ISSN:0975-9646
Madhupriya R et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4353-4356

Customizing Moodle to Facilitate Integration with Other Platforms

Madhupriya R
Department of Computer Science and Engineering,
Manipal Institute of Technology,
Manipal, Karnataka, India

***Abstract* — Moodle is a highly versatile learning management system with a wide variety of tools ranging from simple grade reports to complex assessments. In spite of its resourcefulness, other platforms are still in use, especially due to the scalability constraints of Moodle. In such scenarios, it highly essential to integrate Moodle with other platforms to make the best use of features present in all components. This can be done by customizing Moodle and feeding it with data from other platforms to view combined results. Data from other systems are transferred to Moodle's local database using secure connections. A local plugin is developed in Moodle to fetch the inserted data in addition to the already existing data and create aggregate reports. Various capabilities can be defined in the plugin to impose view and access constraints for various roles.**

Keywords—Moodle, plugin, LMS

I. INTRODUCTION

Moodle is a highly robust, open source, learning management system (LMS) that has eased the lives of millions of course instructors and learners across the globe [1]. The tools provided by Moodle are highly versatile and offer solutions to a wide range of difficulties faced in conventional classroom environments. Moodle has been constantly evolving with a plethora of features such as personalized dashboards, multilingual capabilities, regular security updates and open standards. It is highly secure with multiple authentication and enrollment options available. Also, it has a well formed community to resolve all technical concerns and provide constant support to its users.

In spite of its resourcefulness, Moodle has failed to replace some of the existing LMSs. This is partly due to the scalability constraints of Moodle, when supporting a very large number of users [2]. Its user interface and database are organized very inefficiently and consumes a lot of space. Also, the tables in its database are not well connected and hence simple tasks require complex queries that join multiple tables, thereby causing a huge overhead of space and time. It is difficult to incorporate Moodle in a distributed management model wherein a single institution has multiple schools and departments with smaller groups among them. Student management and human resource systems are kept separately in these institutions to avoid space overhead and ease maintenance. It is difficult to integrate Moodle with these systems as Moodle has its own storage techniques and expects data to be identified with its own specific attributes making it inflexible.

In contemporary times, there are numerous institutions which have adopted the methodology of flipped classrooms. The conventional classroom environment is slowly losing its charm in today's fast paced world. An enhancement in the quality and capabilities of LMSs is the major cause for this growth. Apart from the LMSs, there are numerous course management systems such as Open edX which offer Massive Open Online Courses (MOOC) to millions of users across the globe [3]. These systems are well suited to support several concurrent users accessing the same resources without degrading the user experience. Course management systems are a major requirement to support this new age blended learning practice.

In such a scenario where both course management and learning management systems are required to be used together, it is highly essential to integrate the two into a single unit. It can be argued that either of the two can be used to perform the tasks of the other, but this will just impose a high overhead in the implementation. It is more desirable to keep the systems as distinct entities and use them only for the capabilities they specialize in. Integration is necessary to make use of the best features provided by these systems and summarize the overall system at a single point. There are scenarios where a single institute supports both of these and manage student and instructor data on both the platforms. It is a cumbersome task for an administrator to manually fetch data from both the ends and then create a separate platform to view combined reports.

To integrate these entities, this paper suggests a method of creating a local plugin at the Moodle end. Moodle is highly developer friendly and customizable and therefore a plugin can be easily developed. Data from other systems can be transferred to Moodle's local data store and the plugin can be customized to fetch the required data from it. Also, Moodle is more preferable when it comes to generating reports and statistics and it has specialized tools for doing so.

Various permissions can be set in the plugin so that users with specified roles are only allowed to view data specific to their role. The plugin can be added to the various inbuilt sections that are already present in Moodle and therefore make it user friendly without altering the existing look and feel of Moodle. The overall goal of this integration is to improve the quality of learning and make students and instructors well equipped with the best possible tools that are available today.

II. MOODLE ARCHITECTURE

Overview- Moodle is an open source software which can be easily installed by following a step by step procedure [4]. An Apache server is required to be installed along with additional PHP packages that might not be present in the host computer. Moodle's code is written in PHP. The code is located in the /var/www/html/moodle/ directory, var being the web root. It must be kept in mind that the owner of the moodle directory should be the web server so that it can write to it and make changes. Also, a local repository with the name of moodledata is set up in the web root folder with the required permissions. This is the default location where all uploaded and auto-generated files are stored. Various other parameters such as login details are updated during the web installation of Moodle to suit custom needs. All configuration parameters, network addresses and global variables are located in the config.php file which is present in the Moodle directory. The parameters present in this file can be used throughout Moodle and hence maintain uniformity.

Database- Moodle requires a database which can be MySQL, PostgreSQL, Microsoft SQL Server or Oracle. During its installation, a prefix is supplied by the installer which precedes all table names used in Moodle for interoperability with other databases. The default storage engine is made innodb which ensures high reliability and performance to support high concurrency. The install.xml file present in the db folder of each plugin defines the database structure of Moodle. It is recommended that file shouldn't be modified manually as it is auto-generated. It can be modified from the XMLDB Editor present in the site administration menu. The links between the tables in Moodle are not very apparent and complex. Consequently, elaborate queries are required to fetch simple data items.

Roles- Moodle allows the site administrator to define various roles and capabilities. The most common roles are a student, teacher, manager, guest, non-editing teacher and course creator. A role has a list of permissions, which can be modified. Roles are instrumental in maintaining the hierarchy and organizational structure of the institution that is using Moodle. Roles can be managed, assigned, overridden and switched from the administration menu. For example, a course editing teacher can be allowed to switch to the role of a student to see what the course content looks like for a student. Also, while assigning roles, the context can also be set so that the capabilities defined in the role are applicable to a particular context such as course or category.

Moodle homepage- The main page of Moodle is divided into blocks and sections. The navigation block appears at the top left which is customized according to the profile of the user. It shows the profile of the user along with all course content for which the user is enrolled. By clicking on a particular course, various sections appear in the middle of the page with is generally organized as a weekly list of sections. The settings block is present below the navigation block which also contains the course administration menu to view the grades. The site administrator performs all functions from the list of sections present in the block which include site administration, role switching and editing user profiles. The right hand corner contains optional, informative blocks such as a calendar, latest news and recent activity. The organization and content of all these entities can be modified by turning the editing on from the top right corner.

III. PLUGIN DEVELOPMENT

A local plugin needs to be developed in Moodle when the requirements are not standard and the existing plugins are insufficient. Plugins provide a way for customization and also, to fetch data from newly created tables. They can also be used to communicate with external systems and define new capabilities. These plugins go into the /local directory of Moodle and can be added to various pre-existing blocks of Moodle. The plugin can also be contributed to the open source Moodle community, thereby being beneficial to others facing similar difficulties.

To develop a plugin, a directory structure according to the Moodle documentation needs to be created [5]. Various mandatory files need to be added to the plugin folder. Also, required permissions must be set for the plugin to be accessible by the web server. The directory structure is not initially created in the target folder which is the /local directory. It is initially developed elsewhere. Later, while installing it from the UI of Moodle it is zipped and added to the /local directory.

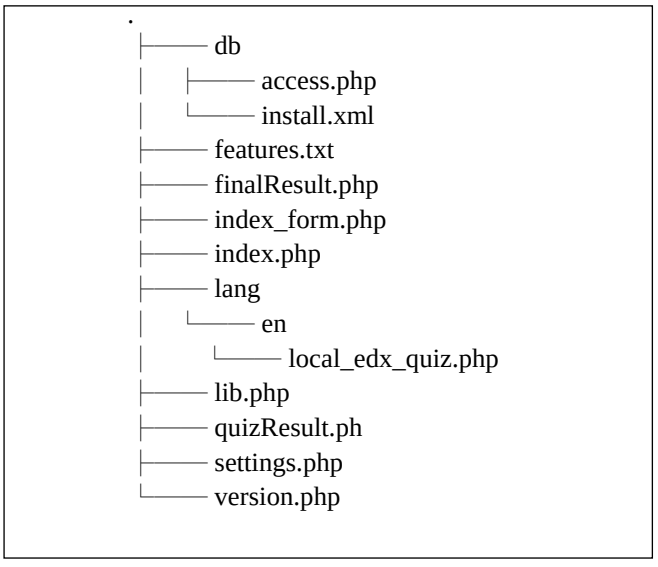

Fig. 1 Directory structure of plugin

The access.php file is used to define various capabilities in the plugin. These are the various view level constraints in accordance with the roles. The install.xml file contains the schema structure that the plugin will use. As mentioned previously, it must not be modified manually and is done by the XMLDB Editor. The lib.php file comprises of all navigational hooks and visibility modes of the plugin. This

file is automatically added by the config.php file that is present by default. The settings.php file is used to define configuration options. New items are added to the admin tree block from here. It is also contains the version details which are highly essential for Moodle to update its database and invalidate all caches. The index.php file is the first page that the plugin redirects to and it can be customized according to the requirements. Various other files can be added to the plugin that can open from links present in the index file. All these files can use the global parameters present in Moodle so that all changes can be made at a single point. Moodle has various HTML and PHP functions that developers can use so that the look and feel of Moodle is not altered. The developed plugin can be compressed and installed from the UI. Changes in the database can be done from the XMLDB Editor. The database must be reloaded and upgraded after the changes are performed. The plugin can be added to various blocks such as the course administration block by modifying the lib.php file.

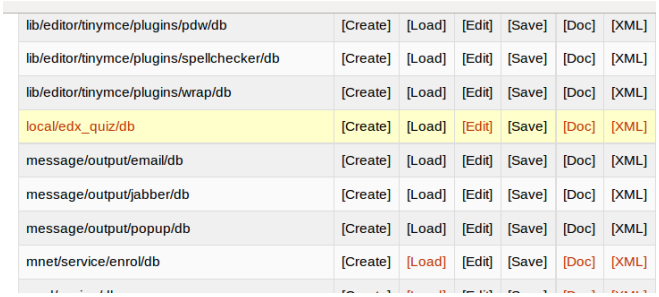

Fig. 2 New plugin added to XMLDB Editor

IV. DATA TRANSFER FROM OTHER PLATFORMS

Data can be easily transferred to the local data store of Moodle from other sources. Various file transfer techniques such as ftp, scp and shp are suggested. Structured data is preferred so that there is lesser overhead at the destination end. All the data conversion should be done before transferring the data at the source itself to ensure clean and concise transfer by using techniques such as HDFS with map reduce. The structured data, usually in the form of csv files can be dumped into new tables in the Moodle database using straightforward queries. The tables must be dynamic so that varying sizes of input can be constantly dumped without any major rework. The transfer mechanism can also be made automatic with web services that have push or pull techniques. The framework can be customized with a rules engine with a stage and forward mechanism. Also, utilities such as cron can be considered viable to schedule jobs. The real challenge here is to create queries that link the fresh data with that which is already present in Moodle.

V. EXPERIMENT

An integration prototype was tried out between a customized version of open edX and Moodle. Data from the former platform was already available in csv format after transforming it as data conversion is immaterial to this experiment. The files were transferred using scp to the database directory of Moodle. The data contained quiz details along with the names and email ids of students. The scenario was that a single student is registered with the same email-id on open-edX and Moodle. There were quizzes conducted on both platforms and hence there was an urgent need for the teacher in-charge to view aggregate results at a single point. The common identifier is the email-id which is required to be the same on both the platforms. Two new tables mdl_edX_quiz and mdl_edX_quiz_results were created in the Moodle database. The former was used to store quiz details while the latter storing the results obtained in these quizzes along with the quiz identifier from the first table.

TABLE I
DESCRIPTION OF MDL_EDX_QUIZ

Field	Type	Null	Key	Default	Extra
Quizid	Integer(11)	NO	PRI	NULL	auto_increment
Quizname	varchar(30)	NO		NULL	
Courseid	varchar(20)	NO		NULL	
Maxmarks	Integer(11)	NO		NULL	

TABLE III
DESCRIPTION OF MDL_EDX_QUIZ_RESULTS

Field	Type	Null	Key	Default	Extra
Emailed	varchar(255)	NO		NULL	
Username	varchar(100)	NO		NULL	
Quizid	Integer(11)	NO	Foreign	NULL	
Courseid	varchar(20)	NO		NULL	
Result	Integer(11)	NO		NULL	

Data from the csv file was inserted into these two tables using MySQL queries. Various permissions were granted on the database to make the transfer possible.

A local plugin by the name edx_course was developed in Moodle and added to the course administration block. Permissions were set in it so that teachers only view results of students for the courses they are undertaking while administrators can view overall results. Students can view their own results in all the quizzes in an aggregated fashion. The primary purpose of the plugin is to fetch data from the edX and Moodle tables and combine those using dynamic queries. The tables used in Moodle were mdl_user, mdl_quiz, mdl_quiz_attempts, mdl_user_enrolments and mdl_role_assignments. Some of the aggregate functions in MySQL had to be used to convert data from rows to columns against the common email-id.

The index page of the plugin had links to two new pages-quizResult.php and finalResult.php. The quiz result page had a dropdown containing the names of all the edX quizzes. The dropdown is inherited from the inbuilt

Moodle forms class. Clicking and submitting any of the names displayed the quiz attempts of all the students who participated in the quiz along with the grades. This is useful when Moodle is just used to view reports while the actual activities are taking place at another point.

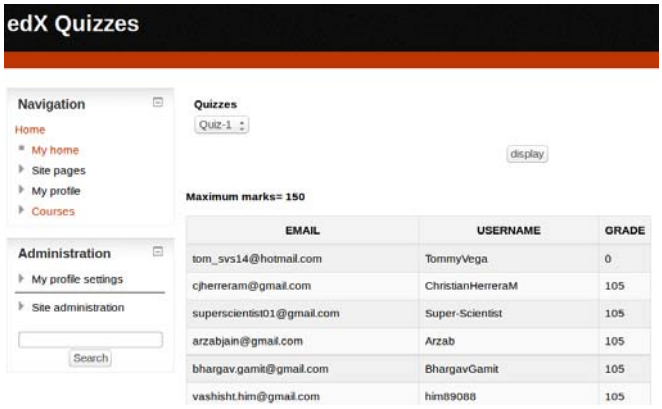

Fig. 3 Quiz Result Page

Creation of the other page-finalResult.php was the most challenging task. This page displayed the edX grades alongside Moodle grades for the same email-id. This was used to assess the aggregate performance of students and give the teacher an overall picture of the student's progress. Moodle contains a rich set of inbuilt APIs such as the html_table element which lets you add tables that retain the look and feel of Moodle.

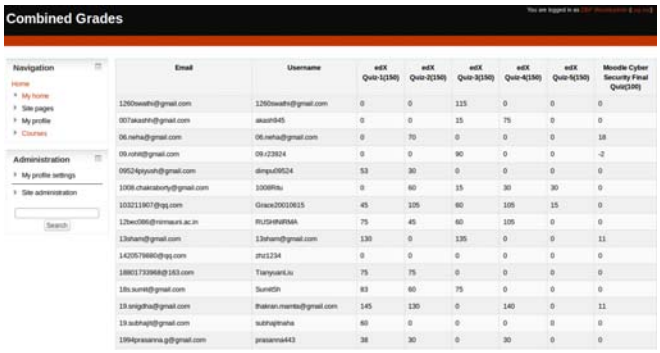

Fig. 4 Final Result Page

VI. CHALLENGES

The primary challenge of integration is the scalability constraint of Moodle [2]. Moodle is a heavy weight platform and since the databases occupy a very large volume of space, proper care is to be taken while writing queries. Code inside a loop and join queries must be strictly monitored. Also, caution must be taken while using external function calls as the function state causes a stack overhead. As a rule of thumb, every page must use only a fixed set of database queries to avoid overburdening it. Also, efficient hardware is required to cater to the challenging performance requirements. Institutions must be well convinced to support integration and data from other platforms must be easily available. If a lot of data at other ends is unstructured, a lot of effort will have to be spent on development. Also, generating dynamic queries to handle varying data sizes from external sources is a key concern and must be done efficiently.

VII. CONCLUSUSION AND FUTURE WORK

The primary objective of this integration is to reduce the manual effort undertaken by administrators of institutions wherein multiple LMSs are in use. Integration is instrumental in the total assessment of students, thereby improving the quality of learning. A complete analysis of the strengths and weaknesses of learners can be understood and worked upon. The prototype that has been created has laid down the foundation for total integration and has immense scope in future. Data transfer can be automated by using web services that keep transferring new data at fresh intervals and synchronizes it. An export facility can be added to the plugin for administrators to use the data obtained from the plugin for future needs. Also, some sort of shared memory can be created with bidirectional transfer of data to create a fully integrated system with minimum transfer latency.

ACKNOWLEDGMENT

I express my sincere gratitude towards Professor Deepak B. Phatak and the Indian Institute of Technology, Bombay for providing me with the platform and resources without which this project would have been impossible.

REFERENCES

[1] Suleiman Alhaji Ahmad, Umar Bawa Chinade, Abdu Muhammad Gambaki, Shehu Ibrahim, and Nasiru Ahmed Ala, "The need for Moodle as a Learning Management System in Nigerian Universities: Digesting University Utara Malaysia Learning zone as a case study," *Academic Research International*, vol. 2, no. 3, pp. 446, May 2012.

[2] Catalyst IT Limited. (2015, Sep 3). *Technical Evaluation of selected Learning Management Systems* [Online]. Available: https://moodle.org/pluginfile.php/1540/mod_folder/content/0/Comparativas/LMS_Technical_E_ion_-_May04.pdf?forcedownload=1

[3] Daradoumis T, Bassi, R, Xhafa, F and Caballe, S, "A Review on Massive E-Learning (MOOC) Design, Delivery and Assessment," *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne*, 2013, pp. 208 – 213

[4] Howtoforge. (2015, Sep 3). *How to install Moodle on Ubuntu 14.04* [Online]. Available: https://www.howtoforge.com/how-to-install-moodle-on-ubuntu-14.04

[5] Moodle. (2015, Sep 3). *MoodleDocs* [Online]. Available: https://docs.moodle.org/