

Chapter 9. API Reference

Last Updated Sunday, 13 May 2007

Introduction

Joomla! provides developers with an Application Programming Interface (API) through which they can develop add-on code to enhance Joomla!'s behaviour through such things as components, modules, mambots, templates and languages. This section of the Joomla! Developer Manual provides documentation for the API and consists of reference pages that each describe, in detail, a single Joomla! class, method or function.

phpDocumentor

In addition to these reference pages, the Joomla! source code itself contains special tags which are used by a program called phpDocumentor (see <http://www.phpdoc.org/> for further information) to automatically generate standard documentation. The phpDocumentor output for the current stable Joomla! release is [here](#). The API Reference pages are written by humans (the API/DB Mini-team) and are intended to complement, not replace, the phpDocumentor output. In particular API Reference pages generally contain more detailed descriptions and example code whereas the phpDocumentor output is more richly cross-referenced.

Finding an API Reference page

Since there are many, many API Reference pages it is advisable to use the filter and search capabilities of the help site to locate the class or function you require. Make use of the filter box at the top of each contents page. You can enter a class name, a function or method name, or a substring of either, in order to narrow your search. You can also perform full text searches using the standard help site search facility.

Classes

Classes are generally documented across several pages, but all pages contain the class name in their title so that using the help site filter box will capture all of them. Each class has a single class overview page and one or more class method pages which each describe a single function (method) defined in that class.

The class overview page contains a description of the class, the path and name of the file where the class is defined, and a list of the class methods available as part of the API. Note that there may be other methods defined in the class which are not considered part of the API (they are "private" methods in the language of object-orientation) and these will not be listed or documented in the API Reference but will be included in the phpDocumentor output.

Each "public" method (that is each method that is considered part of the API) has a page to itself. If the class possesses a constructor method then it too will have its own page with the title "<class> constructor". The form of these pages will be the same in all cases. Method pages include a description of the method, its syntax, and one or more examples of its use. You may notice that some methods are referred to in the form "class::method" whereas others are referred to as "class->method". This is an indication as to whether the method concerned may be called statically. If the class does not need to be instantiated before calling the method then the "class::method" form is used (for example, `mosHTML::selectList`), but if an instance of the class must be created before a class method can be called then the "class->method" form is used (for example, `JDatabase->setQuery`). This reflects actual usage in the code.

Functions

Functions that do not belong to any class are always referred to simply as "functions" and each has a page with the function name as a title. The format is essentially the same as for class method pages but they also include the path and

name of the file where the function is defined.

Syntax

Method and function pages all include a syntax specification. The syntax meta-language has been kept as simple and straightforward as possible and is standardised across all pages. In each case the function name is preceeded by a datatype keyword that indicates the nature of the datatype returned by the method/function. The argument list is given in a stylised format where each argument is preceeded by its expected datatype and optional arguments are surrounded by square brackets. Note that where more than one argument is optional the square bracket will be nested since PHP does not permit arguments to be omitted completely from the list.

The following datatype keywords are used:

- array is a PHP associative array.
- boolean is either true or false.
- float is a floating point number.
- int is an integer (a whole number which may be positive, negative or zero).
- mixed is an indication that the type expected or returned may be of more than one type. However, where a method/function normally returns a particular type (such as object or array), but returns false if an error condition occurred, then the syntax specification for the method/function will show the type normally returned rather than mixed.
- object is a PHP object. Since PHP objects are not strongly typed the description will usually contain some indication of the object type expected or returned.
- string is a string of characters.
- void is used to indicate that the method/function does not return any value at all.

Following the syntax specification, each argument is listed with a description of the nature and purpose of the argument. Note that the names of the arguments may differ from that used in the source code (and the phpDocumentor output) in cases where it is believed that the source code argument names are not sufficiently appropriate or descriptive.

History

Although it is intended that the API Reference should document the current stable Joomla! release, whatever that might be, an attempt has also been made to maintain historical information about each class, method or function. There may also be references to future versions of Joomla! that are still in development and are not yet considered stable. Consequently, when reading the API Reference you should always be aware of the version of Joomla! that you are developing for. Note in particular that there are a substantial number of changes to the API with the release of Joomla! 1.1.

The Joomla! project could be said to have begun with a split from the original Mambo project that occurred in 2005. At the time of the split the current stable release was Mambo 4.5.2.3 and this became the basis of the Joomla! 1.0 release. Versions of Mambo after 4.5.2.3 are not documented here. Versions of Mambo prior to 4.5 are also not documented here

as these would be of purely historical rather than practical value.

OrganisationThe production and maintenance of the API Reference is the responsibility of the API/DB Team which is a mini-team within the Developer Documentation Team. The API/DB Team makes use of the Joomla! Forge to manage its operations and any errors or defects in the API Reference are recorded in the tracker there. The team also uses the task manager to plan and control the creation of new API Reference pages and the maintenance of existing pages using a published workflow procedure. Announcements regarding the API Reference are made in the public Developer Documentation Forum where you are also welcome to raise any questions you may have regarding any aspect of the API Reference.