

DEVELOPMENT OF ONLINE COURSE SYSTEM AND AN OPEN ACCESS ONLINE REPOSITORY

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology

In

Electronics and Communication Engineering

By

ANSHUL BARANWAL

and

BHUPENDRA PAYAL

Under the Guidance of

Prof. S.K.Patra



**Department of Electronics & Communication Engineering
National Institute of Technology
Rourkela**

2009



**National Institute of Technology
Rourkela**

CERTIFICATE

This is to certify that the thesis entitled, “Development of online course system and an open access online repository” submitted by Sri Anshul Baranwal and Sri Bhupendra Payal in partial fulfillments for the requirements for the award of Bachelor of Technology Degree in Electronics & Communication Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Prof. S.K. Patra
Dept. of Electronics & Instrumentation Engg
National Institute of Technology
Rourkela -769008

Date:

Acknowledgement

We avail this opportunity to extend our hearty indebtedness to our guide Prof. S.K. Patra, Electronics and Communication Engineering Department, for his valuable guidance, constant encouragement and kind help at different stages for the execution of this dissertation work. We would like to extend our gratitude to the staff at the Computer Centre for allowing us to carry out our project work with the help of the Computer Centre.

We also express our sincere gratitude to Mr. Madhan Muthu, Assistant Librarian, Biju Patnaik Central Library for providing us all kinds of help and resources for the successful completion of this project.

It is our pleasure to refer Microsoft Word exclusive of which the compilation of this report would have been impossible. An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in reference section. We acknowledge our indebtedness to all of them. Last but not the least, our sincere thanks to all of our friends who have patiently extended all sorts of help for accomplishing this undertaking.

Anshul Baranwal
(10509024)

and

Bhupendra Payal

Abstract

This Project was divided in to two phases: the first phase comprising of development of an online course system for the institute with the help of moodle. Moodle(modular object oriented dynamic learning environment) is an open source software package for producing internet-based courses and web sites. It's an ongoing development project designed to support a social Constructionist framework of education. Moodle is provided freely as Open Source software (under the GNU Public License). Basically this means Moodle is copyrighted, but that we have additional freedoms of improvising the source code.

Moodle will run on any computer that can run PHP and can support many types of database (particularly [MySQL](#)). Moodle takes support from a number of web servers Apache, Tomcat etc. However Apache is a preferred server. The Apache HTTP Server enables web pages to be published on the internet (or an intranet or even on a single test machine). The PHP scripting language, in which Moodle is developed, is tightly integrated with the Apache HTTP Server. After the installation process we straight away come on to the course development in Moodle. Moodle has a wide horizon of applications and modules which would make learning just as much pleasure nevertheless with advantage of editing the code as per your choice.

The 2nd Phase of the project was that of deployment of an open access online repository system using E-prints. **EPrints** is an open source software package for building open access repositories that are compliant with the Open Archives Initiative Protocol for Metadata Harvesting. It shares many of the features commonly seen in Document Management systems, but is primarily used for institutional repositories and scientific journals. EPrints has been developed at the University of Southampton School of Electronics and Computer Science and released under a GPL license.

EPrints is a Web and command-line application based on the LAMP architecture. Configuring an EPrints repository involves modifying configuration files written in Perl or XML. Web based configuration tools are in development. The appearance of a repository is controlled by HTML templates, stylesheets and inline images.

Eprints involve the OAI-PMH protocol i.e. open archives initiative protocol for metadata harvesting. The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. The Open Archives Initiative develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. OAI has its roots in the open access and institutional repository movements.

Contents

Abstract

Page No.

PHASE-I: Developing an online course system

Chapter-1

General Introduction of Moodle 8

Chapter-2

2.1 Philosophy 9

2.2 Features 10

Chapter-3

3.1 Installation of Moodle 15

3.2 Improvised method adopted for installation 22

Chapter-4

4.1 Course settings 24

4.2 General and administrative settings 26

4.3 Quizzes and other course related applications 28

Chapter-5

5.1 Moodle Architecture 29

5.2 How can we contribute 30

5.3 Moodle coding guidelines 33

5.4 Coding Style 34

<i>5.5 Security Issues</i>	37
----------------------------	----

Chapter-6

<i>6.1 Installation of Moodle on Institute server</i>	38
---	----

<i>6.2 Developing a new theme</i>	40
-----------------------------------	----

<i>6.3 Adding certain useful additional applications</i>	42
--	----

<i>6.4 Preparation of Courses</i>	44
-----------------------------------	----

PHASE-II: Deploying an open access online repository

Chapter-1

<i>1.1 Introduction to E-prints</i>	46
-------------------------------------	----

<i>1.2 Technology</i>	46
-----------------------	----

Chapter-2

<i>2.1 Installation of E-prints through binary</i>	47
--	----

<i>2.2 Installation by the source</i>	48
---------------------------------------	----

<i>2.3 Getting started</i>	48
----------------------------	----

<i>2.4 Maintenance</i>	51
------------------------	----

Chapter-3

<i>3.1 E-prints directory structure</i>	52
---	----

<i>3.2 Metadata</i>	52
---------------------	----

<i>3.3 Repository Configuration</i>	53
-------------------------------------	----

Chapter-4

<i>4.1 OAI-PMH(open archives initiative protocol for metadata harvesting)</i>	54
---	----

<i>4.2 Protocol Details</i>	56
-----------------------------	----

<i>4.3 Data providers</i>	59
---------------------------	----

<i>4.4 Service Providers</i>	60
------------------------------	----

<i>4.5 SWORD Protocol</i>	62
---------------------------	----

Chapter-5

5.1 Write in a Plug-in 63

5.2 Translation 65

5.3 Populating the database of ethesis at NITRKL 66

5.4 Developing a theme 66

Conclusion 68

References 69

PHASE -I

Developing an Online Course Website Using



Currently running as

<http://moodle.nitrkl.ac.in>

Introduction:

moodle is a software package for producing internet-based courses and web sites. It's an ongoing development project designed to support a social Constructionist framework of education. The word Moodle was originally an acronym for **Modular Object-Oriented Dynamic Learning Environment**, which is mostly useful to programmers and education theorists. It's also a verb that describes the process of lazily meandering through something, doing things as it occurs to you to do them, an enjoyable tinkering that often leads to insight and creativity. As such it applies both to the way Moodle was developed, and to the way a student or teacher might approach studying or teaching an online course.

Moodle is provided freely as Open Source software (under the GNU Public License). Basically this means Moodle is copyrighted, but that we have additional freedoms of improvising the source code. You are allowed to copy, use and modify Moodle provided that you agree to: **provide the source to others; not modify or remove the original license and copyrights, and apply this same license to any derivative work.** Read the license for full details and please contact the copyright holder directly if you have any questions.

Moodle will run on any computer that can run PHP and can support many types of database (particularly [MySQL](#)). Moodle takes support from a number of web servers Apache, Tomcat etc. However Apache is a preferred server. The Apache HTTP Server enables web pages to be published on the internet (or an intranet or even on a single test machine). The [PHP](#) scripting language, in which Moodle is developed, is tightly integrated with the Apache HTTP Server. A web server is a required component of a Moodle installation. There are two distinct developments of the Apache HTTP Server, version 1 and version 2. Although version 2 should have replaced version 1, the architectures are so different that they both continue to be supported. Either are fully supported by PHP and Moodle. The Moodle page is written to try to make this process as simple as possible for different platforms:

- [Hosting Service](#)
- [Mac OS X](#)
- [Redhat Linux](#)
- [Windows](#)

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. 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.

2.1 Philosophy:

The design and development of Moodle is guided by a particular philosophy of learning, a way of thinking that we may see referred to in shorthand as "*social constructionist pedagogy*". (Some of you scientists may already be thinking "soft education mumbo jumbo" and reaching for your mouse, but please read on - this is useful for every subject area!). Its page tries to explain in simple terms what that phrase means by unpacking **four main concepts** behind it. Note that each of these is summarizing one view of an immense amount of diverse research so these definitions may seem thin if you have read about these before.

1. Constructivism:

This point of view maintains that people actively **construct** new knowledge as they interact with their environment. Everything we read, see, hear, feel, and touch is tested against our prior knowledge and if it is viable within our mental world, may form new knowledge we carry with us. Knowledge is strengthened if we can use it successfully in your wider environment. We are not just a memory bank passively absorbing information, nor can knowledge be "transmitted" just by reading something or listening to someone.

2. Constructionism

Constructionism asserts that learning is particularly effective when constructing something for others to experience. This can be anything from a spoken sentence or an internet posting, to more complex artifacts like a painting, a house or a software package.

For example, you might read this page several times and still forget it by tomorrow - but if you were to try and explain these ideas to someone else in your own words, or produce a slideshow that explained these concepts, then I can guarantee you'd have a better understanding that is more integrated into your own ideas. This is why people take notes during lectures, even if they never read the notes again.

3. Social Constructivism

This extends the above ideas into a social group constructing things for one another, collaboratively creating a small culture of shared artifacts with shared meanings. When one is immersed within a culture like this, one is learning all the time about how to be a part of that culture, on many levels.

A very simple example is an object like a cup. The object can be used for many things, but its shape does suggest some "knowledge" about carrying liquids. A more complex example is an online course - not only do the "shapes" of the software tools indicate certain things about the

way online courses should work, but the activities and texts produced within the group as a whole will help shape how each person behaves within that group.

4. Connected and Separate

This idea looks deeper into the motivations of individuals within a discussion. **Separate** behaviour is when someone tries to remain 'objective' and 'factual', and tends to defend their own ideas using logic to find holes in their opponent's ideas. **Connected** behaviour is a more empathic approach that accepts subjectivity, trying to listen and ask questions in an effort to understand the other point of view. **Constructed** behaviour is when a person is sensitive to both of these approaches and is able to choose either of them as appropriate to the current situation.

In general, a healthy amount of connected behaviour within a learning community is a very powerful stimulant for learning, not only bringing people closer together but promoting deeper reflection and re-examination of their existing beliefs.

2.2 Features:

Moodle is an active and evolving product. This page lists just some of the many features it contains:

Overall design

- Promotes a social constructionist pedagogy (collaboration, activities, critical reflection, etc)
- Suitable for 100% online classes as well as supplementing face-to-face learning
- Simple, lightweight, efficient, compatible, low-tech browser interface
- Easy to install on almost any platform that supports PHP. Requires only one database (and can share it).
- Full database abstraction supports all major brands of database (except for initial table definition)
- Course listing shows descriptions for every course on the server, including accessibility to guests.
- Courses can be categorized and searched - one Moodle site can support thousands of courses
- Emphasis on strong security throughout. Forms are all checked, data validated, cookies encrypted etc
- Most text entry areas (resources, forum postings etc) can be edited using an embedded WYSIWYG HTML editor

Site management

- Site is managed by an admin user, defined during setup
- Plug-in "themes" allow the admin to customize the site colours, fonts, layout etc to suit local needs

- Plug-in activity modules can be added to existing Moodle installations
- Plug-in language packs allow full localization to any language. These can be edited using a built-in web-based editor. Currently there are language packs for over [43 languages](#).
- The code is clearly-written PHP under a GPL license - easy to modify to suit your needs

User management

- Goals are to reduce admin involvement to a minimum, while retaining high security
- Supports a range of authentication mechanisms through plug-in authentication modules, allowing easy integration with existing systems.
- Standard email method: students can create their own login accounts. Email addresses are verified by confirmation.
- LDAP method: account logins can be checked against an LDAP server. Admin can specify which fields to use.
- IMAP, POP3, NNTP: account logins are checked against a mail or news server. SSL, certificates and TLS are supported.
- External database: any database containing at least two fields can be used as an external authentication source.
- Each person requires only one account for the whole server - each account can have different access
- An admin account controls the creation of courses and creates teachers by assigning users to courses
- A course creator account is only allowed to create courses and teach in them
- Teachers may have editing privileges removed so that they can't modify the course (eg: for part-time tutors)
- Security - teachers can add an "enrolment key" to their courses to keep out non-students. They can give out this key face-to-face or via personal email etc
- Teachers can enroll students manually if desired
- Teachers can un-enroll students manually if desired, otherwise they are automatically un-enrolled after a certain period of inactivity (set by the admin)
- Students are encouraged to build an online profile including photos, description. Email addresses can be protected from display if required.
- Every user can specify their own time zone, and every date in Moodle is translated to that time zone (eg posting dates, assignment due dates etc)
- Every user can choose the language used for the Moodle interface (English, French, German, Spanish, Portuguese etc)

Course management

- A full teacher has full control over all settings for a course, including restricting other teachers
- Choice of course formats such as by week, by topic or a discussion-focussed social format
- Flexible array of course activities - Forums, Quizzes, Resources, Choices, Surveys, Assignments, Chats, Workshops

- Recent changes to the course since the last login can be displayed on the course home page - helps give sense of community
- Most text entry areas (resources, forum postings etc) can be edited using an embedded WYSIWYG HTML editor
- All grades for Forums, Quizzes and Assignments can be viewed on one page (and downloaded as a spreadsheet file)
- Full user logging and tracking - activity reports for each student are available with graphs and details about each module (last access, number of times read) as well as a detailed "story" of each student's involvement including postings etc on one page.
- Mail integration - copies of forum posts, teacher feedback etc can be mailed in HTML or plain text.
- Custom scales - teachers can define their own scales to be used for grading forums and assignments
- Courses can be packaged as a single zip file using the Backup function. These can be restored on any Moodle server.

Assignment Module

- Assignments can be specified with a due date and a maximum grade.
- Students can upload their assignments (any file format) to the server - they are date-stamped.
- Late assignments are allowed, but the amount of lateness is shown clearly to the teacher
- For each particular assignment, the whole class can be assessed (grade and comment) on one page in one form.
- Teacher feedback is appended to the assignment page for each student, and notification is mailed out.
- The teacher can choose to allow resubmission of assignments after grading (for regrading)

Chat Module

- Allows smooth, synchronous text interaction
- Includes profile pictures in the chat window
- Supports URLs, smilies, embedded HTML, images etc
- All sessions are logged for later viewing, and these can also be made available to students

Choice Module

- Like a poll. Can either be used to vote on something, or to get feedback from every student (eg research consent)
- Teacher sees intuitive table view of who chose what
- Students can optionally be allowed to see an up-to-date graph of results

Forum Module

- Different types of forums are available, such as teacher-only, course news, open-to-all, and one-thread-per-user.
- Discussions can be viewed nested, flat or threaded, oldest or newest first.
- Individual forums can be subscribed to by each person so that copies are forwarded via email, or the teacher can force subscription for all
- The teacher can choose not to allow replies (eg for an announcements-only forum)
- Discussion threads can be easily moved between forums by the teacher
- Attached images are shown inline

Quiz Module

- Teachers can define a database of questions for re-use in different quizzes
- Questions can be stored in categories for easy access, and these categories can be "published" to make them accessible from any course on the site.
- Quizzes are automatically graded, and can be re-graded if questions are modified
- Quizzes can have a limited time window outside of which they are not available
- At the teacher's option, quizzes can be attempted multiple times, and can show feedback and/or correct answers
- Quiz questions and quiz answers can be shuffled (randomized) to reduce cheating
- Questions allow HTML and images
- Questions can be imported from external text files
- Attempts can be cumulative, if desired, and finished over several sessions
- Multiple-choice questions supporting single or multiple answers
- Short Answer questions (words or phrases), Random questions.
- True-False questions, Matching questions, Numerical questions (with allowable ranges)
- Embedded-answer questions (cloze style) with answers within passages of text
- Embedded descriptive text and graphics

Resource Module

- Supports display of any electronic content, Word, PowerPoint, Flash, Video, Sounds etc
- Files can be uploaded and managed on the server, or created on the fly using web forms (text or HTML)
- External content on the web can be linked to or seamlessly included within the course interface.

Survey Module

- Built-in surveys (COLLES, ATTLS) have been proven as instruments for analysing online classes
- Online survey reports always available, including many graphs. Data is downloadable as an Excel spreadsheet or CSV text file.
- Survey interface prevents partly-finished surveys.

Workshop Module

- Allows peer assessment of documents, and the teacher can manage and grade the assessment.
- Supports a wide range of possible grading scales
- Teacher can provide sample documents for students to practice grading
- Very flexible with many options.

3.1 Installation of

1. Requirements:

Moodle is primarily developed in Linux using Apache, MySQL and PHP (also sometimes known as the LAMP platform), but is also regularly tested with PostgreSQL and on Windows XP, Mac OS X and Netware 6 operating systems. The requirements for Moodle are as follows:

1. Web server software. Most people use Apache, but Moodle should work fine under any web server that supports PHP, such as IIS on Windows platforms.
2. PHP scripting language (version 4.1.0 or later).
3. a working database server: [MySQL](#) or [PostgreSQL](#) are completely supported and recommended for use with Moodle.

Installing from scratch:

1-Installing the Apache web server: The Apache **HTTP Server**, commonly referred to simply as **Apache**, is a web server .Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows, OS/2, TPF, and [eComStation](#). Released under the Apache License, Apache is characterized as free software and open source software.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support mod perl, mod python, [Tcl](#), and PHP. Popular authentication modules include mod_access, mod_auth, mod_digest, and mod_auth_digest, the successor to mod_digest. A sample of other features include SSL and TLS support (mod_ssl), a proxy module, a URL rewriter (also known as a rewrite engine, implemented under mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter). Popular compression methods on Apache include the external extension module, mod_gzip, implemented to help with reduction of the size (weight) of web pages served over HTTP. Apache logs can be analyzed through a web browser using free scripts such as [AWStats](#)/W3Perl or Visitors.

Virtual hosting allows one Apache installation to serve many different actual websites. Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs)

The various steps involved in installing the Apache web server are:

- **Download Apache:** The best place to get Apache is from the Apache HTTP Server download site. Download the binary for Windows32.

- **Extract the Files:** Once you've downloaded the binary, run it to start the self-installation.
- **Customize:** Assuming that there were no problems, you are ready to customize your Apache configuration. This really just amounts to editing the httpd.conf file. This file is located in the C:\Program Files\Apache Group\Apache2\conf directory. You can edit the conf file with any text editor. you can use any text editor, like Notepad. Note: you shouldn't use Word to edit this file. Follow the instructions in this file to edit your configuration the way you want it. More help is available on the Apache Web site.
- **Test Your Server:** Open a Web browser on the same machine and type http://localhost/ in the address box. Specifically, it will say in big letters "Seeing this instead of the website you expected?" This is good news, as it means your server installed correctly.

Installing the MySQL server for database:

- Download the MySQL database installer from the MySQL database site. Be sure to download the Windows (x86) version which includes a Windows Installer. Run the installer .exe file. After the installation check whether the application has correctly installed or not. Run the application and this will open a command window that is asking you for a password. Enter the password that you chose as your root password and we if we find the entry to the mysql directory then the application has installed successfully.

Installing the Moodle :

Download and copy files into place :

After downloading and unpacking the archive, or checking out the files via CVS, you will be left with a directory called "moodle", containing a number of files and folders. You can either place the whole folder in your web server documents directory, in which case the site will be located at **http://yourwebserver.com/moodle**, or you can copy all the contents straight into the main web server documents directory, in which case the site will be simply **http://yourwebserver.com**.

If you are downloading Moodle to your local computer and then uploading it to your web site, it is usually better to upload the whole archive as one file, and then do the unpacking on the server. Even web hosting interfaces like Cpanel allow you to uncompress archives in the "File Manager".

Run the Installer script to create config.php

To run the installer script (install.php), just try to access your Moodle main URL using a web browser, or access **http://yourserver/install.php** directly. Moodle will detect that configuration is necessary and will lead you through some screens to help you create a new configuration file called **config.php**. At the end of the process Moodle will try and write the file into the right location, otherwise you can press a button to download it from the installer and then upload config.php into the main Moodle directory on the server.

General web server settings

Firstly, make sure that your web server is set up to use index.php as a default page (perhaps in addition to index.html, default.htm and so on). In Apache, this is done using a `DirectoryIndex` parameter in your `httpd.conf` file. usually looks like this:

```
DirectoryIndex index.php index.html index.htm
```

Just make sure index.php is in the list (and preferably towards the start of the list, for efficiency).

Secondly, **if you are using Apache 2**, then you should turn on the `AcceptPathInfo` variable, which allows scripts to be passed arguments like `http://server/file.php/arg1/arg2`. This is essential to allow relative links between your resources, and also provides a performance boost for people using your Moodle web site. You can turn this on by adding these lines to your `httpd.conf` file.

```
AcceptPathInfo on
```

Thirdly, Moodle requires a number of PHP settings to be active for it to work. **On most servers these will already be the default settings.** However, some PHP servers (and some of the more recent PHP versions) may have things set differently. These are defined in PHP's configuration file (usually called `php.ini`):

```
magic_quotes_gpc = 1  (preferred but not necessary)
magic_quotes_runtime = 0  (necessary)
file_uploads = 1
session.auto_start = 0
session.bug_compat_warn = 0
```

If you don't have access to `httpd.conf` or `php.ini` on your server, or you have Moodle on a server with other applications that require different settings, then don't worry, you can often still **OVERRIDE** the default settings.

To do this, you need to create a file called **.htaccess** in Moodle's main directory that contains lines like the following. This only works on Apache servers and only when Overrides have been allowed in the main configuration.

```
DirectoryIndex index.php index.html index.htm
```

```
<IfDefine APACHE2>
  AcceptPathInfo on
</IfDefine>
```

```
php_flag magic_quotes_gpc 1
php_flag magic_quotes_runtime 0
php_flag file_uploads 1
```

```
php_flag session.auto_start 0
php_flag session.bug_compat_warn 0
```

You can also do things like define the maximum size for uploaded files:

```
LimitRequestBody 0
php_value upload_max_filesize 2M
php_value post_max_size 2M
```

The easiest thing to do is just copy the sample file from **lib/htaccess** and edit it to suit your needs. It contains further instructions. For example, in a Unix shell:

```
cp lib/htaccess .htaccess
```

Creating a database

You need to create an empty database (eg "*moodle*") in your database system along with a special user (eg "moodleuser") that has access to that database (and that database only). You could use the "root" user if you wanted to for a test server, but this is not recommended for a production system: if hackers manage to discover the password then your whole database system would be at risk, rather than just one database. If you are using a webhost, they will probably have a control panel web interface for you to create your database.

The **Cpanel** system is one of the most popular of these. To create a database in Cpanel,

1. Click on the "**MySQL Databases**" icon.
2. Type "moodle" in the database field and click "**Add Database**".
3. Type a username and password (not one you use elsewhere) in the respective fields and click "**Add User**".
4. Now use the "**Add User to Database**" button to give this new user account "**ALL**" rights to the new database.
5. Note that the username and database names may be prefixed by your Cpanel account name. When entering this information into the Moodle installer - use the full names.

If you have access to Unix command lines then you can do the same sort of thing by typing commands.

Here are some example Unix command lines for MySQL:

```
# mysql -u root -p
> CREATE DATABASE moodle;
> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,INDEX,ALTER ON
moodle.*
    TO moodleuser@localhost IDENTIFIED BY 'yourpassword';
> quit
```

```
# mysqladmin -p reload
```

And some example command lines for PostgreSQL:

```
# su - postgres  
> psql -c "create user moodleuser createdb;" template1  
> psql -c "create database moodle;" -U moodleuser template1  
> psql -c "alter user moodleuser nocreatedb;" template1
```

Creating a data directory

Moodle will also need some space on your server's hard disk to store uploaded files, such as course documents and user pictures. The Moodle installer tries hard to create this directory for you but if it fails then you will have to create a directory for this purpose manually.

For security, it's best that this directory is NOT accessible directly via the web. The easiest way to do this is to simply locate it OUTSIDE the web directory, but if you must have it in the web directory then protect it by creating a file in the data directory called .htaccess, containing this line:

```
deny from all
```

To make sure that Moodle can save uploaded files in this directory, check that the web server software (eg Apache) has permission to read, write and execute in this directory.

On Unix machines, this means setting the owner of the directory to be something like "nobody" or "apache", and then giving that user read, write and execute permissions.

On Cpanel systems you can use the "File Manager" to find the folder, click on it, then choose "Change Permissions". On many shared hosting servers, you will probably need to restrict all file access to your "group" (to prevent other webhost customers from looking at or changing your files), but provide full read/write access to everyone else (which will allow the web server to access your files).

Go to the admin page to continue configuration

Once the basic **config.php** has been correctly created in the previous step, trying to access the front page of your site will take you the "admin" page for the rest of the configuration.

The first time you access this admin page, you will be presented with a GPL "shrinkwrap" agreement with which you **must** agree before you can continue with the setup.

Now Moodle will start setting up your database and creating tables to store data. Firstly, the main database tables are created. You should see a number of SQL statements followed by status messages (in green or red) that look like this:

```
CREATE TABLE course ( id int(10) unsigned NOT NULL auto_increment, category int(10)
unsigned NOT NULL default '0', password varchar(50) NOT NULL default "", fullname
varchar(254) NOT NULL default "", shortname varchar(15) NOT NULL default "", summary text
NOT NULL, format tinyint(4) NOT NULL default '1', teacher varchar(100) NOT NULL default
'Teacher', startdate int(10) unsigned NOT NULL default '0', enddate int(10) unsigned NOT
NULL default '0', timemodified int(10) unsigned NOT NULL default '0', PRIMARY KEY (id))
TYPE=MyISAM
```

SUCCESS

...and so on, followed by: [Main databases set up successfully](#).

If you don't see these, then there must have been some problem with the database or the configuration settings you defined in config.php. Check that PHP isn't in a restricted "Safe Mode" (commercial web hosts sometimes have safe mode turned on). You can check PHP variables by creating a little file containing `<?php phpinfo() ?>` and looking at it through a browser. Check all these and try this page again.

Scroll down the very bottom of the page and press the "Continue" link.

You should now see a form where you can define more configuration variables for your installation, such as the default language, SMTP hosts and so on. Don't worry too much about getting everything right just now - you can always come back and edit these later on using the admin interface. The defaults are designed to be useful and secure for most sites. Scroll down to the bottom and click "Save changes".

If (and only if) you find yourself getting stuck on this page, unable to continue, then your server probably has what I call the "buggy referrer" problem. This is easy to fix: just turn off the "secureforms" setting, then try to continue again.

Next you will see more pages that print lots of status messages as they set up all the tables required by the various Moodle module. As before, they should all be [green](#).

Scroll down the very bottom of the page and press the "Continue" link.

The next page is a form where you can define parameters for your Moodle site and the front page, such as the name, format, description and so on. Fill this out (you can always come back and change these later) and then press "Save changes".

Finally, you will then be asked to create a top-level administration user for future access to the admin pages. Fill out the details with your own name, email etc and then click "Save changes". Not all the fields are required, but if you miss any important fields you'll be re-prompted for them.

Make sure you remember the username and password you chose for the administration user account, as they will be necessary to access the administration page in future.

(If for any reason your install is interrupted, or there is a system error of some kind that prevents you from logging in using the admin account, you can usually log in using the default username of "**admin**", with password "**admin**".)

Once successful, you will be returned to the home page of your new site! Note the administration links that appear down the left hand side of the page (these items also appear on a separate Admin page) - these items are only visible to you because you are logged in as the admin user. All your further administration of Moodle can now be done using this menu, such as:

- creating and deleting courses
- creating and editing user accounts
- administering teacher accounts
- changing site-wide settings like themes etc

But you are not done installing yet! There is one very important thing still to do (see the next section on cron).

Set up cron -- IMPORTANT!

Some of Moodle's modules require continual checks to perform tasks. For example, Moodle needs to check the discussion forums so it can mail out copies of posts to people who have subscribed.

The script that does all this is located in the admin directory, and is called cron.php. However, it can not run itself, so you need to set up a mechanism where this script is run regularly (eg every five or ten minutes). This provides a "heartbeat" so that the script can perform functions at periods defined by each module. This kind of regular mechanism is known as a **cron service**.

Note that the machine performing the cron **does not need to be the same machine that is running Moodle**. For example, if you have a limited web hosting service that does not have a cron service, then you can might choose to run cron on another server or on your home computer. All that matters is that the cron.php file is called regularly.

The load of this script is not very high, so 5 minutes is usually reasonable, but if you're worried about it you can reduce the time period to something like 15 minutes or even 30 minutes. It's best not to make the time period too long, as delaying mail-outs can slow down activity within the course.

First, test that the script works by running it directly from your browser:

<http://example.com/moodle/admin/cron.php>

Now, you need to set up some of way of running the script automatically and regularly.

On Windows systems: The simplest way is to use this little package **moodle-cron-for-windows.zip** which makes this whole thing very easy by installing a small Windows service. Run it and forget about it!

On web hosting services: Your web-based control panel may have a web page that allows you to set up this cron process. For example, on Cpanel system, look for a button called "Cron jobs". In there you can put the same sort of Unix commands as listed below.

3.2 Improvised method adopted by the author for Installation:

Installing the moodle, Apache and MySQL is a very tedious task and installation work can get up very hectic. However these are the basic requirements for running this application there is another way of installation which is much simpler and saves a lot of space also. As a beginner who needs to understand the way the application works and simulating the whole application on a home pc which does not act as a dedicated server for the application. For this we needed a kind of package which would take lesser space and would be much simpler to install. For this a PHP simulator proved out to be quite handy. The simulator known as Easy PHP works for simulation anykind of PHP simulations providing it a support of Apache and Mysql database. Easy PHP creates a virtual server machine for the application to be run on it smoothly. **EasyPHP** is a WAMP software bundle that installs web server services on to the Windows computer and allows quick and easy development of PHP and [MySQL](#) on a [localhost](#) (also known as [127.0.0.1](#)). The package includes an Apache server, a MySQL database, and the PHP extension.

Install EasyPHP:

1. Run the downloaded file: *easyphp1-8_setup.exe*.
2. At the end of the install leave the checkbox selected to Start EasyPHP.
3. If all went well Apache, PHP and MySQL are all installed and running! You should see a black E in your toolbar tray. You can right click on it to get a menu which will let you control the running

Create a database

1. The next thing we need to do is to set up a database for Moodle to use. Right-click the black E in the toolbar tray and choose Administration, then click DB Management (beside PHPMyAdmin). Alternatively, using a browser, go to <http://localhost/mysql/>
2. If asked for a username, use "root" with a *blank password*. You should see a phpMyAdmin web interface that allows you to create a new databases and user accounts.
3. Create a new database by typing "moodle" into the field and hitting the "Create" button. That was easy!
4. You can also create a new user to access your database if you want.

Installing Moodle

1. Copy your moodle files into *C:\Program Files\EasyPHP\www*. You can either copy the entire moodle directory (ie *C:\Program Files\EasyPHP\www\moodle*) or copy all the contents of the moodle directory. If you choose this second option then you will be able to access your Moodle home page using *http://localhost/* instead of *http://localhost/moodle/*.
2. Make a new empty folder somewhere else for Moodle to store uploaded files in, eg: *C:\moodledata*
3. Go into your Moodle folder. Make a copy of *config-dist.php*, and call it *config.php*
====Editing the the config.php file==== Edit *config.php* using a text editor (Notepad will do, just be careful that it doesn't add unwanted spaces at the end).

1. Put in all your new database info:
2.

```
$CFG->dbtype = 'mysql';
```

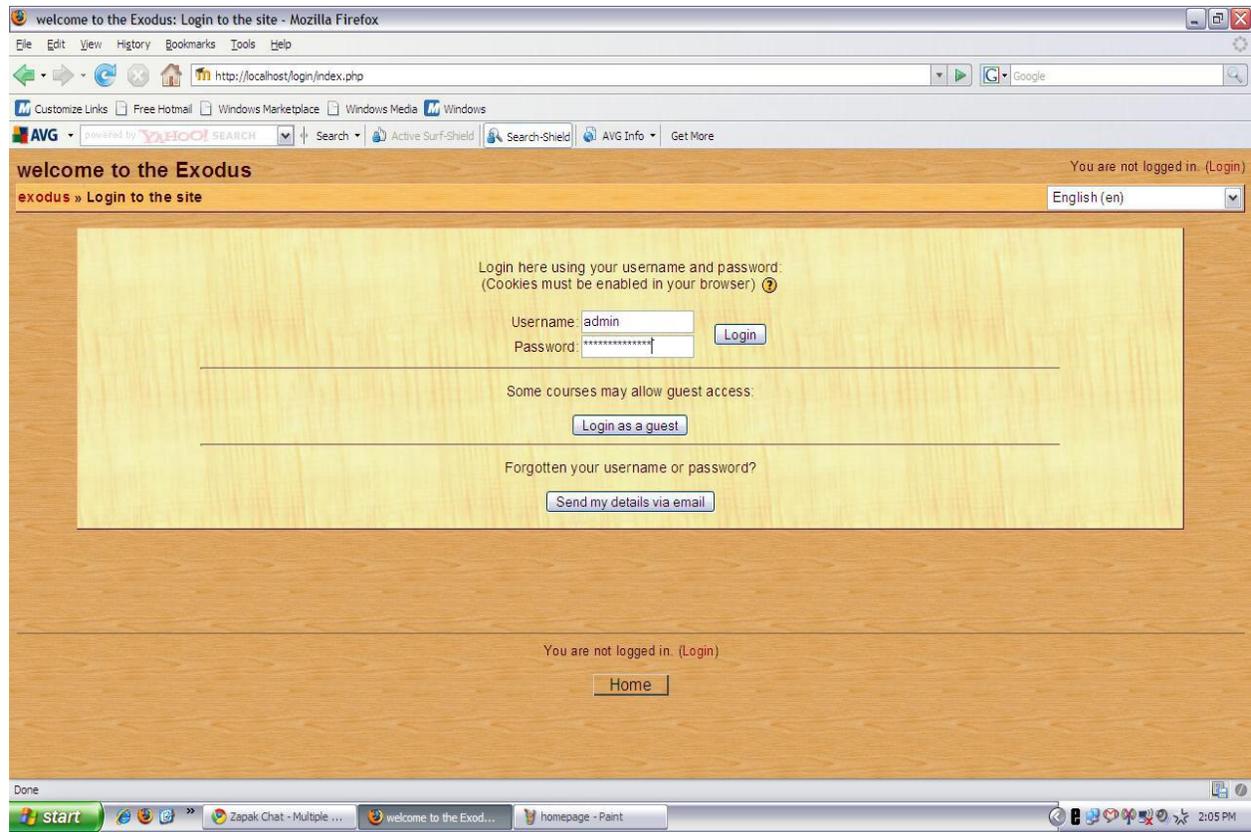
 - ```
$CFG->dbhost = 'localhost';
```
  - ```
$CFG->dbname = 'moodle';
```
 - ```
$CFG->dbuser = 'root';
```
  - ```
$CFG->dbpass = '';
```
 - ```
$CFG->dbpersist = true;
```
  - ```
$CFG->prefix = 'mdl_';
```
3. And put in all your file paths
 - ```
$CFG->wwwroot = 'http://localhost/moodle';
```

 // Use an external address if you know it.
  - ```
$CFG->dirroot = 'C:\Program Files\EasyPHP\www\moodle';
```
 - ```
$CFG->dataroot = 'C:\moodledata';
```
4. Save config.php - you can ignore the other settings if there are any.

This setup proves out to be very handy when we are running out of space or the home pc is not the customized server and has to be used for other purposes also. EasyPhp provides the same kind of environment needed for php application in a much lesser space and with a lot of simplicity. Although Easyphp is not designed to be used on production server it could be very widely used at home PCs for testing or simulating the moodle application.

# CHAPTER-4

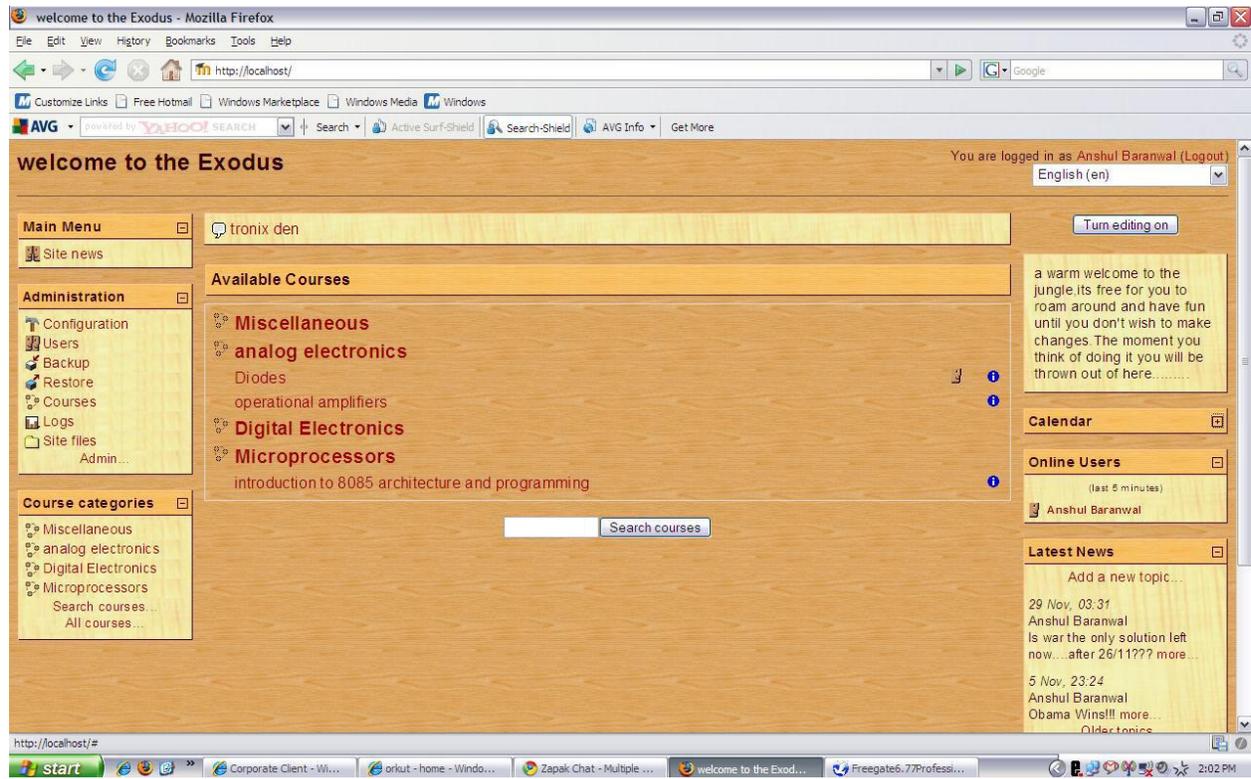
Understanding the various features of the application and leaning how they could be useful in developing a complete course.



## 4.1 Course settings:

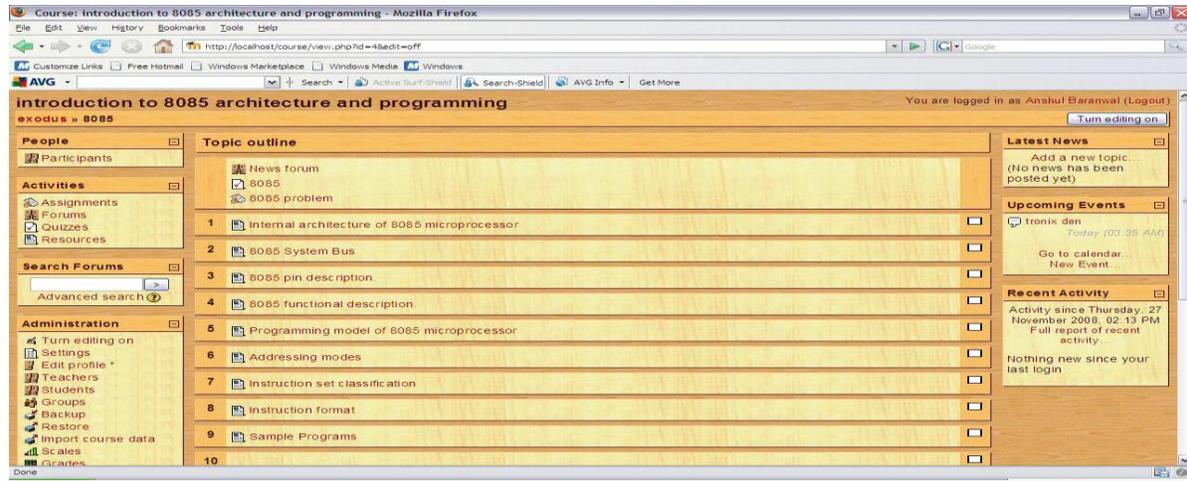
The first thing you should do is look under the “Administration” on your course home page and click on “**Settings...**” (Note that this link, and in fact the whole Administration section is only available to you (and the site administrator). Students will not even see these links).

On the Settings page you can change a number of settings about your course, ranging from its name to what day it starts. The course format that you choose will decide the basic layout of your course, like a template.



## 1. use these icons

-  - the **edit icon** lets you edit whatever it is next to.
-  - the **help icon** will provide you with a **popup help window**
-  - the **open-eye icon** will let you hide something from students
-  - the **closed-eye icon** will make a hidden item available



## 4.2 GENERAL:

### Cookie:

Two cookies are used by this site.

The essential one is the session cookie, usually called **MoodleSession**. It provides continuity and maintain your login from page to page. When you log out or close the browser this cookie is destroyed (in your browser and on the server).

The other cookie is called **MOODLEID**. It just remembers your username within the browser.

### How to Search:

Full-text searching supports a number of options, listed below. You can combine these to specify your search more precisely.

**search for these words**

For basic searching of one or more words anywhere in the texts, just type them separated by spaces. All words longer than two characters are used.

**+search +for these words**

The previous example would also match "information" because it contains "for". To force exact matching of a word, use the plus sign.

**+search -engine**

Use the minus sign if there are particular exact words you don't want included in the search.

**"search engine"**

To search for a particular phrase, use double quotes around it.

**user:Kim**

To search for texts by a particular user, prefix a word from their name with "user:".

**userid:6**

If you know the user id of a particular user, you can search for them like this.

**subject:assessment**

To search for a word within the subject or title of a text only, prefix the word with "subject:".

For advanced searching, press the search button without typing anything in the words field - you will see a complete form that makes it easier to do advanced searches.

## **Upload users:**

Firstly, note that **it is usually not necessary to import users in bulk** - to keep your own maintenance work down you should first explore forms of authentication that do not require manual maintenance, such as connecting to existing external databases or letting the users create their own accounts. See the Authentication section in the admin menus.

If you are sure you want to import multiple user accounts from a text file, then you need to format your text file as follows:

- Each line of the file contains one record
- Each record is a series of data separated by commas
- The first record of the file is special, and contains a list of fieldnames. This defines the format of the rest of the file.

**Required fieldnames:** these fields must be included in the first record, and defined for each user

`username, password, firstname, lastname, email`

**Default fieldnames:** these are optional - if they are not included then the values are taken from the primary admin

`institution, department, city, country, lang, auth, timezone`

**Optional fieldnames:** all of these are completely optional. The course names are the "shortnames" of the courses - if present then the user will be enrolled as students in those courses. Group names must be associated to the corresponding courses, i.e. group1 to course1, etc.

`idnumber, icq, phone1, phone2, address, url, description, mailformat, maildisplay, htleditor, autosubscribe, course1, course2, course3, course4, course5, group1, group2, group3, group4, group5`

- Commas within the data should be encoded as `&#44` - the script will automatically decode these back to commas.
- For Boolean fields, use 0 for false and 1 for true.
- Note: If a user is already registered in the Moodle user database, this script will return the userid number (database index) for that user, and will enrol the user as a student in any of the specified courses WITHOUT updating the other specified data.

Here is an example of a valid import file:

```
username, password, firstname, lastname, email, lang, idnumber, maildisplay,
course1, group1
jonest, verysecret, Tom, Jones, jonest@someplace.edu, en, 3663737, 1,
Intro101, Section 1
reznort, somesecret, Trent, Reznor, reznort@someplace.edu, en_us, 6736733, 0,
Advanced202, Section 3
```

**Language Editing:** To edit languages on this page the files need to be writable by the web server process. Within some of the strings you will notice **\$a** and **\$a->something**. These represent variables that will be replaced by names or other words from within Moodle. When there is only one variable, the \$a form is used. When there are two or more variables, then each one has a name.

### **4.3 QUIZZES:**

**Quiz attempts:** Students may be allowed to have multiple attempts at a quiz. This can help make the process of taking the quiz more of an educational activity rather than simply an assessment.

**Grading method:** When multiple attempts are allowed, there are different ways you can use the grades to calculate the student's final grade for the quiz.

**Highest grade:** The final grade is the highest (best) grade in any attempt.

**Average grade:** The final grade is the average (simple mean) grade of all attempts.

**First grade:** The final grade is the grade gained on the first attempt (other attempts are ignored).

**Last grade:** The final grade is the grade gained on the most recent attempt only.

**Shuffle questions :** If you enable this option, then the order of questions in the quiz will be randomly shuffled each time a student attempts the quiz. This is not related to the use of Random Questions, this is only about the displayed order of questions. The intention is to make it a little harder for students to copy from each other.

## Objectives:

- **Studying the architecture and source code of Moodle.**
- **Exploiting the open-source advantage and improvising some codes.**

### 5.1 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).

## 5.2 How can we 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 is 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).

## 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 only create new languages but to refine existing ones.

## 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.

## 5.3 Moodle Coding Guidelines:

Any collaborative project needs consistency and stability to stay strong. These guidelines are to provide a goal for all Moodle code to strive to. It's true that some of the older existing code falls short in a few areas, but it will all be fixed eventually. All new code definitely must adhere to these standards as closely as possible.

### General Rules

1. All code files should use the .php extension.
2. All template files should use the .html extension.
3. All text files should use Unix-style text format (most text editors have this as an option).
4. All php tags must be 'full' tags like `<?php ?>` ... not 'short' tags like `<? ?>`.
5. All existing copyright notices must be retained. You can add your own if necessary.
6. Each file should include the main config.php file.
7. Each file should check that the user is authenticated correctly, using `require_login()` and `isadmin()`, `isteacher()`, `iscreator()` or `isstudent()`.
8. All access to databases should use the functions in `lib/datalib.php` whenever possible - this allows compatibility across a wide range of databases. You should find that almost anything is possible using these functions. If you must write SQL code then make sure it is: cross-platform; restricted to specific functions within your code (usually a `lib.php` file); and clearly marked.
9. Don't create or use global variables except for the standard `$CFG`, `$SESSION`, `$THEME` and `$USER`.
10. All variables should be initialised or at least tested for existence using `isset()` or `empty()` before they are used.
11. All strings should be translatable - create new texts in the "lang/en" files with concise English lowercase names and retrieve them from your code using `get_string()` or `print_string()`.

12. All help files should be translatable - create new texts in the "en/help" directory and call them using `helpbutton()`.

If you need to update a help file:

- with a minor change, where an old translation of the file would still make sense, then it's OK to make the change but you should notify [translation@moodle.org](mailto:translation@moodle.org)
  - for a major change you should create a new file by adding an incrementing number (eg `filename2.html`) so that translators can easily see it's a new version of the file. Obviously the new code and the help index files should also be modified to point to the newest versions.
13. Incoming data from the browser (sent via GET or POST) automatically has `magic_quotes` applied (regardless of the PHP settings) so that you can safely insert it straight into the database. All other raw data (from files, or from databases) must be escaped with `addslashes()` before inserting it into the database.
  14. **IMPORTANT:** All texts within Moodle, especially those that have come from users, should be printed using the `format_text()` function. This ensures that text is filtered and cleaned correctly.

## 5.4 Coding Style:

I know it can be a little annoying to change your style if you're used to something else, but balance that annoyance against the annoyance of all the people trying later on to make sense of Moodle code with mixed styles. There are obviously many good points for and against any style that people use, but the current style just **is**, so please stick to it.

1. **Indenting** should be consistently 4 spaces. Don't use tabs AT ALL.
2. **Variable names** should always be easy-to-read, meaningful lowercase English words. If you really need more than one word then run them together, but keep them short as possible. Use plural names for arrays of objects.

GOOD: `$quiz`

GOOD: `$errorstring`

GOOD: `$assignments` (for an array of objects)

GOOD: `$i` (but only in little loops)

BAD: `$Quiz`

BAD: `$aReallyLongVariableNameWithoutAGoodReason`

BAD: `$error_string`

3. **Constants** should always be in upper case, and always start with the name of the module. They should have words separated by underscores.

```
define("FORUM_MODE_FLATOLDEST", 1);
```

4. **Function names** should be simple English lowercase words, and start with the name of the module to avoid conflicts between modules. Words should be separated by underscores. Parameters should always have sensible defaults if possible. Note there is no space between the function name and the following (brackets).

```
function forum_set_display_mode($mode=0) {
 global $USER, $CFG;

 if ($mode) {
 $USER->mode = $mode;
 } else if (empty($USER->mode)) {
 $USER->mode = $CFG->forum_displaymode;
 }
}
```

5. **Blocks** must always be enclosed in curly braces (even if there is only one line). Moodle uses this style:

```
if ($quiz->attempts) {
 if ($numattempts > $quiz->attempts) {
 error($strtoomanyattempts, "view.php?id=$cm->id");
 }
}
```

6. **Strings** should be defined using single quotes where possible, for increased speed.

```
$var = 'some text without any variables';
$var = "with special characters like a new line \n";
$var = 'a very, very long string with a '.$single.' variable in it';
$var = "some $text with $many variables $within it";
```

7. **Comments** should be added as much as is practical, to explain the code flow and the purpose of functions and variables.
  - o Every function (and class) should use the popular [phpDoc format](#). This allows code documentation to be generated automatically.
  - o Inline comments should use the // style, laid out neatly so that it fits among the code and lines up with it.

```
/**
 * The description should be first, with asterisks laid out exactly
 * like this example. If you want to refer to a another function,
 * do it like this: { @link clean_param()}. Then, add descriptions
 * for each parameter as follows.
 *
 * @param int $postid The PHP type is followed by the variable name
 * @param array $scale The PHP type is followed by the variable name
```

```

* @param array $ratings The PHP type is followed by the variable name
* @return mixed
*/
function forum_get_ratings_mean($postid, $scale, $ratings=NULL) {
 if (!$ratings) {
 $ratings = array(); // Initialize the empty array
 if ($rates = get_records("forum_ratings", "post", $postid)) {
 // Process each rating in turn
 foreach ($rates as $rate) {
 ...etc
 }
 }
 }
}

```

8. **Space** should be used liberally - don't be afraid to spread things out a little to gain some clarity. Generally, there should be one space between brackets and normal statements, but no space between brackets and variables or functions:

```

foreach ($objects as $key => $thing) {
 process($thing);
}

if ($x == $y) {
 $a = $b;
} else if ($x == $z) {
 $a = $c;
} else {
 $a = $d;
}

```

## Database structures

- Every table must have an auto-incrementing **id** field (INT10) as primary index.
- The main table containing instances of each module must have the same name as the module (eg **widget**) and contain the following minimum fields:
  - id** - as described above
  - course** - the id of the course that each instance belongs to
  - name** - the full name of each instance of the module
- Other tables associated with a module that contain information about 'things' should be named **widget\_things** (note the plural).
- Column names should be simple and short, following the same rules as for variable names.
- Where possible, columns that contain a reference to the id field of another table (eg **widget**) should be called **widgetid**. (Note that this convention is newish and not followed in some older tables)
- Boolean fields should be implemented as small integer fields (eg INT4) containing 0 or 1, to allow for later expansion of values if necessary.
- Most tables should have a **timemodified** field (INT10) which is updated with a current timestamp obtained with the PHP **time()** function.

## 5.5 Security Issues (and handling form and URL data):

1. Do not rely on 'register\_globals'. **Every** variable must be properly initialised in **every** code file. It must be obvious where the variable came from
2. Initialise all arrays and objects, even if empty. `$a = array()` or `$obj = new stdClass();`.
3. Do not use the `optional_variable()` function. Use the `optional_param()` function instead. Pick the correct `PARAM_XXXX` value for the data type you expect. To check and set an optional value for a variable, use the `set_default()` function.
4. Do not use the `require_variable()` function. Use the `required_param()` function instead. Pick the correct `PARAM_XXXX` value for the data type you expect.
5. Use `data_submitted()`, with care. Data must still be cleaned before use.
6. Do not use `$_GET`, `$_POST` or `$_REQUEST`. Use the appropriate `required_param()` or `optional_param()` appropriate to your need.
7. Do not check for an action using something like `if (isset($_GET['something']))`. Use, e.g., `$something = optional_param('something', -1, PARAM_INT)` and then perform proper test for it being in its expected range of values e.g., `if ($something >= 0) { ...`
8. Wherever possible group all your `required_param()`, `optional_param()` and other variables initialisation at the beginning of each file to make them easy to find.
9. Use 'sesskey' mechanism to protect form handling routines from attack. Basic example of use: when form is generated, include `<input type="hidden" name="sesskey" value="<?php echo sesskey(); ?>" />`. When you process the form check with `if (!confirm_sesskey()) {error('Bad Session Key');}`.
10. All filenames must be 'cleaned' using the `clean_filename()` function, if this has not been done already by appropriate use of `required_param()` or `optional_param()`
11. Any data read from the database must have `addslashes()` applied to it before it can be written back. A whole object of data can be hit at once with `addslashes_object()`.
12. Wherever possible, data to be stored in the database must come from POST data (from a form with `method="POST"`) as opposed to GET data (ie, data from the URL line).
13. Do not use data from `$_SERVER` if you can avoid it. This has portability issues.
14. If it hasn't been done somewhere else, make sure all data written to the database has been through the `clean_param()` function using the appropriate `PARAM_XXXX` for the datatype.
15. If you write custom SQL code, make very sure it is correct. In particular watch out for missing quotes around values. Possible SQL 'injection' exploit.
16. Check all data (particularly that written to the database) in **every** file it is used. Do not expect or rely on it being done somewhere else.

# Installing Moodle on Institute server & managing the course system

## 6.1 Installation of :

### 1. Requirements:

Moodle is primarily developed in Linux using Apache, MySQL and PHP (also sometimes known as the LAMP platform), but is also regularly tested with PostgreSQL and on Windows XP, Mac OS X and Netware 6 operating systems. The requirements for Moodle are as follows:

4. Web server software. Most people use Apache, but Moodle should work fine under any web server that supports PHP, such as IIS on Windows platforms.
5. PHP scripting language (version 4.1.0 or later).
6. a working database server: [MySQL](#) or [PostgreSQL](#) are completely supported and recommended for use with Moodle.

**1-Installing the Apache web server:** The Apache HTTP Server, commonly referred to simply as **Apache**, is a web server .Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, FreeBSD, Linux, Solaris, Novell NetWare, Mac OS X, Microsoft Windows, OS/2, TPF, and [eComStation](#). Released under the Apache License, Apache is characterized as free software and open source software.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support mod perl, mod python, [Tcl](#), and PHP. Popular authentication modules include mod\_access, mod\_auth, mod\_digest, and mod\_auth\_digest, the successor to mod\_digest. A sample of other features include SSL and TLS support (mod\_ssl), a proxy module, a URL rewriter (also known as a rewrite engine, implemented under mod\_rewrite), custom log files (mod\_log\_config), and filtering support (mod\_include and mod\_ext\_filter). Popular compression methods on Apache include the external extension module, mod\_gzip, implemented to help with reduction of the size (weight) of web pages served over HTTP. Apache logs can be analyzed through a web browser using free scripts such as [AWStats](#)/W3Perl or Visitors.

Virtual hosting allows one Apache installation to serve many different actual websites. Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs)

The various steps involved in installing the Apache web server are:

- **Download Apache:** The best place to get Apache is from the Apache HTTP Server download site. Download the binary for Windows32.
- **Extract the Files:** Once you've downloaded the binary, run it to start the self-installation.
- **Customize:** Assuming that there were no problems, you are ready to customize your Apache configuration. This really just amounts to editing the httpd.conf file. This file is located in the C:\Program Files\Apache Group\Apache2\conf directory. You can edit the conf file with any text editor. you can use any text editor, like Notepad. Note: you shouldn't use Word to edit this file. Follow the instructions in this file to edit your configuration the way you want it. More help is available on the Apache Web site.
- **Test Your Server:** Open a Web browser on the same machine and type http://localhost/ in the address box. Specifically, it will say in big letters "Seeing this instead of the website you expected?" This is good news, as it means your server installed correctly.

#### **Installing the MySQL server for database:**

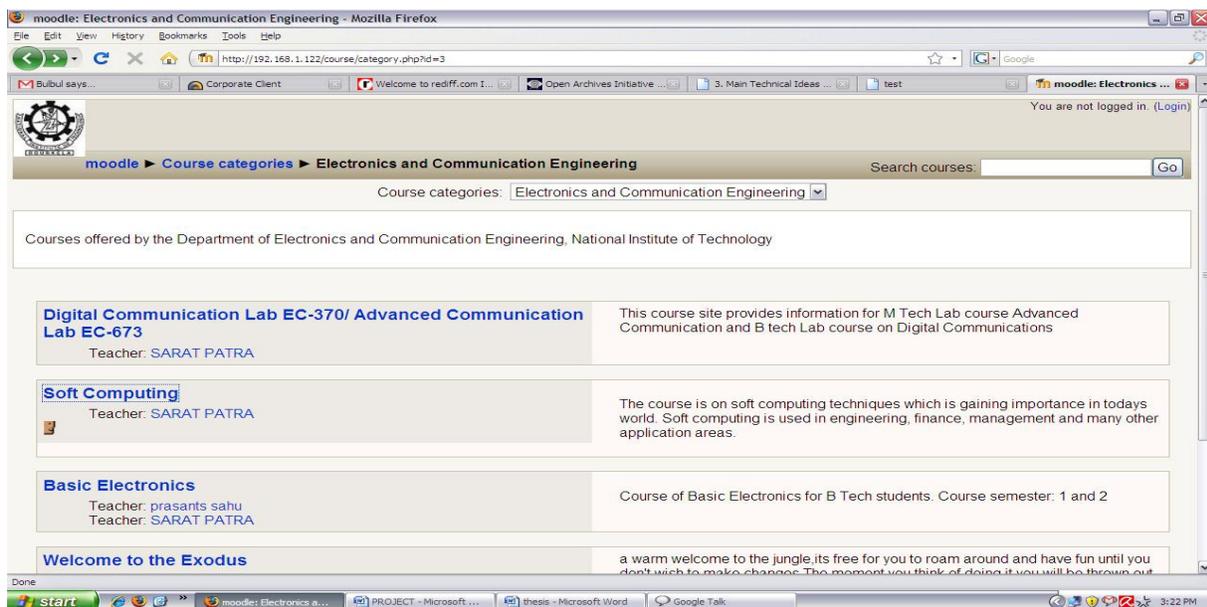
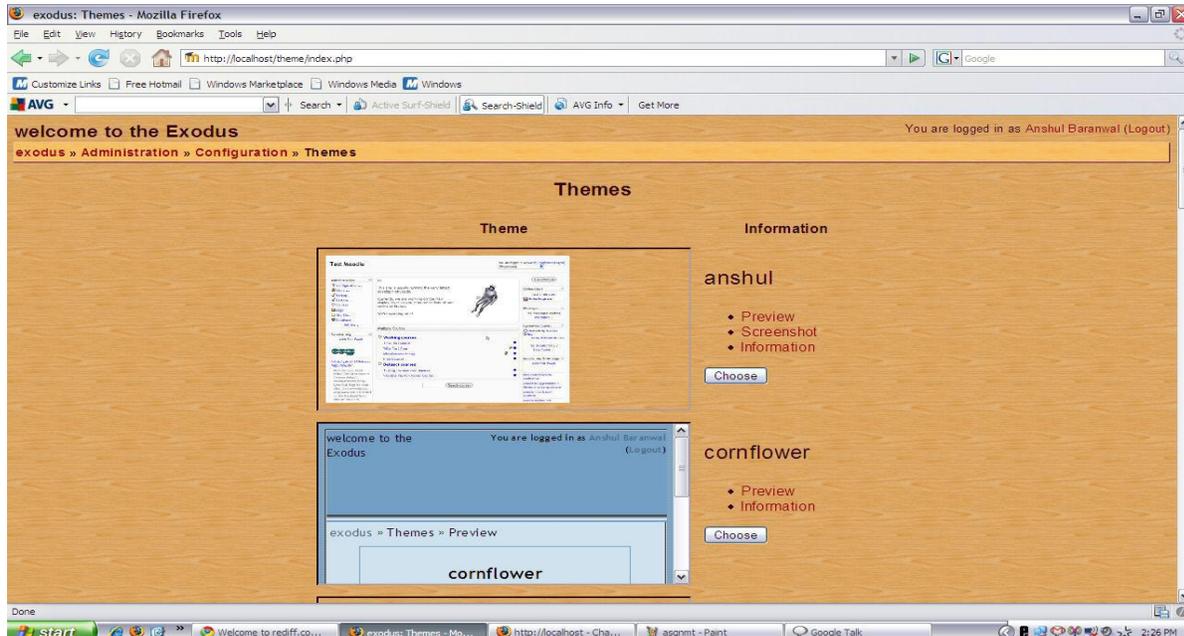
- Download the MySql database installer from the MySQL database site. Be sure to download the Windows (x86) version which includes a Windows Installer. Run the installer .exe file. After the installation check whether the application has correctly installed or not. Run the application and this will open a command window that is asking you for a password. Enter the password that you chose as your root password and we if we find the entry to the mysql directory then the application has installed successfully.

**Installing the Moodle :** The we need to run the moodle installer package to install the moodle contents in the server.

**A more detailed procedure of moodle installation has been described in Chapter 2.**

## 6.2 Developing a customized theme:

After the successful installation of moodle package on the institute server, we needed to make a customized theme for the moodle course system of our institute. We developed a couple of customized themes by using the HTML,CSS and PHP codes . Changed the header and footer labels also by editing the source codes.



## **Customizing the header and footer :**

The lines that you need to customize in `header.html` are copied below. I have added the line numbers. You will not find these in the code, and do not need to add them. I will refer to the line numbers as we discuss how to customize the code:

```
1.<code> if ($home) { // home page ?></code>
2.<code> <div id="header-home"></code>
3.<code> <h1 class="headermain"></h1></code>
4.<code> <div class="headermenu"><?php echo $menu ?></div></code>
5.<code> </div></code>
6.<code> <?php } else if ($heading) { //This is what gets printed on any other
 page with a heading.</code>
7.<code> ?></code>
8.<code> <div id="header-home"></code>
9.<code> <h1 class="headermain"></h1>.</code>
10.<code> <div class="headermenu"><?php echo $menu ?></div>.</code>
11.<code> </div></code>
```

Line 1 tests whether you are on the front page. If you are, then lines 2 through 5 create the header. If you are not on the site's front page, lines 6 through 11 create the header. The only real difference between these two headers is in line 10, which creates the breadcrumb menu at the top of each page:

## **Customizing the icons**

Icons for your Moodle site are in the directory `/moodle/pix`. Subdirectories inside `/pix` organize the icons based on their purpose:

```
/moodle/pix/c holds course icons. For example, it holds the icons that appear
before course names, event names, and
groups.
/moodle/pix/f mostly holds icons for the various file formats. For example,
it holds icons for text files, videos,
and Excel files.
```

/moodle/pix/g holds the pictures that teachers upload to their personal profiles. The default Moodle installation has two files in this directory: a large and a small happy face.

/moodle/pix/i holds Moodle's navigation and function icons. For example, there are icons for the edit function, to hide and show items, and to show the news.

/moodle/pix/m holds icons for currency.

/moodle/pix/s holds icons for smilies—angry, sad, wink, and so on

/moodle/pix/t holds icons that appear in a teacher's functions: backup, delete, hide, and restore.

/moodle/pix/u holds pictures that students upload to their personal profiles.

### **6.3 Adding certain additional modules :**

These modules have been downloaded from the open source directory forum of moodle and some of them have also been coded edited to make it compatible to our version of moodle.

- **Calendar Event Reminder System:** Allows even reminders to be added to a calendar event a specified amount of time before the event is to occur. When the reminder is to be sent, a message or e-mail will be sent to the users affected by the event in question (message if site messaging is enabled).With this feature, the adding calendar event dialogue will allow you to specify reminder information.
- **Question Type: Image target question type:** The image target question asks students to identify a particular area of an image. The javascript version has a draggable bull's eye that the student drags onto the image. The non-javascript version has the student click on the image to submit a response.
- **Moodle Google (Moogle):** This is the Google Search Services as a block. This includes Google Search Services: Web, Image, Video, Book, Blog, News, Map and Custom Website. This is more customizable block in terms of results from the Google.



- **Quiz e-mail notification:** This module would enable an email to be sent automatically on completion (i.e. when all allowed attempts are exhausted) of a quiz. Different emails can be sent to multiple recipients based on pass or fail.
- **Certificate generation:** Generates PDF certificates for students. Some of the features Prints student's name, date, course name, activity or course grade, other text, a code number, the teacher(s) names. Can choose to add images such as seals, logos and watermarks. Can be delivered in the browser, via email, or downloaded. Email alerts can be sent to teachers or others when certificates are issued. Supports different certificate type plugins that can be customized for different needs.
- **Algebra Filter:** It converts algebra code into nice GIF images and provides easy insertion of algebra codes.
- **Course Results:** This block shows information about the highest grades in a course on some grade item. It looks just like the quiz results block but has several differences.
  1. You can choose any grade item to display on, it may be any activity, a grade category result or a whole course result.
  2. You can set block's title. A header text or footer in html format.
  3. We can choose to display user pictures near user names in the table
- **Plagiarism detection block:** In the institutes e often face the plagiarism cases and in case of assignment submission it is quite prominent. The professors are often found quoting that students copy but still they couldn't do anything. This block can prove helpful in curbing plagiarism.  
 This plugin is aimed on digital plagiarism detection: assignments copied from peers from the same institute as well as assignments copied from the internet. Block is based on the document fingerprinting technique for pair wise document comparison.  
 It works in 2 modes
  1. Local plagiarism detection:- look out for similarities in local documents
  2. Internet Plagiarism detection:-Look for similar documents on the internet.
- **Internet Webmail for Moodle:** It is an internet e-mail system for moodle having functionality similar to that of 'webmail'. It also has quickmail option to send group mails to those enrolled in different courses.
- **Generate PDF files from HTML:** This plugin generates PDF documents from the HTML code inside the editor's text area. Prerequisite for this plugin is the TinyMCE 3 integration plugin. The actual conversion from HTML to PDF is performed by dompdf-0.5.1.  
 Clicking this button will generate a PDF document from the current HTML content of the editor's text area.

## **6.4 Preparation of courses:**

1. Basic learning material for micro-processors, diodes, bipolar junction transistor and operational amplifiers were compiled from various sources and then uploaded in the course server.
2. The other things that were uploaded on the server were quiz, forums, chat module and assignment submission modules.
3. These modules were put to use in quiz tests and regular assignment submissions in the ECE branch.
4. A variety of number of problems compiled for the quizzes on the topics operational amplifiers ,bipolar junction transistors & diodes. Almost 100 multiple choice questions were collected for each topic.

The sources from where the problems were collected were:

- [http://wps.prenhall.com/chet\\_paynter\\_electronic2/23/6136/1570916.cw/index.html](http://wps.prenhall.com/chet_paynter_electronic2/23/6136/1570916.cw/index.html)
- [http://wps.pearson.co.uk/ema\\_uk\\_he\\_story\\_electronic\\_3/42/10999/281599.cw/index.html](http://wps.pearson.co.uk/ema_uk_he_story_electronic_3/42/10999/281599.cw/index.html)
- [http://wps.prenhall.com/basic\\_electronic/63/143](http://wps.prenhall.com/basic_electronic/63/143)
- <http://www.learnwell.org/electro.htm>
- <http://www.johnloomis.org/ece323/tests/test3Fo7.pdf>
- <http://www.johnloomis.org/ece323/asgn/asgns/agns.html>
- <http://cobweb.ecn.purdue.edu/~ee255/practice%20exams/e2fa1101.pdf>
- <http://nptel.iitm.ac.in/courses.php3branch=ece>
- [http://www.meriteducation.com/basic\\_electronics\\_tutorial\\_course\\_online.html](http://www.meriteducation.com/basic_electronics_tutorial_course_online.html)
- <http://www.universalclass.com/i/crn/7550149.htm>
- <http://www.science-ebooks.com/electronics/basic-electronics.htm>

### **Books referred for making courses:**

- Micro-electronic circuits by Sedra and Smith
- Electronic circuits & systems by Boylsted
- Electronic devices & circuits by- Jinamic J. Cathey by shaums publications

# PHASE -II

---

## Deployment of an open archives repository



Currently running as

<http://ethesis.nitrkl.ac.in>

# CHAPTER-1

## 1.1 Introduction:

**EPrints** is an open source software package for building open access repositories that are compliant with the Open Archives Initiative Protocol for Metadata Harvesting. It shares many of the features commonly seen in Document Management systems, but is primarily used for institutional repositories and scientific journals. EPrints has been developed at the University of Southampton School of Electronics and Computer Science and released under a GPL license.

The EPrints software is not to be confused with "eprints" (or "e-prints"), which are preprints (before peer review) and [postprints](#) (after peer review), of research journal articles: "[eprints](#)" = preprints + [postprints](#)

## 1.2 Technology:

EPrints is a Web and command-line application based on the LAMP architecture (but is written in Perl rather than PHP). It has been successfully run under Linux, Solaris and Mac OS X. A version for Microsoft Windows is being developed but will be released under a non-GPL license.

Version 3 of the software introduced a (Perl-based) [plugin](#) architecture for importing and exporting data, converting objects (for search engine indexing) and user interface widgets.

Configuring an EPrints repository involves modifying configuration files written in Perl or XML. Web based configuration tools are in development. The appearance of a repository is controlled by HTML templates, stylesheets and inline images. While Eprints is shipped with an English translation it has been translated to other languages through (redistributable) language-specific XML phrase files. Existing translations include Bulgarian, French, German, Hungarian, Italian, Japanese, Russian, Spanish and Ukrainian.

## CHAPTER-2

### 2.1 Installing EPrints through Binary:

---

Currently supported: RedHat Enterprise 4 32bit. This also requires the DAG repository for some Perl dependencies.

#### Adding the EPrints Yum Repository

In `/etc/sysconfig/rhn/sources` add:

```
Dag RPM Repository for Red Hat Enterprise Linux
yum dag http://apt.sw.be/redhat/el4/en/\$ARCH/dag
EPrints RPM Repository for Red Hat Enterprise Linux
yum eprints http://rpm.eprints.org/rhel4/
```

Import the DAG and EPrints distribution keys:

```
wget http://dag.wieers.com/rpm/packages/RPM-GPG-KEY.dag.txt
rpm --import RPM-GPG-KEY.dag.txt
rm RPM-GPG-KEY.dag.txt
wget http://rpm.eprints.org/RPM-GPG-KEY-tdb01r
rpm --import RPM-GPG-KEY-EPRINTS-2007
rm RPM-GPG-KEY-EPRINTS-2007
```

#### Dependencies

**This will be redundant when there is a release EPrints 3, for now install dependencies and then use the release tarball.**

```
up2date perl-DBI perl-DBD-MySQL perl-Data-ShowTable \
perl-MIME-Lite perl-Unicode-String perl-Time-HiRes \
perl-XML-LibXML perl-XML-GDOME
```

#### Install EPrints 3

You can optionally install XML : : GDOME using:

```
up2date perl-XML-GDOME
```

To switch to using GDOME

modify `/opt/eprints3/perl_lib/EPrints/SystemSettings.pm`:

```
enable_libxml => 0
```

## 2.2 Installing E-prints through source:

EPrints requires Apache (with mod\_perl), MySQL and Perl with some extra modules. Ideally you also want wget, tar and unzip. EPrints bundles some perl modules which it uses, to save you installing them.

Other tools required are **wget**, **tar**, **gunzip** and **unzip** are required to allow users to upload files as .tar.gz or .zip or to captures them from a URL. As well as we need PDF to text i.e. xpdf package, HTML and LaTeX tool that allows eprints to look at the strings that look like Latex equations and render them as images.

## 2.3 Getting Started:

### Creating an Archive

---

EPrints 3 can run multiple archives under one install. Multiple archives will require giving additional DNS aliases to the machine running EPrints, EPrints can then create all the parts of the apache configuration file needed to run the virtual hosts.

#### **Running epadmin**

Make sure MySQL is actually running.

Change to your eprints user (probably "eprints").

Change directory to the eprints directory (/opt/eprints3 by default) and run

```
bin/epadmin create
```

You will get the following prompts (note that when you see something in [square brackets], it's the default value and can be selected by simply hitting enter)

- Archive ID - the system name for your archive. It's probably a good idea to think of something short and memorable. Once entered, an archive/<archive\_id> directory will be created, and the standard configuration files will be copied in.

- Configure vital settings - Hit enter to say 'yes'. This will lead to more prompting about core settings:
  - Hostname - What someone will type into a web browser to get to your archive. Make sure that your systems team have a DNS alias pointing to your server for this.
  - Webservice Port - Which port to you want to serve the archive on? The default is 80, so unless you can think of a good reason not to, just hit enter to accept the default.
  - Alias - You can enter any number of aliases that will take users to this archive. Enter a '#' when you don't want to enter any more. You could have your archive served `oneprints.myorganisation.org` and `eprints.myorg.org`. As with the Hostname, your systems team need to be informed about these aliases too.
  - Administrator Email - Enter the email address of the repository administrator. This will allow your repository users to send email to the right person.
  - Archive Name - The full name of your archive. By default, this will be used on many of the pages, and in the title bar of the browser.
  - Write these core settings - If you don't say 'yes', then you entered all that data for nothing.
- Configure database - EPrints makes extensive use of a MySQL database. Enter 'yes' to configure this.
  - Database Name - The internal name of your database. It makes sense to use the Archive ID for this, but you don't have to. You don't need to create this database, epadmin will do it for you.
  - MySQL Host - The address of the server that the database is running on. If the database is on the same machine as the EPrints installation, enter 'localhost'.
  - MySQL Port - You probably don't need to enter a value. If you have problems connecting to the database, talk to your systems team.
  - MySQL Socket - As with MySQL Port, it's unlikely that you need to enter anything.
  - Database User - The username with which to log into the MySQL Database. You don't need to create this user, epadmin will do it for you. If you enter a MySQL username that already exists, it will be overwritten by epstats.
  - Database Password - The password for the Database User.
  - Write these database settings - You should write them, or you'll lose them.
  - Create database <Database Name> - Say yes, and epadmin can create the database and populate it with all the right tables. If you've already created a database and a user for this archive, say no.

- MySQL Root Password - To create the database and the user, eadmin needs the MySQL Root Password. This is not saved anywhere. It is used to log into mysql, create the database and create the user with the right access rights. The password is then forgotten.
- Create database tables - say yes to have eadmin create all the database tables.
- Create an initial user -
  - Do you want to build the static web pages - There are a number of pages in EPrints which change very rarely. These are the static pages. The Home page and the About page are examples of static pages. Stylesheets are also static. These pages need to be built.
  - Do you want to update the apache config files? (you still need to add the 'Include' line) - Your archive has a number of files which it uses to configure the web server. These should be updated, so say 'yes'.
  - Before exiting, eadmin will display information about configuring the webserver.

***example configuration***

The script will run through a number of configuration options, an example of which is listed below. Please change the settings to suit your site configuration.

```

Create an EPrint Repository
Please select an ID for the repository, which will be used to
create a directory
and identify the repository. Lower case letters and numbers, may
not start with
a number. examples: "lemurprints" or "test3"
Archive ID? uclapr
We need to create /var/lib/eprints3/archives/uclapr, doing it
now...
Getting uid and gid information for apache
UID: 48
GID: 48
Configure vital settings? [yes] ? yes
Hostname? eprints.adastral.ucl.ac.uk
Webserver Port [80] ? 80
Alias (enter # when done) [#] ? #
Administrator Email? *****@adastral.ucl.ac.uk
Archive Name [Test Repository] ? UCL Adastral Park Repository
Write these core settings? [yes] ? yes
Configuring Database for: uclapr
Database Name [uclapr] ? *****
MySQL Host [localhost] ? localhost
MySQL Port (# for no setting) [#] ?
MySQL Socket (# for no setting) [#] ?
Database User [*****] ?

```

```
Database Password?
Write these database settings? [yes] ? yes
Create database "uclapr" [yes] ? yes
MySQL Root Password?
Create database tables? [yes] ? yes
Create an initial user? [yes] ? yes
Enter a username [admin] ? admin
Select a user type (user|editor|admin) [admin] ? admin
Enter Password? *****
Email? *****@adastral.ucl.ac.uk
Do you want to build the static web pages? [yes] ? yes
Do you want to import the LOC subjects? [yes] ? no (I have a
custom subject file)
Do you want to update the apache config files? (you still need
to add the 'Include' line) [yes] ? yes
```

Open a browser, and enter the hostname in the address bar. You should see your new archive, ready to be [branded](#).

## **2.4 Maintenance(required):**

EPrints front end web pages and abstracts *\*are not\** automatically updated when you make changes to the repository. To apply your changes and update the web pages:

### **Generate Views**

```
eprints@host$ bin/generate_views yourarchivename
```

### **Generate Statics**

```
eprints@host$ bin/generate_static yourarchivename
```

### **Generate Abstracts**

```
eprints@host$ bin/generate_abstracts yourarchivename
```

### Studying the architecture of e-prints

#### 3.1 EPrints Directory Structure:

- eprints3/
  - **AUTHORS** - Text file telling you who did what in EPrints.
  - **CHANGELOG** - Text file documenting every single change made since EPrints 2 Alpha.
  - **COPYING** - The GPL License
  - **NEWS** - Summary of major changes between versions.
  - **README** - Text file telling you how to install and where to find help.
  - **SIGNATURES** - Checksums of installed files. Used by upgrade tool to see if files have been modified since installation.
  - **VERSION** - The current version of EPrints.
  - **archives/** - Contains configuration and data for each repository
  - **bin/** - Contains command line tools.
  - **cfg/** - Contains site-wide configuration files.
  - **cgi/** - Contains CGI files (dynamic web pages)
  - **lib/** - Contains the read-only data used by EPrints (do not edit!)
  - **perl lib/** - Contains the perl-libraries used by EPrints
  - **testdata/** - Contains a tool + data to import example data into your repository.
  - **var/** - Used to store files output by EPrints including logs and process ID files.

#### 3.2 Metadata:

Here are many different types of metadata field. The type controls how a field is rendered, indexed, searched and so forth. A field always has a type and a name property, and usually has several more. Most properties are documented on this page, but some properties are only available to certain types of field, and they are listed on the page for that field.

Some of these subclasses provide very rich features, others very simple. For example the url field works just like the text field except that it's only valid if it looks like a url and when rendered it is a hyper-link.

A metadata field describes one field of data in one type of Data Object. For example the "title" field of an EPrint Object or the "email" field in a User Object. Every Data Object has system fields (which are set by the system, and not alterable), but the User Object and EPrint Object have additional fields which are configured on a per-repository basis. These can be customised in the user\_fields.pl and eprint\_fields.pl files. Note that changing these files does not automatically modify the underlying database so should (generally) only be done before the database is created. Some metadata properties do not affect the database, and are marked as such. Default values marked \*config indicate that the default value for the repository may be modified in the configuration file field\_property\_defaults.pl

### **3.3 Repository Configuration:**

This is the entire configuration for a single EPrints repository. There are several different types of configuration file.

- **eprints3/archives/ARCHIVEID/cfg**
  - **apache.conf** - Repository specific apache config options which appear OUTSIDE the virtual host block.
  - **apachevhost.conf** - Repository specific apache config options which appear INSIDE the virtual host block. Nb. That this is in addition to all the directives created automatically.
  - **autocomplete/** - Location of autocomplete datafiles.
  - **cfg.d/** - General configuration files.
  - **citations/** - Citation files (these describe how to show objects in search results and so forth).
  - **lang/** - Language specific files for this repository (phrases, static pages and images, site template)
  - **namedsets/** - These are files which contain lists of values for named set fields.
  - **static/** - Source for pages and images in the repository which are not language dependent.
  - **subjects** - Text file containing initial subject tree that will be imported with eprints3/bin/import\_subjects
  - **workflows/** - Workflow files describing the pages which edit EPrint and User records.

# CHAPTER-4

## 4.1 OAI-PMH(Open Archives Initiative-Protocol for Metadata Harvesting):

Eprints is based on this important open initiative known as open archives initiatives.

The Open Archives Initiative develops and promotes interoperability standards that aim to facilitate the efficient dissemination of content. OAI has its roots in the open access and institutional repository movements. Over time the work of OAI has expanded to promote broad access to digital resources for eScholarship, eLearning, and eScience.

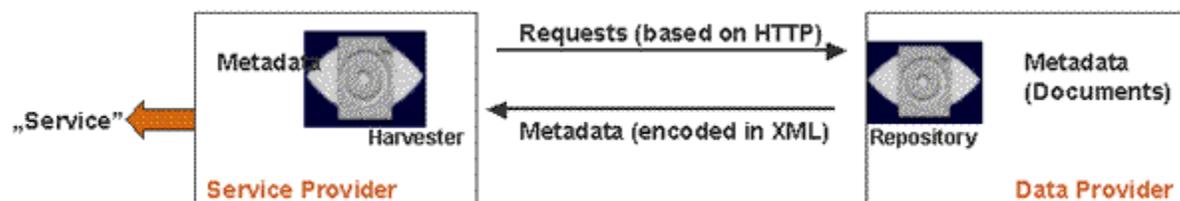
Eprints involve the OAI-PMH protocol i.e. open archives initiative protocol for metadata harvesting. The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a low-barrier mechanism for repository interoperability. *Data Providers* are repositories that expose structured metadata via OAI-PMH. *Service Providers* then make OAI-PMH service requests to harvest that metadata. OAI-PMH is a set of six verbs or services that are invoked within HTTP.

### Main Technical Ideas of OAI-PMH:

- world-wide consolidation of scholarly archives
- free access to the archives (at least: metadata)
- consistent interfaces for archives and service provider
- low barrier protocol / effortless implementation (e.g., because based on HTTP, XML, DC)

### OAI-PMH: overview and structure model

There are two groups of 'participants': Data Providers and Service Providers.



**Data Providers** (open archives, repositories) provide free access to metadata, and may, but do not necessarily, offer free access to full texts or other resources. OAI-PMH provides an easy to implement, low barrier solution for Data Providers.

**Service Providers** use the OAI interfaces of the Data Providers to harvest and store metadata. Note that this means that there are no live search requests to the Data Providers; rather, services are based on the harvested data via OAI-PMH. Service Providers may select certain subsets from Data Providers (e.g., by set hierarchy or date stamp). Service Providers offer (value-added) services on the basis of the metadata harvested, and they may enrich the harvested metadata in order to do so.

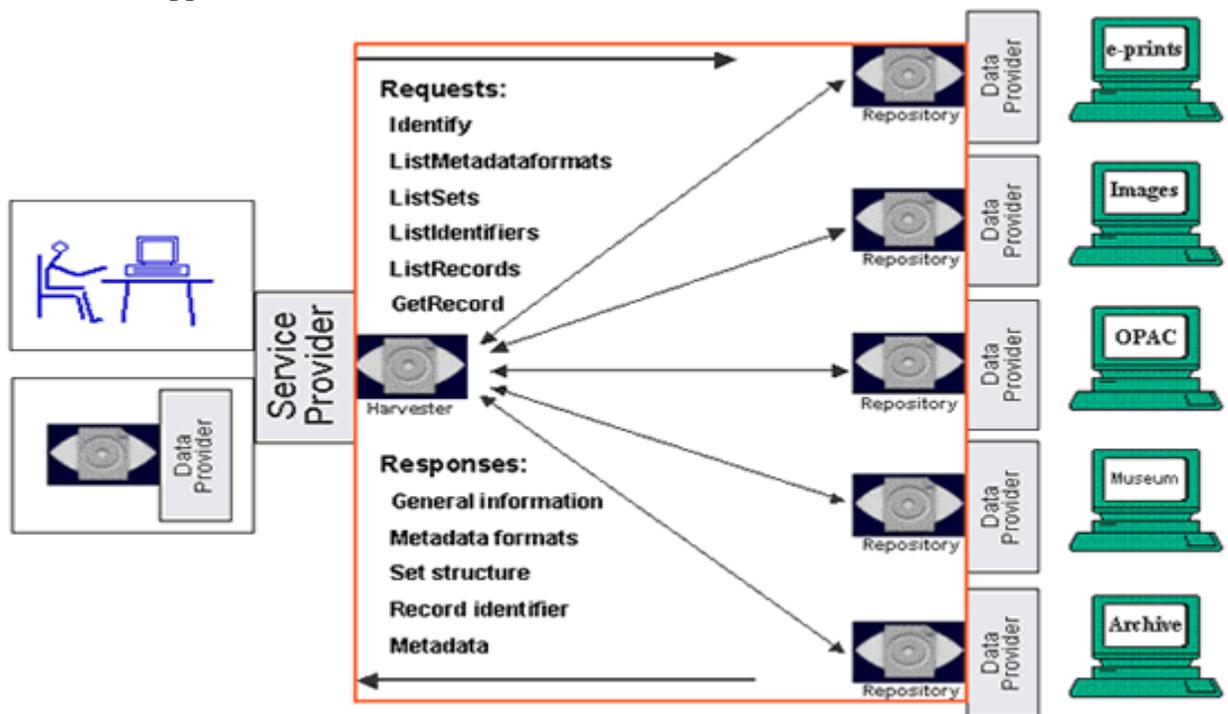
The OAI-PMH protocol is based on HTTP. Request arguments are issued as GET or POST parameters. OAI-PMH supports six request types (known as "verbs"), e.g., <http://archive.org?verb=ListRecords&from=2002-11-01>.

Responses are encoded in XML syntax. OAI-PMH supports any metadata format encoded in XML. Dublin Core is the minimal format specified for basic interoperability.

Error messages are HTTP-based.

Data Providers may define a logical set hierarchy to support levels of granularity for harvesting by Service Providers. Date stamps flag the last change of the metadata set, and thus provide further support for granularity of harvesting.

OAI-PMH supports flow control.



## 4.2 Protocol details:

### *-- Records --*

A record is the metadata of a resource in a specific format. A record has three parts: a header and metadata, both of which are mandatory, and an optional about statement. Each of these is made up of various components as set out below.

header (mandatory)

- identifier (mandatory: 1 only)
- timestamp (mandatory: 1 only)
- setSpec elements (optional: 0, 1 or more)
- status attribute for deleted item

metadata (mandatory)

- XML encoded metadata with root tag, namespace
- repositories must support Dublin Core, may support other formats

about (optional)

- rights statements
- provenance statements

### *-- Timestamps --*

A timestamp is the date of last modification of a metadata record. Timestamp is a mandatory characteristic of every item. It has two possible levels of granularity: YYYY-MM-DD or YYYY-MM-DDThh:mm:ssZ.

The function of the timestamp is to provide information on metadata that enables selective harvesting using from and until arguments. Its applications are in incremental update mechanisms. It gives either the date of creation, last modification, or deletion. Deletion is covered with three support levels: no, persistent, transient.

### *-- Metadata schema --*

OAI-PMH supports dissemination of multiple metadata formats from a repository. The properties of metadata formats are:

- id string to specify the format (metadataPrefix)
- metadata schema URL (XML schema to test validity)
- XML namespace URI (global identifier for metadata format)

Repositories must be able to disseminate unqualified Dublin Core. Further arbitrary metadata formats can be defined and transported via the OAI-PMH. Any returned metadata must comply

with an XML namespace specification. The Dublin Core Metadata Element Set contains 15 elements. All elements are optional, and all elements may be repeated.

### The Dublin Core Metadata Element Set:

|             |             |          |
|-------------|-------------|----------|
| Title       | Contributor | Source   |
| Creator     | Date        | Language |
| Subject     | Type        | Relation |
| Description | Format      | Coverage |
| Publisher   | Identifier  | Rights   |

#### *-- Sets --*

Sets enable a logical partitioning of repositories. They are optional archives do not have to define Sets. There are no recommendations for the implementation of Sets. Sets are not necessarily exhaustive of the content of a repository. They are not necessarily strictly hierarchical. It is important and necessary to have negotiated agreements within communities defining useful sets for the communities.

- function: selective harvesting (**set** parameter)
- applications: subject gateways, dissertation search engine, and others
- examples
  - publication types (thesis, article, ?)
  - document types (text, audio, image, ?)
  - content sets, according to DNB (medicine, biology, ?)

#### *-- Request format --*

Requests must be submitted using the GET or POST methods of HTTP, and repositories must support both methods. At least one key=value pair: verb=RequestType (where RequestType is some type of request such as ListRecords) must be provided. Additional key=value pairs depend on the request type.

example for GET request: `http://archive.org/oai?verb=ListRecords&metadataPrefix=oai_dc`

The encoding of special characters must be supported; for example, ":" (host port separator) becomes "%3A"

## -- Response --

Responses are formatted as HTTP responses. The content type must be text/xml. HTTP-based status codes, as distinguished from OAI-PMH errors, such as 302 (redirect) and 503 (service not available) may be returned. Compression codes are optional in OAI-PMH, only identity encoding is mandatory. The response format must be well-formed XML with markup as follows:

1. XML declaration  
(`<?xml version="1.0" encoding="UTF-8" ?>`)
2. root element named **OAI-PMH** with three attributes  
(`xmlns`, `xmlns:xsi`, `xsi:schemaLocation`)

## -- Flow control --

Four of the request types return a list of entries. Three of them may reply with 'large' lists. OAI-PMH supports partitioning. Those managing a repository make the decisions on partitioning: whether to partition and how. The response to a request includes:

- incomplete list
- resumption token
  - + expiration date,
  - size of complete list,
  - cursor (optional)

For a new request with same request type:

- resumption token as parameter
- all other parameters omitted!

The response includes the next (which may be the last) section of the list and a resumption token. That resumption token is empty if the last section of the list is enclosed.

## -- Errors and exceptions --

Repositories must indicate OAI-PMH errors by the inclusion of one or more error element



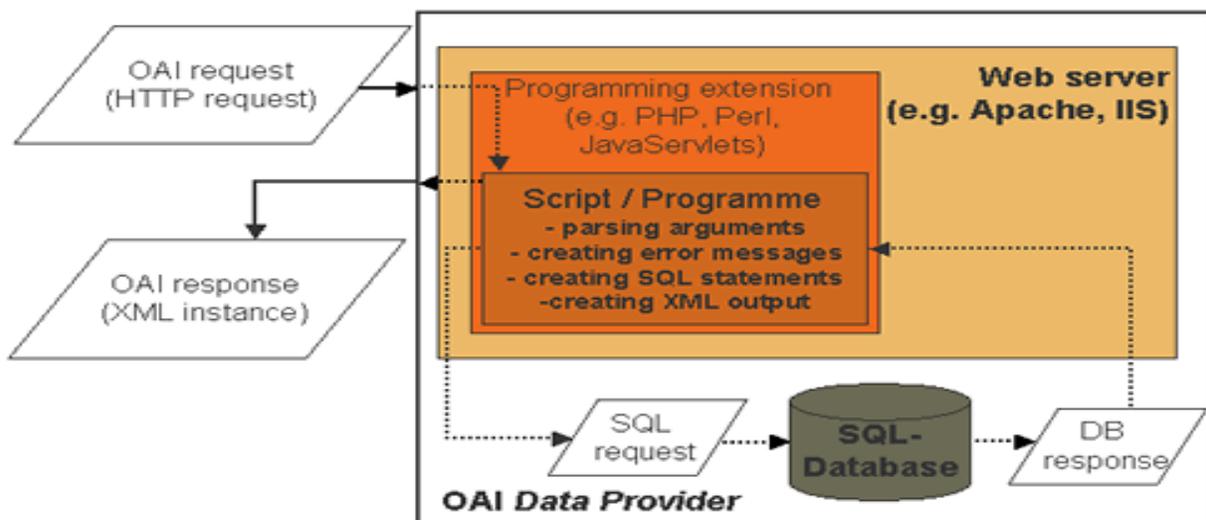
## 4.3 Data Providers:

### **Prerequisites:**

These are the things you must, should, or may have in place in order to implement OAI-PMH as a Data Provider:

- **metadata** on resources ("items")  
These should be stored in a database (such as an SQL database). A file system may be necessary. It is necessary to have a unique identifier for each item.
- **Web server**, accessible via the Internet, e.g. Apache, IIS
- **programming interface / API**
  - e.g. Perl, PHP, Java-Servlet
  - web server extension
  - access to database (or filesystem)
- **archive identifier / base URL**
- **unique identifier for each item**
- **metadata format** (one or more; at least: unqualified Dublin Core)
- **datestamps for metadata** (created / last modified)
- **logical set hierarchy** (may have)  
This is most usefully by agreement within communities, especially subject communities
- **flow control** by implementation of resumption token (optional, but 'larger' repositories should have it)

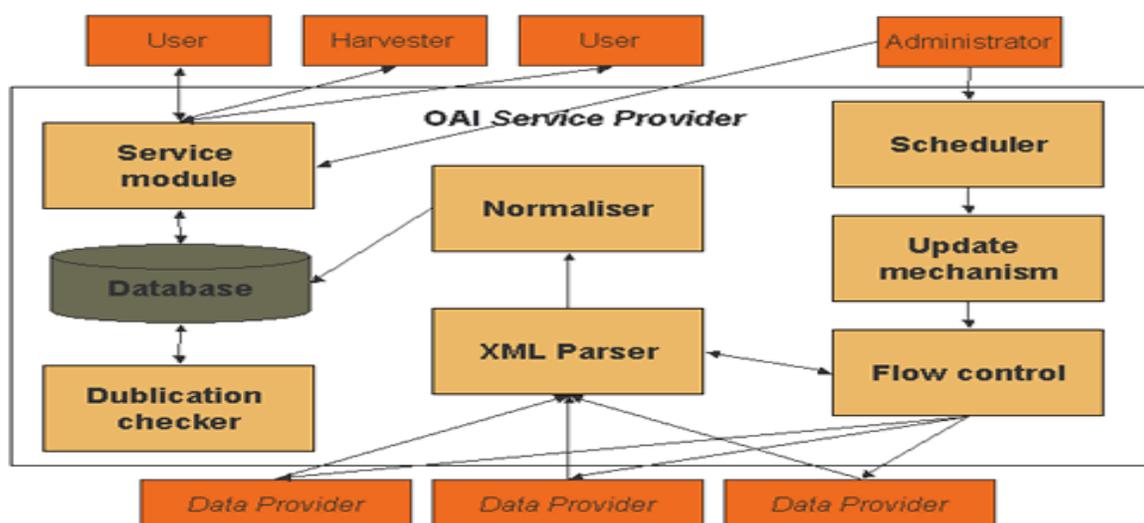
### Data Provider: components and architecture



## Components:

- **Argument Parser** validates OAI requests.
- **Error Generator** creates XML responses with encoded error messages.\
- **Database Query / Local Metadata Extraction** retrieves metadata from the repository, according to the required metadata format.
- **XML Generator / Response Creation** creates XML responses with encoded metadata information.
- **Flow Control** realises incomplete list sequences for 'larger' repositories. It uses resumption token as the control mechanism.

## 4.4 Service Provider:



## prerequisites :

There are three technical infrastructure prerequisites for implementing an OAI-PMH Service Provider that will harvest metadata from Data Providers via OAI-PMH:

- an Internet-connected server
- a database system (relational or XML)
- a programming environment. (The programming environment must be one that can issue HTTP requests to web servers, can issue database requests, and includes an XML parser.)

## Service Provider: components and architecture

- **Archive management** involves the selection of repositories to be harvested. Entries to your list of repositories to be harvested may be made manually or you can automatically add or remove archives using the official registry.
- **Request Component** creates HTTP requests and sends them to OAI repositories (Data Provider). It demands metadata using the allowed verbs of the OAI-PMH. It may do selective harvesting using the set parameter.
- **Scheduler** realises timed and regular retrieval of the associated archives. The simplest case would be manual initiation of the jobs, but this can be automated, e.g., as a cron job.
- **Flow Control** is implemented via resumption token, partitioning of the result list into incomplete sections with a new request to retrieve more results. An HTTP error 503 (service not available) allows analysis of the response to extract a “retry-after” period.
- **Update Mechanism** realises the consolidation of metadata which have been harvested earlier (merge old and new data). The easiest case would be to delete all ‘old’ metadata from each repository before harvesting it again. A reasonable alternative is to do an incremental update (from parameter) – insert *new* metadata and overwrite *changed / deleted* metadata (assignment using the unique identifiers).
- **XML Parser** analyses the responses received from the repositories, with validation using the XML schema, and transforms the metadata encoded in XML into the internal data structure.
- **Normaliser** transforms data in different metadata formats into a homogenous structure. It harmonises representation of, for example, date, author, language code. It may map between or translate different languages.
- **Database** receives the output of the normaliser mapping the XML structure of the metadata into a relational database that will handle multiple values of elements. An alternative is to use an XML database.
- **Duplication Checker** merges identical records from different data providers. One possibility for implementing this is by the unique identifier for each item (for example, by URN). However, this solution is often not easily practicable and is not risk or error free.
- **Service Module** provides the actual service to the 'public'. The basis for a service provided is the harvested and stored records of the associated archives. That is, it uses only the local database for requests etc., and thus it does not make calls on the Data Providers during operation.

## **4.5 SWORD PROTOCOL:**

SWORD is a lightweight protocol for remotely depositing content into repositories. The core of the Sword Project is a cross platform library written in C++, providing access, search functions and other utilities to a growing collection of over 200 texts in over 50 languages. Any software based on their API can use this collection.

The project is one of the primary implementers of and contributors to the Open Scripture Information Standard (OSIS), a standardized XML language for the encoding of scripture. The software is also capable of utilizing certain resources encoded in using the Text Encoding Initiative (TEI) format and maintains deprecated support for Theological Markup Language (ThML) and General Bible Format (GBF).

### **configuring sword :**

configuring sword.pl

archives/<your repo>/cfg/cfg.d/sword.pl

# **Exploiting the open-source advantage and improvising the codes.**

### **How to contribute:**

There's a number of different ways you can contribute to the EPrints project. Always make an entry on <http://files.eprints.org/> for your contribution, even if you don't upload the files. This will make it the one-stop place for people to find EPrints extensions.

- Plug ins
- Scripts
- Themes
- Translations
- Other neighborly things to do
- Extension Packages

### **5.1 Write a Plug-in:**

The plugin system for EPrints 3 has been developed to make it easy and share the most common extensions to the code without having to hack the core system (and causing yourself problems with upgrades etc.)

When the system loads, it automatically loads all modules in the `perl_lib/EPrints/Plugin/` directory, so for simple plugins you just drop them in that directory and you're done!

When a plugin is loaded it has a registration method which is called which tells the core EPrints system what this plugin does. EPrints then makes it available as appropriate.

Clever plugins can detect features they need and adapt to use the tools available, or disable themselves if they are missing required tools (rather than crash the system). Some specialised plugins are disabled in their default state and must be enabled in the repository configuration.

Another cool thing is that plugins are Perl Objects, which means you can subclass them. Here's an real-world example: We have a research group which uses BibTeX but over the years standardised within the group on an extra field. This is not a valid bibtex field, but are essential to their working because they have ancient and essential scripts which depend on it. To handle this we can subclass the default BibTeX Export plugin and override a single method (the data

mapping one). We then just call the original parent plugins mapping method to do all the heavy lifting, then just add our non-standard extra field

### ***Import Plugin***

These are used to import data into a repository. They can take datafiles directly, or they can take an ID of a record that can be retrieved in a known way, or a URL of a file, or... whatever.

These are a bit trickier to write than export plugins as parsing data is harder than just "print"ing it, but they are still reasonably straight forward.

### ***Screen Plugin***

These handle (almost) all the user interface screens. Pages like "Review" and "Profile" are just built-in plugins. You can add your own very easily.

### **Examples that we could create...**

- Birds Eye View - a view of various statistics on the database, all in one page.
- Spellchecking Tab - an additional tab in the item control page which checks the spelling on certain fields.
- Bulk Delete tool - a tool which takes a list of eprintid's and deletes them all in a fell swoop.

Look at the existing Screen Plugins for an idea of how they work. They can be very simple.

### ***Input Component Plugin***

These handle how the workflow components are rendered. Built in components include the default (one field) component, the multiple fields component, the upload component, the subject component (which does pretty things to a field of type "subject") and the XHTML component. You can add your own or sub-class existing ones.

### **Convert Plugin:**

These are used for two things, currently.

- Converting the full text of documents into utf-8 text for search indexing
- Converting images and pdfs into thumbnails and previews

### **Some examples that we could create:**

- RTF to utf-8 to allow rich text documents to be indexed.
- Powerpoint to Thumbnail to allow thumbnail and previews of powerpoint slides
- Video to Thumbnail/Preview to make a still preview of a video file.

## 5.2 Translation:

In a multilingual site you probably let the visitors to choose the language of the session. It is determined by the default setting of the browser preferences, but sometimes users want to change this. Manual setting of the session's language can be done by the port of the *set\_language* script from earlier Eprints versions, and should go into the [cgi](#) directory. A handy place for the *set\_language* URL is in the menu at the top of the page. As the the default page template is repository dependent, you should edit `archives/ArchiveID/lang/en/templates/default.xml`, the English language template for that repository as follows (but see the remarks above):

```
<ul class="ep_tm_menu">
 Home
 Language
 About
 . . .

```

The template page on other languages should not contain this *Language* item as not necessarily will people recognise it. Rather use a button which reverts the language to English, i.e. insert the following into `archives/ArchiveID/lang/XX/templates/default.xml` for all other languages XX except for English:

```
In
English
```

If you want a little more fancy layout, you might consider using flag images to be copied to the directory `lib/static/style/images/flags/`. If you define a phrase (see the `#Phrases` section) of the form `"cgi/set_lang:lang_XX"` then the *set\_lang* script will use that phrase to render a link to that language. A typical format could be

```
<epp:phrase id="cgi/set_lang:lang_hu"><epc:pin name="link">

 Hungarian</epc:pin>
</epp:phrase>
```

## 5.3 Populating the database of thesis of NIT Rourkela at the ethesis site:

All the thesis and dissertations of B.Tech, M.Tech and Phd submitted in the year 2007 and 2008 have been uploaded on the ethesis server and could be easily accessed from anywhere in the institute. Once the domain is open for public viewing these dissertations could be google searched through OAI-protocol for meta data harvesting and viewed from anywhere outside the campus.

## 5.4 Themes:

EPrints supports "themes" for the basic layout. Creating a few variations will make the software friendlier to people using it for demos and provide some alternate starting points for peoples look and feel. We can easily write out or edit the existing theme scripts and create new themes as we wish.

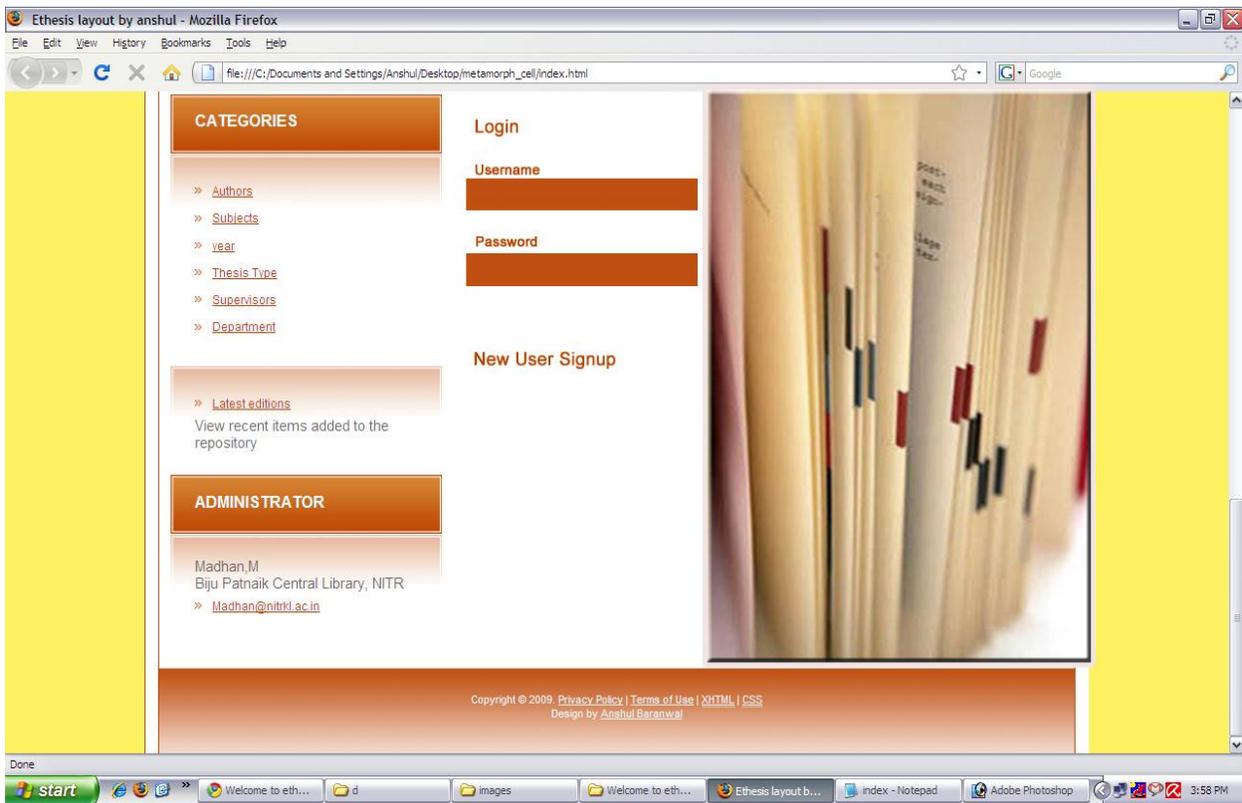
The proper customized themes could be written in xhtml and css with scripting of JavaScript. These edited theme codes can be uploaded in the server and the layout can be displayed directly by redirecting the theme page to this script.

The theme that we developed for the institute link of eprints i.e <http://ethesis.nitrkl.ac.in>

The screenshot shows a web browser window displaying the 'ethesis@nitr' website. The browser's address bar shows 'http://ethesis.nitrkl.ac.in/'. The website has a dark blue header with the text 'ethesis@nitr' and 'National Institute of Technology Rourkela'. Below the header is a navigation menu with links: Home, About, Year, Subject, Departments, Authors, Supervisors, Thesis Type. There is also a search bar and a 'Login | Create Account' link. The main content area is titled 'Welcome to ethesis' and includes links for Atom, RSS 1.0, and RSS 2.0. The content is organized into three columns. The left column contains links for 'Upload your thesis', 'ethesis@nitr is indexed by:' (with links to Google, OAlster, Scientific Commons, BASE, and other service providers), and 'Policies'. The middle column contains the main text: 'ethesis@nitr is the official institutional Open Access theses repository of National Institute of Technology Rourkela. Here all theses produced by students as a partial fulfillment of degree are uploaded. For more details: [more](#)'. It also lists the administrator: 'Madhan, M, Biju Patnaik Central Library, NITR' and email: 'madhan@nitrkl.ac.in'. The right column contains links for 'Search Repository', 'Browse Repository', 'Year', 'Creators', 'Supervisors', 'Departments', 'Subjects', and 'Latest Additions'. The footer includes the NIT Rourkela logo and 'Powered by eprints'.

**The basic layout currently displayed**

# The new layout designed by us for thesis site.



## **Conclusion:**

The Moodle and e-prints were successfully deployed in the institute and are proving to be useful for the NITRKL community in every possible way they can. Several novel changes were made in the codes and installation process to make them unique and most efficient. It was tried a lot to exploit the open source advantage of these two packages to the fullest. A number of new modules were added and developed by taking help from the various sources present. By working on these two things a deep understanding of the open source software development was observed. Moodle which works on PHP due to its open source background, a large number of new modules could be developed according to our needs and applied. The OAI-PMH which is a relatively new field which applies the open source philosophy in a new dimension of metadata harvesting for open archives which is an initiative to combine the scattered knowledge in the form of thesis and dissertations to be available openly to the people. This OAI-PMH is the sole concept behind development of eprints and it was found that still there is a lot that could be easily done with the open source advantage in our hand.

# References:

- 1- <http://docs.moodle.org>
- 2- <http://moodle.org>
- 3- [www.goohio.com](http://www.goohio.com)
- 4- [www.apache.org](http://www.apache.org)
- 5- [www.mysql.com](http://www.mysql.com)
- 6- [www.easyphp.org](http://www.easyphp.org)
- 7- [www.Sourceforge.net](http://www.Sourceforge.net)
- 8- [www.codeproject.com/](http://www.codeproject.com/)
- 9- <http://partners.moodle.com/>
- 10- <http://moodle.org/mod/data/view.php?id=6009>
- 11- <http://moodlemodules.netcourse.org/>
- 12- <http://moodlerooms.com/support/moodle-hosting/basic-hosting-services/>
- 13- <http://websites tips.com/php/resources>
- 14- <http://www.phparch.com/c/books>
- 15- <http://webdesigns1.com/php/>
- 16- <http://www.eprints.org/software/>
- 17- [http://wiki.eprints.org/w/EPrints\\_Manual](http://wiki.eprints.org/w/EPrints_Manual)
- 18- <http://www.openarchives.org/>
- 19- <http://www.openarchives.org/OAI/openarchivesprotocol.html>
- 20- <http://www.oaforum.org/english/page4.htm>
- 21- <http://roar.eprints.org/>
- 22- <http://drupal.org/project/oai2>
- 23- Introduction to the Perl Scripting Language, Norman Matloff,  
University of California, Davis,
- 24- <http://www.comptechdoc.org/independent/web/cgi/perlmanual/>
- 25- <http://www.w3schools.com>
- 26- <http://htmlcsstutorials.com/>
- 27- [http://wps.prenhall.com/chet\\_paynter\\_electronic2/23/6136/1570916.cw/index.html](http://wps.prenhall.com/chet_paynter_electronic2/23/6136/1570916.cw/index.html)
- 28- [http://wps.pearson.co.uk/ema\\_uk\\_he\\_story\\_electronic\\_3/42/10999/281599.cw/index.html](http://wps.pearson.co.uk/ema_uk_he_story_electronic_3/42/10999/281599.cw/index.html)
- 29- <http://www.learnwell.org/electro.htm>
- 30- <http://cobweb.ecn.purdue.edu/~ee255/practice%20exams/e2fa1101.pdf>
- 31- <http://nptel.iitm.ac.in/courses.php3branch=ece>
- 32- <http://www.science-ebooks.com/electronics/basic-electronics.htm>
- 33- <http://www.universalclass.com/i/crn/7550149.htm>