WikiPlot Documentation

Contents

WikiPlot Administrators Guide

A System Administrators Guidelines

Authors:*Jonas F. Jensen.*

WikiPlot@Jopsen.dk

This guide is intended for system administrators. It will help you install WikiPlot, and maintain you installation updated, with as little effort as possible. If you are maintainer/administrator of a MediaWiki installation, this is your guide for WikiPlot.

Installation

WikiPlot can be installed as any other MediaWiki extension, first you obtain a copy of WikiPlot, we recommend that you download the latest stable from our homepage. You can also checkout a(n) (un)stable from SVN, but if you do that you must also rename some directories and stuff, which is not covered here.

Once you have obtained a stable copy of WikiPlot, extract it and place the WikiPlot directory in the extensions directory of your MediaWiki installation. Now you will have to activate you extension in the LocalSettings.php file of your MediaWiki installation. This is done by adding following line:

```
//Enable the WikiPlot extension
require_once("extensions/WikiPlot/WikiPlot.php");
```

Now you have installed the WikiPlot extension, but you NOT all done yet. WikiPlot requires access to a cache, before it can work. Configuration of WikiPlot and its cache is covered next.

Settings/Caching

Once you have installed WikiPlot, you must configure its cache before you can use WikiPlot. First you must have a place to save cached images, create an empty directory and give php write permissions to the directory (hint: the directory must be accessible from http). Once you have create a directory for the cache, you must tell WikiPlot where this directory is located, you do that by opening WikiPlotSettings.php from extensions/WikiPlot/WikiPlotSettings.php, and redefining the constants defined there. Like this:

<ul mark = "bullet">

- **WikiPlotCachePath**

  The path to the cache directory you have just created. This path is relative to DOCUMENT_ROOT, you DOCUMENT_ROOT is usually /public_html/, which means that if your cache directory is in /public_html/catch/ then WikiPlotCachePath should be /cache/. You can find you DOCUMENT_ROOT in your phpinfo() data.
- **WikiPlotCacheURL**

The path to the cache directory same directory as used in WikiPlotCachePath, but this time your path is relative to http://, which means that if you site is mysite.com and you cache directory is placed in DOCUMENT_ROOT (usually /public_html/) then the URL (WikiPlotCacheURL) to your cache should be http://mysite.com/cache/

More specific configuration of you cache can be done in WikiPlotSettings.php, if you have any questions regarding these settings, take a look at the sourcecode documentation or feel free to contact us. More advanced configuration of the cache is not covered here.

Keep the cache clean

If your users uses WikiPlot a lot, and changes the plots a lot, WikiPlot will generate an awful lot of unused images in the cache, at some point this might be considered a waste of server resources. Therefore it is recommend that you cleanup the catch on a regular basis. You can cleanup the cache by running /extensions/WikiPlot/CleanupCache.php, you may run this script as a cron job.

The script removes all images that have not been accessed for a given period. It is also possible to remove all images of a certain age. You can find the settings for this file in WikiPlotSettings.php, advanced configuration of this script is not covered here, see sourcecode documentation for further information.

Stay updated

Once you have installed WikiPlot, you might want to make sure that you installation is kept updated, both for security and performance reasons. And perhaps there will come some new features as well. To stay updated on the WikiPlot releases subscribe to our announcement list, members of this list will only be contacted regarding new releases, and in case of serious security issues. Subscribe and stay updated.

Support/Help

If you have any questions or problems with WikiPlot, feel free to contact us, you can use our development mailinglist. It is not just a list for WikiPlot development, but a list for everything regarding WikiPlot, help, bug, questions and what ever.

WikiPlot Userguide

Introduction to WikiPlot

Authors:*Jonas F. Jensen.*

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 would not be surprised if you would consider that a little too basic. So just to match some basic functionality, we are going to add an other graph with the expression y(x)=3*x-3, and some optional parameters to modify the output image.

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

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

To simplify the example, I have divided and explained it here:

<ul mark = "bullet">

- **height**

  Height of the output image in pixels.
- **width**

  Width of the output image in pixels.
- **caption**

  Caption on the output image.
- **xspan and yspan**

  Values representing minimum x and maximum x, in coordinate space. It you set xspan="-50;75" the lowest x values on you image will be -50 and the highest will be 75. This does not have anything with width to do, and is in no way related to pixels! This feature enables you to zoom in and out on the coordinate space, independent of image size. xspan and yspan are completely similar except for the fact that they change the y or x coordinate space respectively.
- **x+4 and 3*x-3**

  These are mathematical expressions defining the 2 graphs on the image.
- **label**

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

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

If you do not understand this, please feel free to contact us, or post 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 confuse users, so please help us improve.

WikiPlot Syntax Reference

Complete WikiPlot syntax reference

Authors:*Jonas F. Jensen.*

WikiPlot@Jopsen.dk

WikiPlot Syntax Reference

This is a complete syntax reference for WikiPlot, if you are note familiar with xml or the most common WikiPlot syntax, it is recommend that you read the WikiPlot UserGuide first. Below you will find a documentation of the parameters and content for the wikiplot and graph tag, respectively. At the end of the article you will find a complete example of a plot with use of all parameters.

wikiplot parameters and content

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

<ul mark = "bullet">

caption

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

defines coordinate space to be viewed.

yspan

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

axis

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

grid

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

gridspace

Two semicolon separated integers, defining the space between the 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 grid-lines on the x-axis will be 10 and the distance between the grid-lines on the x-axis will be 20.

gridfont

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

gridcolor

Three semicolon separated integers, defining the color of the grid-lines, 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

he graph tags represents different mathematical expressions, that are to be plotted onto the coordinate-system defined by the rrounding/parent wikiplot tag. The graph tag contains the mathematical expression, it is representing. This mathematical pression 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
und in evalmath.class.php. We have not documented this class because we have not written it. The graph tag also takes
rtain parameters that allow you to affect the way it is represented on the plot. The graph tag take following parameters:

<ul mark = "bullet">
bel

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.
lor

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

Complete Example

ing is an advanced example of how WikiPlot could be used. Normally you don't need to used all parameters, most basic ones
vered in depth in the WikiPlot Userguide. This is a pretty extreme example of how to use all parameters:

```
lot height="400" width="800" caption="Complete Example"
pan="-100;100" yspan="-200;200" gridspace="10;20"
tionfont="4" gridfont="2" axis="true" grid="true">
aph label="A red graph" color="255,0,0">x^2+4</graph>
aph label="A blue graph" color="0,0,255">3*x-3</graph>
plot>
```

Package WikiPlot Procedural Elements

cache.class.php

e used to control cache

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

- **Package** WikiPlot
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

e_once **"WikiPlotSettings.php"** *[line 29]*

quire local settings

This file is needed to control the cache correctly.

CleanupCache.php

e 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
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

e_once **"cache.class.php"** *[line 28]*

quire cache class

This class is needed to control the cache.

WikiPlot.php

e MediaWiki extension

This is the MediaWiki extension it self, everything else is just functions and liberaries for this file.

- **Package** WikiPlot
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

unction RenderWikiPlot($input, $argv, [$parser = null]) *[line 198]*

nction Parameters:

string **$input** The content of the wikiplot tag
array **$argv** Hash-array of the parameters of the wikiplot tag, with parameter-name as key and parameter-value as value.
Parser **$parser** The parser of MediaWiki, if null parser is obtained from global variable

nderWikiPlot CallBack function

This is the function that handles MediaWiki callbacks, and renders the actual plot.

- **Uses** WikiPlotDeserializeInteger()
- **Uses** WikiPlotDeserializeMixed()
- **Uses** WikiPlotDeserializeString()
- **Uses** XMLParser
- **Uses** WikiPlotDeserializeColor()
- **Uses** WikiPlotDeserializeBoolean()
- **Uses** Cache
- **Uses** Graph
- **Uses** Plot
- **Access** private

on wfWikiPlotExtension() *[line 57]*

d hooks

Adds hooks so MediaWiki will perform callback, when it hits the wikiplot tag.

on WikiPlotDeserializeBoolean($value, &$SetTo) *[line 72]*
nction Parameters:

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

serialize boolean

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

- **Usedby** RenderWikiPlot()
- **Access** private

on WikiPlotDeserializeColor($value, &$SetTo) *[line 159]*
nction Parameters:

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

serialize Color

Deserializes an array representation of a rgb color from string, this function is used when you want to erialize parameters given in the WikiML. This function can deserialize colors written as 5,255,255" (rgb) or "#000000" (hex). If it is impossible to deserialize the value, the
put object is not initialized at all.

- **Usedby** RenderWikiPlot()
- **Access** private

on WikiPlotDeserializeInteger($value, &$SetTo) *[line 137]*
nction Parameters:

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

serialize Integer

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

- **Usedby** RenderWikiPlot()
- **Access** private

on WikiPlotDeserializeMixed($value, &$SetTo1, &$SetTo2) *[line 114]*
nction 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.

serialize Coordiante

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

- **Usedby** RenderWikiPlot()
- **Access** private

on WikiPlotDeserializeString($value, &$SetTo) *[line 94]*
nction Parameters:

string **$value** The string you wish to deserialize.

string **&$SetTo** The variable you want the values parsed to.

serialize String

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

- **Usedby** RenderWikiPlot()
- **Access** private

e_once **"PlotClass/plot.class.php"** *[line 29]*

ude plot.class.php

Requires PlotClass to render plots.

e_once **"xml.class.php"** *[line 36]*

ude xml.class.php

Requires XMLParser to parse xml to plot.

e_once **"cache.class.php"** *[line 43]*

ude cache.class.php

Requires Cache to control the cache.

WikiPlotSettings.php

e used to store settings

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

- **Package** WikiPlot
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file
- **Author** WikiPlot development team.

otCacheAge = 0 *[line 39]*

x Cache Age

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

- **Var** Cache age in days.
- **Throws** no exceptions thrown

otCachePath = "./cache/" *[line 19]*

h to the cache

Path to the cache, relative to the DOCUMENT_ROOT.

- **Var** Path relative to DOCUMENT_ROOT
- **Throws** no exceptions thrown
- **See** $CacheURL

otCacheURL = "http://example.com/cache/" *[line 29]*

L to cache

URL to cache directory define in $CachePath.

- **Var** absolute url
- **Throws** no exceptions thrown
- **See** $CachePath

otMaxUnusedAge = 14 *[line 48]*

x Unused Age

Maximun unused age before deletion.

- **Var** Age in days.
- **Throws** no exceptions thrown

xml.class.php

file contains XMLParser class

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

- **Package** WikiPlot
- **Throws** no exceptions thrown
- **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]

che controlling class

Class used to control the cache.

- **Package** WikiPlot
- **Throws** no exceptions thrown
- **Usedby** RenderWikiPlot()
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

unction Cache::CachePath([$FileName = null]) *[line 164]*

nction Parameters:

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

cache Path

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

- **Throws** no exceptions thrown
- **Uses** Cache::FileExist()

- **Access** public

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

anup the cache

Cleans up the cache by removing old and unused files.

- **Throws** no exceptions thrown
- **Uses** Cache::CleanupUnused()
- **Uses** Cache::CleanupMaxAge()
- **Access** public

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

anup cache from old files

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

- **Throws** no exceptions thrown
- **Usedby** Cache::CleanupCache()
- **Access** public

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

anup unused files from cache

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

- **Throws** no exceptions thrown
- **Usedby** Cache::CleanupCache()
- **Access** public

n function Cache::FileExist($FileName) *[line 129]*
nction Parameters:

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

es file exist in cache

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

- **Throws** no exceptions thrown
- **Usedby** Cache::CachePath()
- **Usedby** Cache::FileURL()
- **Access** public

unction Cache::FileURL($FileName) *[line 144]*
nction Parameters:

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

file URL

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

- **Throws** no exceptions thrown
- **Uses** Cache::FileExist()
- **Access** public

Class XMLParser

[line 65]

LParser class

This class parses a given XML data to a multidimensional array by using a user-defined tag. The ault 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** RenderWikiPlot()
- **Throws** no exceptions thrown
- **Usedby** XMLParser::CreateInputArray()
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

arser::$Attributes

rray = [line 122]

ributes of interested tag

The variable will always be an array whether the interested tag has any attributes or not. If the rested 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

arser::$Input

tring = [line 82]

L data given by user

Stores the XML data given by user as it is

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

arser::$Parser

nixed = [line 73]

ated 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

arser::$Separator

tring = [line 157]

e interested tag

The variable is our iterested tag. It means the tag that we are iterested to finde in the given XML data. e way you should definde your interested tag is as follows: If your interested tag is <Tag> than you uld change the $Separator variable to XMLParser::Separator = "<Tag" not ag>" or something else!

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

arser::$Tag

rray = [line 111]

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
)
```

u 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 d 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 wo elements, while one element is possible.

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

arser::$TagData

rray = [line 133]

a of the tag

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

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

- **Access** private

arser::$Tags

rray = [line 143]

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

uctor *XMLParser* function XMLParser::XMLParser($Data) *[line 181]*

nction Parameters:

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

nstructor of XMLParser class

The function initializes the fallowing variables: $Parser, $Input, $Tags, $Attributes and $Separator. It kes it possible to use XML Parser within an object by using the function xml_set_object. Besides it uses o two more XML Parser Functions xml_set_element_handler(), xml_set_character_data_handler() and _parser_free().

- **Uses** XMLParser::GetCharData()
- **Uses** XMLParser::OpenTag()
- **Uses** XMLParser::Parse()
- **Throws** no exceptions thrown
- **Uses** XMLParser::ExplodeInputData()
- **Uses** ColseTag()
- **Uses** XMLParser::$Attributes
- **Uses** XMLParser::$Input
- **Uses** XMLParser::$Parser
- **Uses** XMLParser::$Tags
- **Access** private

on XMLParser::CloseTag($Parser, $Tag) *[line 358]*
nction Parameters:

mixed **$Parser**
string **$Tag**

ndles end/closing tag

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

- **Throws** no exceptions thrown
- **Uses** XMLParser::$Parser
- **Access** private

h function XMLParser::CreateInputArray() *[line 377]*

ates an array containing all parsed XML data

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

- **Throws** no exceptions thrown
- **Uses** XMLParser
- **Uses** XMLParser::$Tags

on XMLParser::CreateTagArray() *[line 247]*

s parsed data into an array

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

- **Usedby** XMLParser::Parse()
- **Throws** no exceptions thrown
- **Uses** XMLParser::$TagData
- **Uses** XMLParser::$Tag
- **Uses** XMLParser::$Attributes
- **Access** private

on XMLParser::ExplodeInputData() *[line 271]*

des the interested tag in XML Data

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

- **Usedby** XMLParser::XMLParser()
- **Throws** no exceptions thrown
- **Uses** XMLParser::$Tags
- **Uses** XMLParser::$Separator
- **Uses** XMLParser::$Input
- **Access** private

on XMLParser::GetCharData($Parser, $CharData) *[line 339]*

nction Parameters:

mixed **$Parser**
string **$CharData**

s data of the tag

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

- **Throws** no exceptions thrown
- **Usedby** XMLParser::XMLParser()
- **Uses** XMLParser::$TagData
- **Uses** XMLParser::$Parser
- **Access** private

on XMLParser::OpenTag($Parser, $Tag, $Attributes) *[line 308]*
nction Parameters:

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

ndles attribute(s) of a tag

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

- **Throws** no exceptions thrown
- **Usedby** XMLParser::XMLParser()
- **Uses** XMLParser::$Parser
- **Uses** XMLParser::$Attributes
- **Access** private

on XMLParser::Parse($Data) *[line 224]*
nction Parameters:

string **$Data**

ses the given XML data

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

- **Throws** no exceptions thrown
- **Usedby** XMLParser::XMLParser()
- **Uses** XMLParser::CreateTagArray()
- **Access** private

evalmath.class.php

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file

graph.plot.class.php

containing Graph representation

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

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

plot.class.php

e use to draw plots

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

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown
- **Filesource** Source Code for this file
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

e_once **'graph.plot.class.php'** *[line 37]*

udes Graph representation class

Graph is used as a representation of a graph.

e_once **'evalmath.class.php'** *[line 30]*

udes EvalMath

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

Class EvalMath

[line 97]

alution of expressions

Safe evaluation of mathematical expressions

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown
- **Usedby** Plot::DrawPlots()

ath::$f

ixed = array() *[line 103]*

ath::$fb

ixed = 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]*

ath::$last_error

ixed = null *[line 100]*

ath::$suppress_errors

ixed = false *[line 99]*

ath::$v

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

ath::$vb

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

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

- **Throws** no exceptions thrown

on EvalMath::e($expr) *[line 117]*
nction Parameters:

$expr

- **Throws** no exceptions thrown

on EvalMath::evaluate($expr) *[line 121]*
nction Parameters:

$expr

- **Throws** no exceptions thrown
- **Usedby** Plot::DrawPlots()

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

- **Throws** no exceptions thrown

on EvalMath::nfx($expr) *[line 178]*
nction Parameters:

$expr

- **Throws** no exceptions thrown

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

$tokens
$vars

- **Throws** no exceptions thrown

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

$msg

- **Throws** no exceptions thrown

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

- **Throws** no exceptions thrown

Class EvalMathStack

[line 374]

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown

athStack::$count

ixed = 0 *[line 377]*

athStack::$stack

ixed = array() *[line 376]*

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

nction Parameters:

$n

- **Throws** no exceptions thrown

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

- **Throws** no exceptions thrown

on EvalMathStack::push($val) *[line 379]*
nction Parameters:

$val

- **Throws** no exceptions thrown

Class Graph

[line 36]

presentation of a graph

Class used to represente graphs on a plot.

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown
- **Usedby** RenderWikiPlot()
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

::$Color

rray = array(0,0,0) *[line 88]*

or of the graph

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

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

::$EnableLabel

oolean = true *[line 66]*

able label

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

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

::$Exp

tring = *[line 77]*

pression

The mathematical expression representing the graph.

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

::$Label

tring = [line 46]

el 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

::$LabelFont

nteger = 2 [line 56]

t 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

unction Graph::GetHash() *[line 98]*

hash

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

- **Throws** no exceptions thrown
- **Usedby** Plot::GetHash()
- **Access** private

Class Plot

[line 50]

ss used to draw plots

Class containing functions to draw plots to an image.

- **Package** WikiPlot
- **Sub-Package** PlotClass
- **Throws** no exceptions thrown
- **Usedby** RenderWikiPlot()
- **Copyright** Copyright 2006, WikiPlot development team.
- **Author** WikiPlot development team.
- **License** GNU General Public License

$BackgroundColor

rray = array(255,255,255) *[line 235]*

ckground color

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

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

$Caption

tring = null *[line 73]*

tion of the plot

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

- **Usedby** Plot::DrawCaption()
- **Usedby** Plot::GetHash()
- **See** Plot::DrawCaption()
- **Access** public

CaptionFont

nteger = 5 *[line 84]*

tion 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

EnableAxis

oolean = true *[line 170]*

ble Axis

Defaults to true and draws 2 axis.

- **Usedby** Plot::GeneratePlot()
- **Usedby** Plot::GetHash()
- **See** Plot::DrawAxis()
- **Access** public

$EnableGrid

oolean = true *[line 180]*

able Grid

Defaults to true and draws a grid.

- **Usedby** Plot::GeneratePlot()
- **Usedby** Plot::GetHash()
- **See** Plot::DrawGrid()
- **Access** public

$Graphs

rray = array() *[line 61]*

aphs to plot

Array containing list of Graphs to plot.

- **Usedby** Plot::DrawPlots()
- **Usedby** Plot::GetHash()
- **See** Graph
- **Access** public

$GridColor

rray = array(240,240,240) *[line 196]*

d color

Defaults to gray, and determains the color of the grid. This is an array of three integers, one for red, en 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

$GridFont

Integer = 1 *[line 206]*

d 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

$Height

Integer = 100 *[line 106]*

ght 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

$MaxX

nteger = 100 *[line 133]*

ximum X

Maximum X in coordinate space. Together with MinX this variable defines width of the plot in coordinate ce. 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

$MaxY

nteger = 100 *[line 159]*

ximum Y

Maximum Y in coordinate space. Together with MinY this variable defines height of the plot in rdinate 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

$MinX

nteger = -10 *[line 120]*

imum X

Minimum X in coordinate space. Together with MaxX this variable defines width of the plot in coordinate ce. 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

$MinY

nteger = -10 *[line 146]*

imum Y

Minimum Y in coordinate space. Together with MaxY this variable defines height of the plot in rdinate 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

$Width

nteger = 100 *[line 95]*

lth 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

sXGridSpace

nteger = null *[line 216]*

rid space

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

- **Usedby** Plot::GetXGridSpace()
- **Usedby** Plot::GetHash()
- **See** Plot::GetXGridSpace()
- **Access** public

sYGridSpace

nteger = null *[line 226]*

rid space

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

- **Usedby** Plot::GetYGridSpace()
- **Usedby** Plot::GetHash()
- **See** Plot::GetYGridSpace()
- **Access** public

on Plot::DisplayPlot([$DisplayType = "png"], [$ImageResource = null], [$ChangeSize = false]) *[line 721]*
nction 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?

play plot as image

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

- **Throws** no exceptions thrown
- **Uses** Plot::GeneratePlot()
- **Access** public

on Plot::DrawAxis(&$ImageResource) *[line 690]*
nction Parameters:

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

w axis

Draw both x and y axis to the plot.

- **Uses** Plot::GetImageY()
- **Usedby** Plot::GeneratePlot()
- **Throws** no exceptions thrown
- **Uses** Plot::GetImageX()
- **Uses** Plot::$MinY
- **Uses** Plot::$MaxX
- **Uses** Plot::$MaxY
- **Uses** Plot::$MinX
- **Access** private

on Plot::DrawCaption(&$ImageResource) *[line 439]*
nction Parameters:

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

w caption to ImageResource

Draws the caption to an ImageResource representation of the plot.

- **Usedby** Plot::GeneratePlot()
- **Throws** no exceptions thrown
- **Uses** Plot::$Width
- **Uses** Plot::$CaptionFont
- **Uses** Plot::$Caption
- **Access** private

on Plot::DrawGrid(&$ImageResource) *[line 540]*
nction Parameters:

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

w grids

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

- **Throws** no exceptions thrown
- **Usedby** Plot::GeneratePlot()
- **Uses** Plot::DrawYGrid()
- **Uses** Plot::DrawXGrid()
- **Access** private

on Plot::DrawPlots(&$ImageResource) *[line 372]*

nction Parameters:

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

ImageResource of the plot

Generates ImageResource representation of the plot.

- **Uses** Graph::$EnableLabel
- **Uses** Graph::$Color
- **Uses** Graph::$Label
- **Uses** Graph::$LabelFont
- **Throws** no exceptions thrown
- **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

on Plot::DrawXGrid(&$ImageResource) *[line 565]*

nction Parameters:

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

ws x-grid

Drawing X grid on the plot.

- **Uses** Plot::GetImageY()
- **Uses** Plot::GetImageX()
- **Uses** Plot::GetXGridSpace()
- **Uses** Plot::ShortNumber()
- **Throws** no exceptions thrown
- **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

on Plot::DrawYGrid(&$ImageResource) *[line 630]*
nction Parameters:

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

ws y-grid
Drawing y grid on the plot.

- **Uses** Plot::GetImageY()
- **Uses** Plot::GetImageX()
- **Uses** Plot::GetYGridSpace()
- **Uses** Plot::ShortNumber()
- **Throws** no exceptions thrown
- **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

Resource function Plot::GeneratePlot([$ImageResource = null], [$ChangeSize = false]) *[line 298]*
nction Parameters:

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

ImageResource of the plot
Generates ImageResource representation of the plot.

- **Uses** Plot::$EnableGrid
- **Uses** Plot::DrawPlots()
- **Usedby** Plot::DisplayPlot()
- **Usedby** Plot::SaveAs()
- **Throws** no exceptions thrown
- **Uses** Plot::DrawGrid()
- **Uses** Plot::DrawCaption()
- **Uses** Plot::$EnableAxis
- **Uses** Plot::$BackgroundColor
- **Uses** Plot::$Height
- **Uses** Plot::$Width
- **Uses** Plot::DrawAxis()
- **Access** public

r function Plot::GetCoordinatX($x) *[line 780]*
nction Parameters:

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

nvert to coordinate space

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

- **Usedby** Plot::DrawPlots()
- **Throws** no exceptions thrown
- **Uses** Plot::$Width
- **Uses** Plot::$MinX
- **Uses** Plot::$MaxX
- **Access** private

r function Plot::GetCoordinatY($y) *[line 797]*
nction Parameters:

integer **$y** Y image coordinat to be converted.

nvert to coordinate space

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

- **Throws** no exceptions thrown
- **Uses** Plot::$MinY
- **Uses** Plot::$MaxY
- **Uses** Plot::$Height
- **Access** private

unction Plot::GetHash() *[line 260]*

nerate hash

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

- **Uses** Plot::$Width
- **Uses** Plot::$MinY
- **Uses** Plot::$MinX
- **Uses** Plot::$XGridSpace
- **Uses** Plot::$YGridSpace
- **Throws** no exceptions thrown
- **Uses** Graph::GetHash()
- **Uses** Plot::$MaxY
- **Uses** Plot::$MaxX
- **Uses** Plot::$EnableGrid
- **Uses** Plot::$EnableAxis
- **Uses** Plot::$CaptionFont
- **Uses** Plot::$Graphs
- **Uses** Plot::$GridColor
- **Uses** Plot::$Height
- **Uses** Plot::$GridFont
- **Uses** Plot::$Caption

r function Plot::GetImageX($x) *[line 814]*

nction Parameters:

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

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

- **Usedby** Plot::DrawYGrid()
- **Usedby** Plot::DrawAxis()
- **Throws** no exceptions thrown
- **Usedby** Plot::DrawXGrid()
- **Usedby** Plot::DrawPlots()
- **Uses** Plot::$MaxX
- **Uses** Plot::$MinX
- **Uses** Plot::$Width
- **Access** private

r function Plot::GetImageY($y) *[line 831]*
nction Parameters:

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

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

- **Usedby** Plot::DrawYGrid()
- **Usedby** Plot::DrawAxis()
- **Throws** no exceptions thrown
- **Usedby** Plot::DrawXGrid()
- **Usedby** Plot::DrawPlots()
- **Uses** Plot::$Height
- **Uses** Plot::$MaxY
- **Uses** Plot::$MinY
- **Access** private

r function Plot::GetXGridSpace() *[line 490]*

X grid space

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

- **Throws** no exceptions thrown
- **Usedby** Plot::DrawXGrid()
- **Uses** Plot::$XGridSpace
- **Access** private

r function Plot::GetYGridSpace() *[line 515]*

Y grid space

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

- **Throws** no exceptions thrown
- **Usedby** Plot::DrawYGrid()
- **Uses** Plot::$YGridSpace
- **Access** private

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

nction 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?

e 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.

- **Throws** no exceptions thrown
- **Uses** Plot::GeneratePlot()

- **Access** public

unction Plot::ShortNumber($Number, [$MaxLen = 7]) *[line 465]*
nction Parameters:

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

nerates short numbers
Rewrites numbers into scientific notation, with a certain maximum length.
ample: ShortNumber(501000000) == 5.01e8

- **Throws** no exceptions thrown
- **Usedby** Plot::DrawYGrid()
- **Usedby** Plot::DrawXGrid()
- **Access** private

Appendices

Appendix A - Class Trees

Package WikiPlot

che

ache

lMath

valMath

lMathStack

valMathStack

ph

raph

t

ot

LParser

MLParser

Appendix C - Source Code

Package WikiPlot

File Source for cache.class.php

mentation for this file is available at [cache.class.php]

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

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

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free
re 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
pers.
*
* @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

mentation for this file is available at CleanupCache.php

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

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

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

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

mentation for this file is available at [WikiPlot.php]

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

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

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

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

/**
* The MediaWiki extension
*
* This is the MediaWiki extension it self, everything else is just functions and liberaries for this file.
*
* @package WikiPlot
* @license http://www.gnu.org/licenses/gpl.txt GNU General Public License
* @author WikiPlot development team.
* @copyright Copyright 2006, WikiPlot development team.
*/


/**
*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"          );

/**
*Register the WikiPlot extension
*
*Makes sure the extension is called when MediaWiki is started.
*/
$wgExtensionFunctions[] = "wfWikiPlotExtension"          ;

/**
*Add hooks
*
*Adds hooks so MediaWiki will perform callback, when it hits the wikiplot tag.
*/
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 in
kiML.
*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 the
.
*If it is impossible to deserialize the value, the output object is not initialized at all. Usualy this function
othing.
*
*@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 the
.
*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 in
kiML.
*If it is impossible to deserialize the value, the output object is not initialized at all. Usualy this function
othing 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 deserialize
ters 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);
        }
    }
}

/**
*RenderWikiPlot CallBack function
*
*This is the function that handles MediaWiki callbacks, and renders the actual plot.
*
*@access private
*@param string $input The content of the wikiplot tag
*@param array $argv Hash-array of the parameters of the wikiplot tag, with parameter-name as key and parameter-value
ue.
*@param Parser $parser The parser of MediaWiki, if null parser is obtained from global variable
*@uses WikiPlotDeserializeBoolean()
*@uses WikiPlotDeserializeString()
*@uses WikiPlotDeserializeMixed()
*@uses WikiPlotDeserializeInteger()
*@uses WikiPlotDeserializeColor()
*@uses XMLParser
*@uses Plot
*@uses Graph
*@uses Cache
*@return string HTML that can be directly inserted into any website.
*/
function RenderWikiPlot($input, $argv, $parser = null)
{
    //Get parser if not given as parameter
    if (!$parser) $parser =&     $GLOBALS['wgParser'];
    /*Currently the parser*/

    //Creating instance of plot
    $Plot = new Plot();

    //Getting and deserializing parameters
    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);

    //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
PlotURL'></      a>   "     ;

    return $output;
}

?>
```

File Source for WikiPlotSettings.php

mentation for this file is available at [WikiPlotSettings.php]

```php
<?php
/**
* File used to store settings
*
* This file, is supposed to be manipulated by the user, it contains settings for WikiPlot. Primarily for the caching
onallity.
*
* @package WikiPlot
* @author 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

mentation for this file is available at [xml.class.php]

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

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

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

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

/**
* 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;
```

```php
/**
 * 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 constructer.
 *
 * @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

mentation 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($expr)-1); // strip semicolons at the end
        //===============
        // is it a variable assignment?
        if (preg_match('/^\s*([a-z]\w*)\s*=\s*(.+)$/', $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 and
ure 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,
es)) {
            $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 the
```

```
nts
x
            if (($stack = $this->   nfx($matches[3])) === false) return false; // see if it can be converted to

            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 good
            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 $expecting_op) { // are we putting an operator on the stack?
                if ($ex) { // are we expecting an operator but have a number/variable/function/opening parethesis?
                    $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[$op]
_p[$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 close a
on?
                $fnn = $matches[1]; // get the function name
                $arg_count = $stack->   pop(); // see how many arguments there were (cleverly stored on the
 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 expected)"     );
                } 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, "      .
$this->   f[$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, $matches)) { // may be func, or variable w/
it multiplication against parentheses...
                if (in_array($matches[1], $this->   fb) or array_key_exists($matches[1], $this->   f)) { // it's

                    $stack->   push($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
pace
```

```
                $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 stack,
re unbalanced
            $output[] = $op;
        }
        return $output;
    }

    // evaluate postfix notation

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

        if ($tokens == false) return false;

        $stack = new EvalMathStack;

        foreach ($tokens as $token) { // nice and easy
            // if the token is a binary operator, pop two values off the stack, do the operation, and push the
 back on
            if (in_array($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, $op2)); break;
                }
            // if the token is a unary operator, pop one value off the stack, do the operation, and push it back on
            } elseif ($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
 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-
igger("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

mentation for this file is available at [graph.plot.class.php]

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

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

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

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


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

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

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

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

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

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

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

    /**
    *Get hash
    *
    *Gets a hash of the graphs parameters. Actually is not a hashsum but just all parameter parsed as one string,
s 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 ."_"          .
>   Color[0] . "_"          . $this->   Color[1] . "_"          . $this->   Color[2] . "_"          . $this-
ableLabel;
    }

}
?>
```

File Source for plot.class.php

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

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

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

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

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free
re 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, green
ue. 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
nerated 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
nerated 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 . "_"          . $this-
idFont;
        $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-
ckgroundColor[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-
ight))
        {
            //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-
ckgroundColor[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-
idColor[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-
idColor[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
yType to png, gif or jpeg. Defaults to png, gif not recommanded. Note: this changes the current http-request
pe 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
= (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
!= (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
= (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
!= (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));
    }
}
?>
```

Index

File use to draw plots