Developer’s Manual

Developers Manual

This document describes some of Moodle's design and how you can contribute.

It's a bit thin at the moment - better documentation will come eventually!

Sections in this document:

1. Moodle architecture

From a system administrator's perspective, Moodle has been designed according to the following criteria:

1. **Moodle should run on the widest variety of platforms**

   The web application platform that runs on most platforms is PHP combined with MySQL, and this is the environment that Moodle has been developed in (on Linux, Windows, and Mac OS X). Moodle also uses the ADOdb library for database abstraction, which means Moodle can use more than ten different brands of database (unfortunately, though, it can not yet *set up tables* in all these databases - more on this later).

2. **Moodle should be easy to install, learn and modify**

   Early prototypes of Moodle (1999) were built using Zope - an advanced object-oriented web application server. Unfortunately I found that although the technology was pretty cool, it had a very steep learning curve and was not very flexible in terms of system administration. The PHP scripting language, on the other hand, is very easy to get into

(especially if you've done any programming using any other scripting language). Early on I made the decision to avoid using a class-oriented design - again, to keep it simple to understand for novices. Code reuse is instead achieved by libraries of clearly-named functions and consistent layout of script files. PHP is also easy to install (binaries are available for every platform) and is widely available to the point that most web hosting services provide it as standard.

3. **It should be easy to upgrade from one version to the next**

   Moodle knows what version it is (as well as the versions of all plug-in modules) and a mechanism has been built-in so that Moodle can properly upgrade itself to new versions (for example it can rename database tables or add new fields). If using CVS in Unix for example, one can just do a "cvs update -d" and then visit the site home page to complete an upgrade.

4. **It should be modular to allow for growth**

   Moodle has a number of features that are modular, including themes, activities, interface languages, database schemas and course formats. This allows anyone to add features to the main codebase or to even distribute them separately. More on this below in the next section.

5. **It should be able to be used in conjunction with other systems**

   One thing Moodle does is keep all files for one course within a single, normal directory on the server. This would allow a system administrator to provide seamless forms of file-level access for each teacher, such as Appletalk, SMB, NFS, FTP, WebDAV and so on. The authentication modules allow Moodle to use LDAP, IMAP, POP3, NNTP and other databases as sources for user information. Otherwise, there is work yet to do. Features planned for Moodle in future versions include: import and export of Moodle data using XML-based formats (including IMS and SCORM); and increased use of style sheets for interface formatting (so that it can be integrated visually into other web sites).

2. How you can contribute

As mentioned above, Moodle has a number of features that are modular. Even if you are not a programmer there are things you can change or help with.

Learning Activities

These are by far the most important modules, and reside in the 'mod' directory. There are seven default modules: assignment, choice, forum, quiz, resource, and survey. Each module is in a separate subdirectory and consists of the following mandatory elements (plus extra scripts unique to each module):

- mod.html: a form to set up or update an instance of this module
- version.php: defines some meta-info and provides upgrading code
- icon.gif: a 16x16 icon for the module
- db/: SQL dumps of all the required db tables and data (for each database type)
- index.php: a page to list all instances in a course
- view.php: a page to view a particular instance
- lib.php: any/all functions defined by the module should be in here. If the modulename if called widget, then the required functions include:
    - widget_add_instance() - code to add a new instance of widget
    - widget_update_instance() - code to update an existing instance
    - widget_delete_instance() - code to delete an instance
    - widget_user_outline() - given an instance, return a summary of a user's contribution
    - widget_user_complete() - given an instance, print details of a user's contribution
    - To avoid possible conflict, any module functions should be named starting with widget_ and any constants you define should start with WIDGET_
- Lastly, each module will have some language files that contain strings for that module. See below.

The easiest way to start a new learning activity module is to use the template in **mod/newmodule_template.zip.** Unzip it and follow the README inside.

You might also like to post first in the Activities modules forum on Using Moodle.

Themes

Themes (or skins) define the look of a site. A number of simple themes are provided in the main distribution, but you may want to create your own theme with your own colours, logo, styles and graphics.

Each theme is in a subdirectory of the "theme" directory, and contains at least the following files:

- **config.php**: defines the theme colours used throughout the site
- **styles.php**: the style sheet, containing CSS definitions for standard HTML elements as well as many Moodle elements.
- **header.html**: Included at the top of each page. This is what you need to edit to add a logo at the top of pages, for example.
- **footer.html**: Included at the bottom of each page.

To create your own themes for current versions of Moodle:

1. Copy one of the existing theme folders to one with a new name. I recommend starting with one of the standard themes.
2. Edit config.php and insert your own colours.
3. Edit styles.php and change your CSS styles.
4. Edit header.html and footer.html to add new logos, or change the layout.

Note that all these steps are optional - you can make a radically different look to your site simply by editing the colours in config.php

Note also that Moodle upgrades *may* break themes slightly, so check the release notes carefully if you are using a custom theme.

In particular, Moodle 2.0 will have a completely new display system, probably based on XSL transformations of XML output from Moodle. It is likely that the themes for this will be a completely different format, but the advantage will be a much higher possible degree of customisation (including moving elements around the page).

More discussion about this in the Themes forum on Using Moodle. If you create a nice theme that you think others might want to use,

please post your zip file on the themes forum!

Languages

Moodle has been designed for internationalisation. Each 'string' or 'page' of text that is displayed as part of the interface is drawn from a set of language files. Each language is a subdirectory of the directory 'lang'. The structure of the lang directory is as follows:

lang/en - directory containing all files for one language (eg English)

- moodle.php - strings for main interface
- assignment.php - strings for assignment module
- choice.php - strings for choice module
- forum.php - strings for forum module
- quiz.php - strings for quiz module
- resource.php - strings for resource module
- survey.php - strings for survey module
- .... plus other modules if any.

A string is called from these files using the *get_string()* or *print_string()* functions. Each string supports variable substitution, to support variable ordering in different languages.

eg $strdueby = get_string("assignmentdueby", "assignment", userdate($date));

If a string doesn't exist in a particular language, then the equivalent in English will automatically be used instead.

lang/en/help - contains whole help pages (for popup context-sensitive help)

Main help pages are situated here, while help pages specific to each module are located in subdirectories with the module's name.

You can insert a helpbutton in a page with the helpbutton function.

eg helpbutton("text", "Click here for help about text");

and for modules:

helpbutton("forumtypes", "Forum types", "forum");

Note that you can edit languages online, using the administration web tools under "Check this language". This makes it easy to not to only create new languages but to refine existing ones. If you are starting a new language, please contact me, Martin Dougiamas.

You might also like to post in the Languages forum on Using Moodle.

If you are maintaining a language an ongoing basis, I can give you CVS write access to the Moodle source code so that you can directly maintain the files.

Database Schemas

Given a working database with defined tables, the intentionally simple SQL used in Moodle should work fine with a wide variety of database brands.

A problem exists with **automatically creating** new tables in a database, which is what Moodle tries to do upon initial installation. Because every database is very different, there doesn't yet exist any way to do this in a platform-independent way. To support this automation in each database, schemas can be created that list the required SQL to create Moodle tables in a particular database. These are files in **lib/db** and inside the **db** subdirectory of each module.

Currently, only MySQL and PostgreSQL are fully supported in this way (no-one wrote the schemas for other brands).

Moodle 1.2 will use a new method of database-independent XML schemas that will make all this unnecessary.

Course Formats

Moodle currently supports three different course formats: weekly, topics and social.

These are a little more connected to the rest of the code (and hence, less "pluggable") but it is still quite easy to add new ones.

If you have any ideas for different formats that you need or would like to see, get in touch with me and I'll do my absolute best to have

them available in future releases.

Documentation and articles

If you feel like writing a tutorial, an article, an academic paper or anything else about Moodle, please do!

Put it on the web and make sure you include links to http://moodle.org/

Participating in the bug tracker

Finally, I would like to invite you to register on the "bug tracker" at http://moodle.org/bugs so you can file any bugs that you find and perhaps participate in discussing and fixing them.

"Bugs" not only includes software bugs with current versions of Moodle, but also new ideas, feature requests and even constructive criticism of existing features. The beauty of open source is that anyone can participate in some way and help to create a better product for all of us to enjoy. In this project, your input is very welcome!

Thanks for using Moodle!

Martin Dougiamas, Lead Developer

Moodle Documentation

Version: $Id: developer.html,v 1.11.2.2 2005/07/07 15:46:23 moodler Exp $