMLParser::OpenTag()
- **Usedby** XMLParser::XMLParser()
- **Access** private

MLParser::$Separator

string = [line 147]

The interested tag

The variable is our iterested tag. It means the tag that we are iterested 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!

- **Usedby** XMLParser::ExplodeInputData()
- **Access** public

MLParser::$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 it selv <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
)
```

s 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 cond 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 ore then two elements, while one element is possible.

- **Usedby** XMLParser::CreateTagArray()
- **Access** private

MLParser::$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";

- **Usedby** XMLParser::GetCharData()
- **Usedby** XMLParser::CreateTagArray()
- **Access** private

MLParser::$Tags

array = [line 133]

All interested tags

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

- **Usedby** XMLParser::CreateInputArray()
- **Usedby** XMLParser::ExplodeInputData()
- **Usedby** XMLParser::XMLParser()
- **Access** private

onstructor *XMLParser* function XMLParser::XMLParser($Data) *[line 171]*
Function Parameters:

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

Constructor of XMLParser class
The function initializes the fallowing 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** ColseTag()
- **Uses** XMLParser::$Tags
- **Uses** XMLParser::$Attributes
- **Uses** XMLParser::$Input
- **Uses** XMLParser::$Parser
- **Access** private

nction 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 constructer.

- **Uses** XMLParser::$Parser
- **Access** private

raph 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

nction 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

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

Findes the interested tag in XML Data

The function uses explode() function and the $Separator to finde 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

nction 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 constructer.

- **Usedby** XMLParser::XMLParser()
- **Uses** XMLParser::$TagData
- **Uses** XMLParser::$Parser
- **Access** private

nction 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

nction 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]

Evalution of expressions

Safe evaluation of mathematical expressions

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Usedby** Plot::DrawPlots()

valMath::$f

mixed = array() *[line 103]*

valMath::$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]*

valMath::$last_error

mixed = null *[line 100]*

valMath::$suppress_errors

mixed = false *[line 99]*

valMath::$v

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

valMath::$vb

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

onstructor function EvalMath::EvalMath() *[line 111]*
unction EvalMath::e($expr) *[line 117]*
Function Parameters:

- **$expr**

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

- **$expr**

    - **Usedby** Plot::DrawPlots()

unction EvalMath::funcs() *[line 168]*
unction EvalMath::nfx($expr) *[line 178]*
Function Parameters:

- **$expr**

nction EvalMath::pfx($tokens, [$vars = array()]) *[line 304]*

Function Parameters:

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

nction EvalMath::trigger($msg) *[line 366]*

Function Parameters:

- **$msg**

nction EvalMath::vars() *[line 161]*

Class EvalMathStack

[line 374]

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

alMathStack::$count

mixed = 0 *[line 377]*

alMathStack::$stack

mixed = array() *[line 376]*

nction EvalMathStack::last([$n = 1]) *[line 392]*

Function Parameters:

- **$n**

nction EvalMathStack::pop() *[line 384]*

nction 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

raph::$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

raph::$EnableLabel

boolean = true *[line 66]*

Enable label

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

- **Usedby** Plot::DrawPlots()
- **Access** public

raph::$Exp

string = [line 77]

Expression

The mathematical expression representing the graph.

- **Access** public
- **See** EvalMath::evaluate()

raph::$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

raph::$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

ot::$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

ot::$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

ot::$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

ot::$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

ot::$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 determains the color of the grid. This is an array of three integers, one for red, green and blue. Where integeres has 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

ot::$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

ot::$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::GetCoordinatY()
- **Usedby** Plot::GetImageY()
- **Usedby** Plot::DrawYGrid()
- **Usedby** Plot::DrawXGrid()
- **See** Plot::DrawPlots()
- **See** Plot::$MinY
- **Usedby** Plot::GetHash()
- **Access** public

ot::$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::GetCoordinatX()
- **Usedby** Plot::GetImageX()
- **Usedby** Plot::DrawYGrid()
- **Usedby** Plot::DrawXGrid()
- **See** Plot::DrawPlots()
- **See** Plot::$MaxX
- **Usedby** Plot::GetHash()
- **Access** public

ot::$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

ot::$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

ot::$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

ot::$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

nction 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 recommanded. Note: this changes the current http-request mimetype to the respective image mimetype.

- **Uses** Plot::GeneratePlot()
- **Access** public

nction 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

nction 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

nction 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

nction 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::GetCoordinatX()
- **Access** private

nction 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

nction 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

ageResource 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

eger 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

eger 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

ring function Plot::GetHash() *[line 260]*

Generate hash

Generates a uniqe 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

eger 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

eger 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

eger 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

eger 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

nction 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

ring 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

ache

Cache

valMath

EvalMath

valMathStack

EvalMathStack

raph

Graph

lot

Plot

MLParser

XMLParser

Appendix C - Source Code

Package WikiPlot

File Source for cache.class.php

ocumentation for this file is available at 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
blic License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
ter version.

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

    You should have received a copy of the GNU General Public License along with this program; if not, write to the
ee 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
velopers.
    *
    * @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)))
                {
                    $FileAge = filemtime($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)))
                {
                    $FileAge = fileatime($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 $FileName Filename relative to cache.
        * @return string Returns the URL of the file.
        */
        function FileURL($FileName)
        {
            if($this->   FileExist($FileName))
            {
                return WikiPlotCacheURL . "/"          . $FileName;
            }else{
                return false;
            }
        }

        /**
        * Get cache Path
        *
        * Get absolute path to the cache, returns false if FileName exists.
        *
        *@access public
        *@uses FileExist()
        *@param string $FileName Filename you want the path to, shortcut to detecting if file exists.
        *@return string Path to the cache, false if FileName exists.
        */
        function CachePath($FileName = null)
        {
            if(!is_null($FileName))
            {
                if($this->   FileExist($FileName))
                {
                    return false;
                }else{
                    return $_SERVER["DOCUMENT_ROOT"]          ] . WikiPlotCachePath . $FileName;
                }
            }else{
                return $_SERVER["DOCUMENT_ROOT"]          ] . WikiPlotCachePath;
            }
        }
    }


    ?>
```

File Source for CleanupCache.php

ocumentation 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
blic License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
ter version.

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

    You should have received a copy of the GNU General Public License along with this program; if not, write to the
ee 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

ocumentation for this file is available at [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 $value The string you wish to deserialize.
*@param boolean &$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. Usualy 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. Usualy 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
serialize 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(strstr($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
e 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);
        }

        $output = "     <   a href='$PlotURL' class='image' title='See the plot'><      img
c='$PlotURL'></      a>   "     ;

    /*
        //Render as wikitext:
        //To use external images this must be enabled: $wgAllowExternalImages = true; remeber to inform user.
        $localParser = new Parser();
        $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
        $text = $output->getText();
    */
        return $output;
    }

    ?>
```

File Source for WikiPlotSettings.php

ocumentation 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
ching functionallity.
*
* @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

ocumentation for this file is available at 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 exmple 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
     * iterested 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;
    /**
     * Constructor of XMLParser class
     *
     * The function initializes the fallowing 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().
     *
     * @access private
     * @param string $Data XML Input Data from user
     * @return XMLParser
     * @uses $Parser
     * @uses $Input
     * @uses $Tags
     * @uses $Attributes
     * @uses ExplodeInputData()
     * @uses Parse()
     * @uses OpenTag()
     * @uses ColseTag()
     * @uses GetCharData()
     */
    function XMLParser($Data)
    {
           //Initialize $Parser and creat an XML Parser to use later on
           $this->   Parser = xml_parser_create();
        //Initialize $Input and set it equal to $Data (XML from user)
           $this->   Input = $Data;
           //Initialize $Tags to be an array
           $this->   Tags = array();
           //Initialize $Attributes to be an array
           $this->   Attributes = array();
           //Initialize $Separator to be an array
           $this->   Separator = "<graph"             ;

           //Set XML Parser to use it within object
        xml_set_object($this->   Parser, $this);
        //Set up start and end element handlers for the parser
        xml_set_element_handler($this->   Parser, "OpenTag"          , "CloseTag"          );
        //Set up character data handler for the parser
        xml_set_character_data_handler($this->   Parser, "GetCharData"          );

        //Call ExplodeInputData() to get the interested tags
        $this->   ExplodeInputData();

        //Call Parse() to parse the $Input
        $this->   Parse($this->   Input);

        //Free the XML parser to later use
        xml_parser_free($this->   Parser);
    }

    /**
     * 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.
     *
     * @access private
     * @param string $Data
     * @uses CreateTagArray()
     */
    function Parse($Data)
    {
        //Parse XML Data using the $Parser
           xml_parse($this->   Parser, $Data);
        //Put returned values (Attribute(s) and TagData)
        //from XML praser into an array called $Tag
        $this->   CreateTagArray();
    }

    /**
     * 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.
     *
     * @access private
     * @uses $Attributes
     * @uses $TagData
     * @uses $Tag
     */
    function CreateTagArray()
    {
           if (!empty($this->   Attributes) && !empty(       $this->   TagData))
           {
              $this->   Tag = array($this->   Attributes, $this->   TagData);
           }
           else
           {
                 $this->   Tag = array($this->   TagData);
           }
    }

    /**
     * Findes the interested tag in XML Data
     *
     * The function uses explode() function and the $Separator to finde
     * the interested tag in the given XML Data. When the tags are found
     * it puts them into array called $Tags.
     *
     * @access private
     * @uses $Separator
     * @uses $Input
     * @uses $Tags
     */
    function ExplodeInputData()
    {
           //Split the given XML data by using $Separator
           $InterestedTags = explode($this->   Separator , $this->   Input);

           //Go through the array containing the interesting tags
           //NOTICE: $i must be = 1 because the array contains
           //nothing on 0 position
           //NOTICE: $ must be < lenght of the array and not <= because
           //the last element in the array is not interesting.
           for ($i=1; $i <    count($InterestedTags); $i++)
           {
                  //Put the $Separator into the tag
                  //(the separator vanishes when exploding the data)
                  //fx. If the separator is <tag. The following will take place.
                  //<tag>Hello</tag> will be exploded by <tag and
                  //returned as >Hello</tag>. To complete the tag
                  //we put the separator back on place. <tag + >Hello</tag>
                  //this will return the complete tag = <tag>Hello</tag>
                  array_push($this->   Tags, $this->   Separator . $InterestedTags[$i]);
           }
    }

    /**
     * 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.
     *
     * @access private
     * @param mixed $Parser
     * @param string $Tag
     * @param array $Attributes
     * @uses $Parser
     * @uses $Attributes
     */
    function OpenTag($Parser, $Tag, $Attributes)
    {
           //Check whether $Attributes is an array and is not an empty array
           if (is_array($Attributes) &&         count($Attributes) >    0)
           {
                   //Put $his->Attributes equal to $Attributes while changing the
                   //case of its key(s) to lowercase. The case of the key(s) is
                   //importan due to avoid error later on.
                   $this->   Attributes = array_change_key_case($Attributes, CASE_LOWER);
           }
```

```
            else
            {
                    //$this->Attributes will be an empty array() which is ignored
                    //when adding it to the general array which is return by the
                    //class!
            }
    }

    /**
     * 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.
     *
     * @access private
     * @param mixed $Parser
     * @param string $CharData
     * @uses $Parser
     * @uses $TagData
     */
    function GetCharData($Parser, $CharData)
    {
            //Set $this->TagData equal to $CharData
            //for later use.
               $this->   TagData = $CharData;
    }

    /**
     * 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
     * constructer.
     *
     * @access private
     * @param mixed $Parser
     * @param string $Tag
     * @uses $Parser
     */
    function CloseTag($Parser, $Tag)
    {
        //Have nothing do to! :(
        //But must be present.
    }

    /**
     * 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.
     *
     * @access buplic
     * @return $Graph
     * @uses $Tags
     * @uses XMLParser
     */
    function CreateInputArray()
    {
          //Create an array to store the parsed XML data in it
          //and then return it at the end of the proces.
       $Graph = array();

       //Get each interested tag from $Tags
          foreach( $this->   Tags as $Tag )
          {
              //Create instance of XMLParser and parse the
              //single tag to it
              $XMLParser = new XMLParser($Tag);
            //Store the data parsed by the XMLParser in the $Graph
              array_push($Graph, $XMLParser->   Tag);
          }

          //Return the $Graph to user
          return $Graph;
    }
}
?>
```

File Source for evalmath.class.php

ocumentation for this file is available at evalmath.class.php

```
<?
/**
* Evalution 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($expr)-1); // strip semicolons at the
d
        //===============
        // is it a variable assignment?
        if (preg_match('/^\s*([a-z]\w*)\s*=\s*(.+)$/', $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
d 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?
        } elseif (preg_match('/^\s*([a-z]\w*)\s*\(\s*([a-z]\w*(?:\s*,\s*[a-z]\w*)*)\s*\)\s*=\s*(.+)$/', $expr,
atches)) {
            $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]()'"     );
            }
            $args = explode(","         , preg_replace("/\s+/"         , ""         , $matches[2])); // get
e arguments
            if (($stack = $this->   nfx($matches[3])) === false) return false; // see if it can be converted to
stfix
            for ($i = 0; $i<   count($stack); $i++) { // freeze the state of the non-argument variables
                $token = $stack[$i];
                if (preg_match('/^[a-z]\w*$/', $token) and !in_array($token, $args)) {
                    if (array_key_exists($token, $this->   v)) {
                        $stack[$i] = $this->   v[$token];
                    } else {
                        return $this->   trigger("     undefined variable '$token' in function definition"     );
                    }
                }
            }
            $this->   f[$fnn] = array('args'=>   $args, 'func'=>   $stack);
            return true;
        //===============
        } else {
            return $this->   pfx($this->   nfx($expr)); // straight up evaluation, woo
        }
    }

    function vars() {
        $output = $this->   v;
        unset($output['pi']);
        unset($output['e']);
        return $output;
    }

    function funcs() {
        $output = array();
        foreach ($this->   f as $fnn=>   $dat)
            $output[] = $fnn . '(' . implode(',', $dat['args']) . ')';
        return $output;
    }

    //===================== HERE BE INTERNAL METHODS ====================\\

    // Convert infix to postfix notation

    function nfx($expr) {

        $index = 0;
        $stack = new EvalMathStack;
        $output = array(); // postfix form of expression, to be passed to pfx()
        $expr = trim(strtolower($expr));

        $ops   = array('+', '-', '*', '/', '^', '_');
        $ops_r = array('+'=>   0,'-'=>   0,'*'=>   0,'/'=>   0,'^'=>   1); // right-associative operator?
        $ops_p = array('+'=>   0,'-'=>   0,'*'=>   1,'/'=>   1,'_'=>   1,'^'=>   2); // operator precedence

        $expecting_op = false; // we use this in syntax-checking the expression
                               // and determining when a - is a negation

        if (preg_match("/[^\w\s+*^\/()\.,-]/"          , $expr, $matches)) { // make sure the characters are all
od
            return $this->   trigger("     illegal character '{$matches[0]}'"     );
        }

        while(1) { // 1 Infinite Loop ;)
            $op = substr($expr, $index, 1); // get the first character at the current index
            // find out if we're currently at the beginning of a number/variable/function/parenthesis/operand
            $ex = preg_match('/^([a-z]\w*\(?|\d+(?:\.\d*)?|\.\d+|\()/', substr($expr, $index), $match);
            //===============
            if ($op == '-' and !$expecting_op) { // is it a negation instead of a minus?
                $stack->   push('_'); // put a negation on the stack
                $index++;
            } elseif ($op == '_') { // we have to explicitly deny this, because it's legal on the stack
                return $this->   trigger("illegal character '_'"          ); // but not in the input expression
            //===============
            } elseif ((in_array($op, $ops) or $ex) and $expecting_op) { // are we putting an operator on the
ack?
                if ($ex) { // are we expecting an operator but have a number/variable/function/opening
rethesis?
                    $op = '*'; $index--; // it's an implicit multiplication
                }
                // heart of the algorithm:
                while($stack->   count >    0 and ($o2 = $stack->   last()) and in_array($o2, $ops) and
ops_r[$op] ? $ops_p[$op] <    $ops_p[$o2] : $ops_p[$op] <=    $ops_p[$o2])) {
                    $output[] = $stack->   pop(); // pop stuff off the stack into the output
                }
                // many thanks: http://en.wikipedia.org/wiki/Reverse_Polish_notation#The_algorithm_in_detail
                $stack->   push($op); // finally put OUR operator onto the stack
                $index++;
                $expecting_op = false;
            //===============
            } elseif ($op == ')' and $expecting_op) { // ready to close a parenthesis?
                while (($o2 = $stack->   pop()) != '(') { // pop off the stack back to the last (
                    if (is_null($o2)) return $this->   trigger("unexpected ')'"          );
                    else $output[] = $o2;
                }
                if (preg_match("     /^([a-z]\w*)\($/"     , $stack->   last(2), $matches)) { // did we just
    ose a function?
                    $fnn = $matches[1]; // get the function name
                    $arg_count = $stack->   pop(); // see how many arguments there were (cleverly stored on the
    ack, thank you)
                    $output[] = $stack->   pop(); // pop the function and push onto the output
                    if (in_array($fnn, $this->   fb)) { // check the argument count
                        if($arg_count >    1)
                            return $this->   trigger("     too many arguments ($arg_count given, 1
    pected)"     );
                    } elseif (array_key_exists($fnn, $this->   f)) {
                        if ($arg_count != count($this->   f[$fnn]['args']))
                            return $this->   trigger("     wrong number of arguments ($arg_count given, "      .
    unt($this->   f[$fnn]['args']) . " expected)"          );
                    } else { // did we somehow push a non-function on the stack? this should never happen
                        return $this->   trigger("internal error"          );
                    }
                }
                $index++;
            //===============
            } elseif ($op == ',' and $expecting_op) { // did we just finish a function argument?
                while (($o2 = $stack->   pop()) != '(') {
                    if (is_null($o2)) return $this->   trigger("unexpected ','"          ); // oops, never had a

                    else $output[] = $o2; // pop the argument expression stuff and push onto the output
                }
                // make sure there was a function
                if (!preg_match("     /^([a-z]\w*)\($/"     , $stack->   last(2), $matches))
                    return $this->   trigger("unexpected ','"          );
                $stack->   push($stack->   pop()+1); // increment the argument count
                $stack->   push('('); // put the ( back on, we'll need to pop back to it again
                $index++;
                $expecting_op = false;
            //===============
            } elseif ($op == '(' and !$expecting_op) {
                $stack->   push('('); // that was easy
                $index++;
                $allow_neg = true;
            //===============
            } elseif ($ex and !$expecting_op) { // do we now have a function/variable/number?
                $expecting_op = true;
                $val = $match[1];
                if (preg_match("     /^([a-z]\w*)\($/"     , $val, $matches)) { // may be func, or variable w/
    plicit multiplication against parentheses...
                    if (in_array($matches[1], $this->   fb) or array_key_exists($matches[1], $this->   f)) { //
    's a func
                        $stack->   push($val);
                        $stack->   push(1);
                        $stack->   push('(');
                        $expecting_op = false;
                    } else { // it's a var w/ implicit multiplication
                        $val = $matches[1];
                        $output[] = $val;
                    }
                } else { // it's a plain old var or num
                    $output[] = $val;
                }
                $index += strlen($val);
            //===============
            } elseif ($op == ')') { // miscellaneous error checking
                return $this->   trigger("unexpected ')'"          );
            } elseif (in_array($op, $ops) and !$expecting_op) {
                return $this->   trigger("     unexpected operator '$op'"     );
            } else { // I don't even want to know what you did to get here
                return $this->   trigger("an unexpected error occured"          );
            }
            if ($index == strlen($expr)) {
                if (in_array($op, $ops)) { // did we end with an operator? bad.
                    return $this->   trigger("     operator '$op' lacks operand"     );
                } else {
                    break;
                }
            }
            while (substr($expr, $index, 1) == ' ') { // step the index past whitespace (pretty much turns
itespace
                $index++;                             // into implicit multiplication if no operator is there)
            }

        }
        while (!is_null($op = $stack->   pop())) { // pop everything off the stack and push onto output
            if ($op == '(') return $this->   trigger("expecting ')'"          ); // if there are (s on the
ack, ()s were unbalanced
            $output[] = $op;
        }
        return $output;
    }

    // evaluate postfix notation

    function pfx($tokens, $vars = array()) {

        if ($tokens == false) return false;

        $stack = new EvalMathStack;

        foreach ($tokens as $token) { // nice and easy
            // if the token is a binary operator, pop two values off the stack, do the operation, and push the
sult back on
            if (in_array($token, array('+', '-', '*', '/', '^'))) {
                if (is_null($op2 = $stack->   pop())) return $this->   trigger("internal error"          );
                if (is_null($op1 = $stack->   pop())) return $this->   trigger("internal error"          );
                switch ($token) {
                    case '+':
                        $stack->   push($op1+$op2); break;
                    case '-':
                        $stack->   push($op1-$op2); break;
                    case '*':
                        $stack->   push($op1*$op2); break;
                    case '/':
                        if ($op2 == 0) return $this->   trigger("division by zero"          );
                        $stack->   push($op1/$op2); break;
                    case '^':
                        $stack->   push(pow($op1, $op2)); break;
                }
            // if the token is a unary operator, pop one value off the stack, do the operation, and push it back

            } elseif ($token == "_"          ) {
                $stack->   push(-1*$stack->   pop());
            // if the token is a function, pop arguments off the stack, hand them to the function, and push the
sult back on
            } elseif (preg_match("     /^([a-z]\w*)\($/"     , $token, $matches)) { // it's a function!
                $fnn = $matches[1];
                if (in_array($fnn, $this->   fb)) { // built-in function:
                    if (is_null($op1 = $stack->   pop())) return $this->   trigger("internal error"          );
                    $fnn = preg_replace("/^arc/"          , "a"          , $fnn); // for the 'arc' trig synonyms
                    if ($fnn == 'ln') $fnn = 'log';
                    eval('$stack->push('    . $fnn . '($op1));'); // perfectly safe eval()
                } elseif (array_key_exists($fnn, $this->   f)) { // user function
                    // get args
                    $args = array();
                    for ($i = count($this->   f[$fnn]['args'])-1; $i >=    0; $i--) {
                        if (is_null($args[$this->   f[$fnn]['args'][$i]] = $stack->   pop())) return $this-
  trigger("internal error"          );
                    }
                    $stack->   push($this->   pfx($this->   f[$fnn]['func'], $args)); // yay... recursion!!!!
                }
            // if the token is a number or variable, push it on the stack
            } else {
                if (is_numeric($token)) {
                    $stack->   push($token);
                } elseif (array_key_exists($token, $this->   v)) {
                    $stack->   push($this->   v[$token]);
                } elseif (array_key_exists($token, $vars)) {
                    $stack->   push($vars[$token]);
                } else {
                    return $this->   trigger("     undefined variable '$token'"     );
                }
            }
        }
        // when we're out of tokens, the stack should have a single element, the final result
        if ($stack->   count != 1) return $this->   trigger("internal error"         );
        return $stack->   pop();
    }

    // trigger an error, but nicely, if need be

    function trigger($msg) {
        $this->   last_error = $msg;
        if (!$this->   suppress_errors) trigger_error($msg, E_USER_WARNING);
        return false;
    }
}

// for internal use
class EvalMathStack {

    var $stack = array();
    var $count = 0;

    function push($val) {
        $this->   stack[$this->   count] = $val;
        $this->   count++;
    }

    function pop() {
        if ($this->   count >    0) {
            $this->   count--;
            return $this->   stack[$this->   count];
        }
        return null;
    }

    function last($n=1) {
        return $this->   stack[$this->   count-$n];
    }
}
```

File Source for graph.plot.class.php

ocumentation 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
blic License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
ter version.

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

    You should have received a copy of the GNU General Public License along with this program; if not, write to the
ee 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
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
ring, 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

ocumentation 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
blic License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any
ter version.

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

    You should have received a copy of the GNU General Public License along with this program; if not, write to the
ee 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
```

```
        * @see DrawPlots
        * @see MaxY
        */
        var $MinY = -10;
        /**
        * 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.
        *
        * @var integer
        * @access public
        * @see DrawPlots
        * @see MinY
        */
        var $MaxY = 100;

        /**
        * Enable Axis
        *
        * Defaults to true and draws 2 axis.
        *
        * @var boolean
        * @access public
        * @see DrawAxis
        */
        var $EnableAxis = true;
        /**
        * Enable Grid
        *
        * Defaults to true and draws a grid.
        *
        * @var boolean
        * @access public
        * @see DrawGrid
        */
        var $EnableGrid = true;
        /**
        * Grid color
        *
        * Defaults to gray, and determains the color of the grid. This is an array of three integers, one for red,
een and blue. Where integeres has values between 0 and 255.
        * <code>
        * var $Red = 240;
        * var $Green = 240;
        * var $Blue = 240;
        * $this->GridColor = array($Red,$Green,$Blue);
        * </code>
        *
        * @var array
        * @access public
        * @see DrawGrid
        */
        var $GridColor = array(240,240,240);
        /**
        * Grid font
        *
        * Font of the grids labels, the fonts 1-5 is built in, and behaves as different sizes.
        *
        * @var integer
        * @access public
        * @see DrawGrid
        */
        var $GridFont = 1;
        /**
        * X grid space
        *
        * Distance between grids on the x axis in coordinate space. Defaults to null, leave it null, if you want
togenerated gridspace.
        *
        * @var integer
        * @access public
        * @see GetXGridSpace
        */
        var $XGridSpace = null;
        /**
        * Y grid space
        *
        * Distance between grids on the y axis in coordinate space. Defaults to null, leave it null, if you want
```

```
togenerated gridspace.
        *
        * @var integer
        * @access public
        * @see GetYGridSpace
        */
        var $YGridSpace = null;
        /**
        * Background color
        *
        * Color of the background when using auto ImageResource created by GeneratePlot().
        *
        * @var array
        * @access public
        */
        var $BackgroundColor = array(255,255,255);

        /**
        *Generate hash
        *
        *Generates a uniqe hashsum (md5) for the plot, generated from all parameters.
        *
        *@uses $Caption
        *@uses $CaptionFont
        *@uses $Width
        *@uses $Height
        *@uses $MinX
        *@uses $MaxX
        *@uses $MinY
        *@uses $MaxY
        *@uses $EnableGrid
        *@uses $GridColor
        *@uses $GridFont
        *@uses $EnableAxis
        *@uses $XGridSpace
        *@uses $YGridSpace
        *@uses $Graphs
        *@uses Graph::GetHash()
        *@return string Hash representation of the object.
        */
        function GetHash()
        {
            $Hash  = "C:"           . $this->   Caption;
            $Hash .= "F:"           . $this->   CaptionFont;
            $Hash .= "W:"           . $this->   Width;
            $Hash .= "H:"           . $this->   Height;
            $Hash .= "X:"           . $this->   MinX . "_"           . $this->   MaxX;
            $Hash .= "Y:"           . $this->   MinY . "_"           . $this->   MaxY;
            $Hash .= "A:"           . $this->   EnableAxis;
            $Hash .= "G:"           . $this->   EnableGrid . "_"           . $this->   GridColor . "_"           .
his->   GridFont;
            $Hash .= "S:"           . $this->   XGridSpace . "_"           . $this->   YGridSpace;
            $Hash .= "V:"           . "     $LastChangedRevision: 63 $"     ;
            foreach($this->   Graphs as $key =>    $S)
            {
                $Hash .= "G:"           . $key. "_"           . $S->   GetHash();
            }
            return md5($Hash);
        }

        /**
        *Get ImageResource of the plot
        *
        *Generates ImageResource representation of the plot.
        *
        *@access public
        *@uses EnableGrid
        *@uses DrawGrid()
        *@uses $Width
        *@uses $Height
        *@uses $EnableAxis
        *@uses DrawAxis()
        *@uses DrawCaption()
        *@uses DrawPlots()
        *@uses $BackgroundColor
        *@param ImageResource $ImageResource Defaults to null, will generate empty ImageResource.
        *@param Boolean $ChangeSize May we change the size of the plot to fit given ImageResource?
        *@return ImageResource ImageResource representation of the plot.
        */
        function GeneratePlot($ImageResource = null, $ChangeSize = false)
```

```
    {
        //If ImageResource is null
        if(is_null($ImageResource))
        {
            //Get ImageResource
            $ImageResource = imagecreatetruecolor($this->   Height,$this->   Width);

            //AntiAlias ON
            imageantialias($ImageResource,true);

            //Fill the image with white
            imagefill($ImageResource,0,0,imagecolorexact($ImageResource,$this->   BackgroundColor[0],$this-
BackgroundColor[1],$this->   BackgroundColor[2]));

        }//If ImageResource doesn't fit image and we may not change size
        elseif($ChangeSize==false&&(        imagesx($ImageResource)!=$this-
Width||imagesy($ImageResource)!=$this->   Height))
        {
            //Get ImageResource
            $ImageResource = imagecreatetruecolor($this->   Height,$this->   Width);

            //AntiAlias ON
            imageantialias($ImageResource,true);

            //Fill the image with white
            imagefill($ImageResource,0,0,imagecolorexact($ImageResource,$this->   BackgroundColor[0],$this-
BackgroundColor[1],$this->   BackgroundColor[2]));
        }//If we may change the size of the plot
        elseif($ChangeSize)
        {
            //Changing size of the plot.
            $this->   Width = imagesx($ImageResource);
            $this->   Height = imagesy($ImageResource);
        }

        //If grid is enabled
        if($this->   EnableGrid)
        {
            $this->   DrawGrid($ImageResource);
        }

        //If axis is enabled
        if($this->   EnableAxis)
        {
            $this->   DrawAxis($ImageResource);
        }

        //Draw caption
        $this->   DrawCaption($ImageResource);

        //Draw plots
        $this->   DrawPlots($ImageResource);

        //Return ImageResource
        return $ImageResource;
    }

    /**
    *Get ImageResource of the plot
    *
    *Generates ImageResource representation of the plot.
    *
    *@access private
    *@uses $Width
    *@uses EvalMath
    *@uses EvalMath::evaluate()
    *@uses GetCoordinatX()
    *@uses GetImageX()
    *@uses GetImageY()
    *@uses $Graphs
    *@uses Graph::$Color
    *@uses Graph::$LabelFont
    *@uses Graph::$EnableLabel
    *@uses Graph::$Label
    *@param ImageResource &$ImageResource     ImageResource representation of the plot.
    */
    function DrawPlots(&    $ImageResource)
    {
        //Get a black Color
        $Black = imagecolorexact($ImageResource,0,0,0);
        //Y position for Labels relative to Image
        $LabelY = 5;

        //Plot all graphs
        foreach($this->   Graphs as $key =>    $S)
        {
            //Get Color
            $Color = imagecolorexact($ImageResource,$S->   Color[0],$S->   Color[1],$S->   Color[3]);

            //Set Expression
            $m = new EvalMath;
            $m->   evaluate("f(x) = "           . $S->   Exp);

            //Set OldCoordinat*, don't start with a line from 0,0
            $OldCoordinatX = $this->   GetCoordinatX(0);
            $OldCoordinatY = $m->   evaluate("f("          .$OldCoordinatX.")"         );

            //Plot the graph
            for($ImageX=0;$ImageX<   $this->   Width;$ImageX++)
            {

                //Get some NewCoordinat*
                $NewCoordinatX = $this->   GetCoordinatX($ImageX);
                $NewCoordinatY = $m->   evaluate("f("          .$NewCoordinatX.") "         );

                //Draw a line from OldCoordinat*
                imageline(
                    $ImageResource,
                    $this->   GetImageX($OldCoordinatX),
                    $this->   GetImageY($OldCoordinatY),
                    $this->   GetImageX($NewCoordinatX),
                    $this->   GetImageY($NewCoordinatY),
                    $Color);

                //Get some OldCoordinat*
                $OldCoordinatX = $NewCoordinatX;
                $OldCoordinatY = $NewCoordinatY;
            }

            //Draw label if it is enabled
            if($S->   EnableLabel)
            {
                //Draw label
                imagestring($ImageResource,$S->   LabelFont,5,$LabelY,"- "          .$S->   Label,$Color);

                //Add Label height to next Label X position
                $LabelY += imagefontheight($S->   LabelFont);
            }
        }
    }

    /**
    *Draw caption to ImageResource
    *
    *Draws the caption to an ImageResource representation of the plot.
    *
    *@access private
    *@uses $Width
    *@uses $Caption
    *@uses $CaptionFont
    *@param ImageResource &$ImageResource     ImageResource representation of the plot.
    */
    function DrawCaption(&    $ImageResource)
    {
        //Get a black color for caption
        $Black = imagecolorexact($ImageResource,0,0,0);

        //width of the caption
        $CaptionWidth = strlen($this->   Caption)*imagefontwidth($this->   CaptionFont);

        //X position of the caption, making it centered
        $X = ($this->   Width-$CaptionWidth)/2;

        //Draw the caption
        imagestring($ImageResource,$this->   CaptionFont,$X,0,$this->   Caption,$Black);
    }

    /**
    *Generates short numbers
    *
    *Rewrites numbers into scientific notation, with a certain maximum length.<br>
    *Example: ShortNumber(501000000) == 5.01e8
    *
    *@access private
    *@param integer $Number The number you wish to shorten.
    *@param integer $MaxLen The maximum length of the output default to 7.
    *@return string Scientific notation of the given Number at a certain length.
    */
    function ShortNumber($Number, $MaxLen = 7)
    {
        //If $Number isn't too long return it as it is
        if(strlen($Number)<=   $MaxLen)
        {
            return $Number;
        }else{
            //Convert to scientific notation
            $NSci = sprintf("%e"          ,$Number);

            //Follwing hack prevents the function from showing too many decimals
            $ArrNSci = split($NSci,"e"          );
            return round($ArrNSci[0],$MaxLen-5) . "e"           . $ArrNSci[1];
        }
    }

    /**
    *Get X grid space
    *
    *Returns X grid space, either calculated or from given value if given one.
    *
    *@access private
    *@uses $XGridSpace
    *@return integer The space between grid on x axes.
    */
    function GetXGridSpace()
    {
        if($this->   XGridSpace==null)
        {
            //Text length max 7 when using $this->ShortNumber();
            $XTextLen = 7*imagefontwidth($this->   GridFont) + 10;

            //Convering to coordinate space
            return (($this->   MaxX-$this->   MinX)/$this->   Width)*$XTextLen;
        }else{
            $XGridSpace = $this->   XGridSpace;
        }
        return $XGridSpace;
    }

    /**
    *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!
    *
    *@access private
    *@uses $YGridSpace
    *@return integer The space between grid on y axes.
    */
    function GetYGridSpace()
    {
        if($this->   YGridSpace==null)
        {
            //Text length max 7 when using $this->ShortNumber();
            $XTextLen = 7*imagefontwidth($this->   GridFont) + 10;

            //Convering to coordinate space
            return (($this->   MaxX-$this->   MinX)/$this->   Width)*$XTextLen;
        }else{
            $YGridSpace = $this->   YGridSpace;
        }
        return $YGridSpace;
    }

    /**
    *Draw grids
    *
    *Draws both x and y grid, using DrawXGrid() and DrawYGrid().
    *
    *@access private
    *@uses DrawXGrid()
    *@uses DrawYGrid()
    *@param ImageResource &$ImageResource     ImageResource representation of the plot.
    */
    function DrawGrid(&    $ImageResource)
    {
        $this->   DrawXGrid($ImageResource);
        $this->   DrawYGrid($ImageResource);
    }

    /**
    *Draws x-grid
    *
    *Drawing X grid on the plot.
    *
    *@access private
    *@uses GetXGridSpace()
    *@uses $GridColor
    *@uses $MinX
    *@uses $MaxX
    *@uses $MinY
    *@uses $MaxY
    *@uses GetImageX()
    *@uses GetImageY()
    *@uses $GridFont
    *@uses $Height
    *@uses ShortNumber()
    *@param ImageResource &$ImageResource     ImageResource representation of the plot.
    */
    function DrawXGrid(&    $ImageResource)
    {
        //Get grid width
        $XGridSpace = $this->   GetXGridSpace();

        //Get color to draw with
        $Color = imagecolorexact($ImageResource,$this->   GridColor[0],$this->   GridColor[1],$this-
GridColor[2]);
        //Get text color
        $Black = imagecolorexact($ImageResource,0,0,0);

        //Calculate start and end coordinats of the grid
        $XGridStart = ($this->   MinX-fmod($this->   MinX,$XGridSpace));
        $XGridEnd = $this->   MaxX-fmod(($this->   MaxX-$this->   MinX),$XGridSpace);

        //Draw the grid
        for($XCordinate=$XGridStart;$XCordinate<   $XGridEnd;$XCordinate+=$XGridSpace)
        {
            imageline(
                $ImageResource,
                $this->   GetImageX($XCordinate),
                $this->   GetImageY($this->   MinY),
                $this->   GetImageX($XCordinate),
                $this->   GetImageY($this->   MaxY),
                $Color);

            //If Y axes is not on the image (working in ImageSpace not CoordinatSpace)
            $Y = $this->   GetImageY(0);
            if($Y > (   $this->   Height-imagefontheight($this->   GridFont)))
            {
                $Y = $this->   Height-(imagefontheight($this->   GridFont)+2);
            }else{
                if($Y<   0)
                {
                    $Y = 0;
                }
            }
            imagestring(
                $ImageResource,
                $this->   GridFont,
                $this->   GetImageX($XCordinate)+2,
                $Y+2,
                $this->   ShortNumber($XCordinate),
                $Black);
        }
    }

    /**
    *Draws y-grid
    *
    *Drawing y grid on the plot.
    *
    *@access private
    *@uses GetYGridSpace()
    *@uses $GridColor
    *@uses $MinX
    *@uses $MaxX
    *@uses $MinY
    *@uses $MaxY
    *@uses GetImageX()
    *@uses GetImageY()
    *@uses $GridFont
    *@uses $Width
    *@uses ShortNumber()
    *@param ImageResource &$ImageResource     ImageResource representation of the plot.
    */
    function DrawYGrid(&    $ImageResource)
    {
        //Get grid width
        $YGridSpace =  $this->   GetYGridSpace();

        //Get color to draw with
        $Color = imagecolorexact($ImageResource,$this->   GridColor[0],$this->   GridColor[1],$this-
GridColor[2]);
        //Get text color
        $Black = imagecolorexact($ImageResource,0,0,0);

        //Calculate start and end coordinats of the grid
        $YGridStart = ($this->   MinY-fmod($this->   MinY,$YGridSpace));
        $YGridEnd = $this->   MaxY-fmod(($this->   MaxY-$this->   MinY),$YGridSpace);

        //Draw the grid
        for($YCordinate=$YGridStart;$YCordinate<   $YGridEnd;$YCordinate+=$YGridSpace)
        {
            imageline(
                $ImageResource,
                $this->   GetImageX($this->   MinX),
                $this->   GetImageY($YCordinate),
                $this->   GetImageX($this->   MaxX),
                $this->   GetImageY($YCordinate),
                $Color);

            //If X axes is not on the image (working in ImageSpace not CoordinatSpace)
            $X = $this->   GetImageX(0);
            if($X > (   $this->   Width-(imagefontwidth($this->   GridFont)*7)))
            {
                $X = $this->   Width-(imagefontwidth($this->   GridFont)*7+2);
            }else{
                if($X<   0)
                {
                    $X = 0;
                }
            }
            imagestring(
                $ImageResource,
                $this->   GridFont,
                $X+2,
                $this->   GetImageY($YCordinate)+2,
                $this->   ShortNumber($YCordinate),
                $Black);
        }
    }

    /**
    * Draw axis
    *
    * Draw both x and y axis to the plot.
    *
    *@access private
    *@uses $MinX
    *@uses $MaxX
    *@uses $MinY
    *@uses $MaxY
    *@uses GetImageX()
    *@uses GetImageY()
    *@param ImageResource &$ImageResource     ImageResource representation of the plot.
    */
    function DrawAxis(&    $ImageResource)
    {
        $Black = imagecolorexact($ImageResource,0,0,0);
        //Draw X-axis
        imageline(
            $ImageResource,
            $this->   GetImageX(0),
            $this->   GetImageY($this->   MinY),
            $this->   GetImageX(0),
            $this->   GetImageY($this->   MaxY),
            $Black);
        //Draw Y-axis
        imageline($ImageResource,
            $this->   GetImageX($this->   MinX),
            $this->   GetImageY(0),
            $this->   GetImageX($this->   MaxX),
            $this->   GetImageY(0),
            $Black);
    }

    /**
    *Display plot as image
    *
    *Displays plot as image on the page. This makes current http-request return an image. You can set the
splayType to png, gif or jpeg. Defaults to png, gif not recommanded. Note: this changes the current http-request
metype to the respective image mimetype.
    *
    *@access public
    *@uses GeneratePlot()
    *@param string $DisplayType Type of image to view (png|jpeg|gif).
    *@param ImageResource $ImageResource Defaults to null, will generate empty ImageResource.
    *@param Boolean $ChangeSize May we change the size of the plot to fit given ImageResource?
    */
    function DisplayPlot($DisplayType = "png"          ,$ImageResource = null, $ChangeSize = false)
    {
        if($DisplayType == "png"          )
        {
            header("Content-type: image/png"          );
            imagepng($this->   GeneratePlot($ImageResource, $ChangeSize));
        }
        elseif($DisplayType == "gif"          )
        {
            header("Content-type: image/gif"          );
            imagegif($this->   GeneratePlot($ImageResource, $ChangeSize));
        }
        else
        {
            header("Content-type: image/jpeg"          );
            imagejpeg($this->   GeneratePlot($ImageResource, $ChangeSize));
        }
    }

    /**
    *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.
    *
    *@access public
    *@uses GeneratePlot()
    *@param string $Path Path of file to save.
    *@param string $SaveAs Filetype definition (png|jpeg|gif).
    *@param ImageResource $ImageResource Defaults to null, will generate empty ImageResource.
    *@param Boolean $ChangeSize May we change the size of the plot to fit given ImageResource?
    */
    function SaveAs($Path,$SaveAs = "png"          ,$ImageResource = null, $ChangeSize = false)
    {
        if($SaveAs == "png"          )
        {
            imagepng($this->   GeneratePlot($ImageResource, $ChangeSize),$Path);
        }
        elseif($SaveAs == "gif"          )
        {
            imagegif($this->   GeneratePlot($ImageResource, $ChangeSize),$Path);
        }
        else
        {
            imagejpeg($this->   GeneratePlot($ImageResource, $ChangeSize),$Path);
        }
    }

    /**
    * Convert to coordinate space
    *
    * Converts an x image position to x coordinate position. Coordinate space may differ from Image space, if
```

```
dth!= (MaxX-MinX).
        *
        *@access private
        *@uses $MaxX
        *@uses $MinX
        *@uses $Width
        *@param integer $x X image coordinat to be converted.
        *@return integer Coordiante space representation given parameter.
        */
        function GetCoordinatX($x)
        {
            return (($this->   MaxX-$this->   MinX)/$this->   Width)*$x+$this->   MinX;
        }

        /**
        * Convert to coordinate space
        *
        * Converts an y image position to y coordinate position. Coordinate space may differ from Image space, if
ight!= (MaxY-MinY).
        *
        *@access private
        *@uses $MaxY
        *@uses $MinY
        *@uses $Height
        *@param integer $y Y image coordinat to be converted.
        *@return integer Coordiante space representation given parameter.
        */
        function GetCoordinatY($y)
        {
            return (($this->   MaxY-$this->   MinY)/$this->   Height)*($this->   Height-$y)+$this->   MinY;
        }

        /**
        * Convert to image space
        *
        * Converts an x in coordinate space to x image position. Coordinate space may differ from Image space, if
dth!= (MaxX-MinX).
        *
        *@access private
        *@uses $MaxX
        *@uses $MinX
        *@uses $Width
        *@param integer $x X coordinat to be converted.
        *@return integer Image position representation given parameter.
        */
        function GetImageX($x)
        {
            return ($x-$this->   MinX)*($this->   Width/($this->   MaxX-$this->   MinX));
        }

        /**
        * Convert to image space
        *
        * Converts an y in coordinate space to y image position. Coordinate space may differ from Image space, if
ight!= (MaxY-MinY).
        *
        *@access private
        *@uses $MaxY
        *@uses $MinY
        *@uses $Height
        *@param integer $y Y coordinat to be converted.
        *@return integer Image position representation given parameter.
        */
        function GetImageY($y)
        {
            return $this->   Height-($y-$this->   MinY)*($this->   Height/($this->   MaxY-$this->   MinY));
        }
    }
    ?>
```

File Source for test.php

ocumentation 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