The ASF's Karst Index Database – Under the Bonnet

Author: Michael Lake

Abstract

The Australian Speleological Federation's Karst Index Database is a database of caves, maps, areas, organisations and people and is accessible on the Internet at http://www.caves.org.au. Updating has started in NSW by SUSS and in Victoria by VSA and to date there have been several hundred updates.

This paper will cover the technology that underlies the ASF's server and the KID, details on why we have two KIDs, the user and system documentation, installation and maintenance, inbuilt testing suites, source code management, backup and security. Future work will be suggested.

Introduction

The ASF's Karst Index Database (KID) contains information on 6600 caves and karst features, 2400 cave maps, 192 organisations, 1308 persons, references to 925 articles and covers 355 caving areas in Australia. Most of this information is accessable to anyone via a guest login.

Data on scientific items that may be in the caves, exact location information and some other information is restricted. For cavers or researchers that wish access to this information, a KID Access Group exists and application should be made to them. Details of this are on the ASF's web site.

The first read-only KID was written by a professional programmer engaged by the ASF in 2000 and went online in January 2001 as a read-only system. The programmer was contracted again over the period 2003 – 2005 to extend the functionality to include updating. The updating project took a long time to finish as the database itself together with the updating procedures required were quite complex. Because of this complexity the scope of the updating was reduced during the course of the contract to include updating of caves, areas, maps, organisations and people but not updating of articles.

Updating has started in NSW by SUSS and in Victoria by VSA and to date there have been several hundred updates.

The ASF KID Software Licence

The ASF's KID software is released under the GNU General Public License (GPL) [1]. One of the requirements of the ASF specification to the progammer was that the software used in the KID was open source and the KID code would be released under the GPL. This means that the the ASF did not need to purchase any software or software licences and we can use other people's high quality, open source code in our KID. Because the KID software is released under the GPL it's also available for other speleological groups and individuals to use and hopefully contribute to it.

A number of overseas groups have already downloaded the software or expressed interest in our code: the Database Commission of the Portuguese Speleological Federation, the Swedish Speleological Federation and a Russian Speleological Society.

The data stored in the database of course remains the property of the ASF and contributing clubs and is not subject to the GPL.

Summary of the KID Functionality

The following is a brief summary of the functionality of the KID user interface.

Searching: Searching for caves, cave maps, other maps, information on cave areas, organisations, people and articles related to caves.

Updating: Updating of caves, maps, cave areas, organisations and people via a web based interface. Checking of these updates via a *peer-review* process by other updaters for that cave area is done to minimise mistakes. Only after the checking step has accepted the data does it become visible to others viewing the data.

The history of changes can be viewed showing what was changed by who and when. In addition to activity reports for each updating organisation per month. Note however that this is simple sum of the insertions, deletions and changes made per month so no great emphasis should be placed on the value.

Documentation: Tutorials to assist updaters in using the KID are available on the documentation page in PDF format. This format was chosen for the tutorial so that they could be printed and used in training.

An Administration Page: This includes a web based interface for adding new updaters and setting their access levels. Access levels are quite fine grained: you can set the cave area, the individual cave fields they can read or write to, and the entities they can update such as caves, maps, areas, orgs or people.

The administration page also allows administrators to view any of the KID tables directly and view an audit trail of any update. From this page one can also view extensive and detailed documentation in HTML format of all the Perl modules that were developed or used in the KID.

The ASF Server

The KID runs on a virtual server which is running a Debian GNU/Linux distribution. A virtual server is an operating system hosted by a physical computer which runs multiple operating systems simultaneously. At the hosting provider the real, physical machine is running RedHat Enterprise Linux in a large data centre.

The ASF “owns” one of these virtual servers which is running our Debian Linux. Other organisations might own and run other virtual servers on the same physical machine. Each one is fully independent and appears as a separate server. Users or programs running under one virtual server cannot see or access the memory or files on another virtual server. We also have root access to our own virtual server which means that we have full control over our server; we can install any software we wish such as web servers and databases, add users and control access. The ASF uses a virtual server as this is much cheaper than a stand alone server.

Although we use a Debian Linux distribution, with minor changes to the installation procedure the KID will also install and run under any other Linux distribution or UNIX system such as Apple’s OSX.

The Web Server

The web server is the application that sends web pages from caves.org.au to a user’s web browser and accepts web forms submitted by a user’s browser. We are using the most popular web server in the world, Apache (http://www.apache.org). This versatile, high-performance web server is used by about 60 % of all web sites in the world [2]. It features a modular design and supports dynamic selection of extension modules at runtime. Some of its strong points are its security, range of possible customization, dynamic adjustment of the number of server processes, and a huge range of available modules including many authentication mechanisms, access control, caching and many more.

Whenever you attempt to access a web page in the KID, the Apache web server checks the KID database to see if you should have access to that page. If a username and password is required it will ask you for one. If you send an incorrect response or you shouldn't have access to that page, it will send back to you a short page telling you that the "server could not verify that you are authorized to access the document requested" and denying access. If your request for a page is accepted by the web server, it is passed to a programme (a Perl script) which will query the database, get the results back, format them into a nice web page and pass them back to the web server to be sent back to your browser.

The Programming Language

Next is the programming language. The KID uses Perl (http://www.perl.org) and it's these Perl programs that do most of the work. The Perl programs generate the web pages that are sent to the user when a user visits the KID website, take a users query and send it to the database and get the data back. They take that data and transform it into nicely formatted web pages and then hands the pages to the web server which then sends the page back to the user.

Perl is widely used for CGI scripts although the choice of this language more reflects the programmer's preference and what was mature and available in 2000 when the KID was first commissioned. Other programmers that tendered for the project would have used PHP, which was also widely used in 2000, or Java. Today Ruby or Python would be strong contenders. In hindsight Perl was an excellent choice; it has been very stable as it's a mature language, it runs fast, and we have been able to use the same language for the many command line scripts developed to support the KID. In contrast PHP has evolved considerably in the last several years and we would have had to do significant coding changes to keep up-to-date.

Thousands of small modules have been developed for Perl over many years and these can be used instead of writing your own routines (http://www.cpan.org). They are open source, well written and free. We are using modules that provide: authentication and authorization, reading config files and parsing command line arguments, interfaces for Perl to access databases, Comma separated values manipulation routines, SQL parsing and processing, routines for processing CSV files, interfaces for RSA and MD5 encryption, parsing and extracting information from HTML documents, APIs to the client side of various protocols such as email, ftp etc., persistence for Perl data structures, template processing and routines to produce PDF.

One very useful Perl module that we are using is "Template Toolkit" (http://www.template-toolkit.org). This neatly separates the HTML code for the user interface from the Perl code of the business logic and makes for a cleaner and more easily maintained site.

The Perl programming language has in our case a close connection with the web server. Our version of Apache has been compiled with mod_perl which embeds a copy of the Perl language interpreter into the Apache Web server (http://perl.apache.org). This, together with intelligent caching, results in a considerable speed increase of the Perl scripts. It can produce anywhere from a 400 % to 2000 % speed increase on sites using Perl scripts, and is used on many large script-based web sites like http://slashdot.org.

We are not using any web application development frameworks. These are very popular today, particularly the Model-View-Controller approach. Examples include Ruby-on-Rails, Turbogears for Python, Catalyst for Perl and several others. At the time the KID was started these either didn't exist or were just being developed. They can certainly speed up development and provide an excellent separation of the database, the HTML presentation layer and the busines logic of the application. If the KID were being started today we would consider using such frameworks.

The Database

At the lowest level of the KID is the database that stores the data on the caves, maps, organisations, people and cave areas. The database the KID uses is MySQL (http://www.mysql.com). MySQL is a fast, lightweight and full-featured database which is also reasonably easy to administer. This database also stores the user access privileges of the users such as the guest user and the updaters from various clubs. There are over 500 fields structured into 74 tables. Most of these are the same as the draft table schemas and database fields published by the Informatics Commission of the International Union of Speleology [3].

Two KIDs

There are actually two Karst Index Databases; a production one and another one used for testing and development. They have separate code bases and backend databases. The KID accessed at http://www.caves.org.au/kid uses the production code and the production database and is the one used by guests using the ASF KID and by updaters updating the KID. The KID at http://www.caves.org.au:8080/kid uses the development code base and a development database (the :8080 means Apache is serving the pages from port 8080 instead of the default port 80). It is used by programmers for testing code changes on the database. If a coding mistake is made in development of new features it will not affect the production server. It is also used by updaters as a "play" database where they can familiarise themselves with the updating system without worrying about making incorrect changes to the production database. Guests can also access the test KID.

At some point in time a decision is made to release the development version to production. This is usually when sufficient changes have been made to the development KID and it is deemed stable and there are no known bugs. When this occurs the current production system is moved away and the development system is put in its place. The system is given a quick final test and the web server is restarted. During this brief time the web site may be down for a few seconds.

You can find the software version listed on the Help page. The software version of the production KID is always an even number, e.g. version 1.32, and the version of the development KID is always an odd number, e.g. 1.33.

User Documentation

On the documentation page at http://www.caves.org.au/kid/doc are two sections: "Documents for Updaters" and "ASF KID Specifications".

The "Documents for Updaters" section consists of a general introduction and a series of tutorials on how to update people, caves, add new areas and organisations and a description of access control. These are in PDF format so they can be printed out and referred to easily. The PDF documents are written using LaTeX – a high level typesetting language which is designed for the production of high quality technical documents [4,5]. A LaTeX document is written in plain text with markup that describes bold face, italics, tables, lists etc. This plain text is then compiled into PDF output. LaTeX is very nice to use; it is mature and produces output of the the highest typographic quality.

The "ASF KID Specifications" section contains complete documentation in HTML format for all the database fields used in the KID and the database table schemas. These HTML pages are produced on-the-fly from the KID database itself. The advantage of this is if the database is changed, this part of the documentation is automatically updated. These fields and their definitions are based on the field definitions defined by the Informatics Commission of the International Union of Speleology, of which the ASF is a member.

System Documentation

Detailed installation and maintenance documentation is available in HTML format on the software page at http://www.caves.org.au/kid/docs/software. This describes how to install the KID, how to make updates and releases and general maintenance of the system.

For every Perl module that the programmer has written the object classes and methods are documented in the module code itself. The system has scripts which generate HTML formatted documentation from the module documentation in a standard format for all modules. Other third party Perl modules which we have used also have their HTML documentation in the same format on that page.

Inbuilt Regression Testing

The KID also has some inbuilt regression tests available to test that code changes have not broken the system. Before a code change the output pages of several searches can be saved. After code changes the search pages afterwards are compared to the ones before. If changes to the search pages are not expected the test suite should report that the files match.

Regression testing is also used to check the CSV (comma separated variable) upload functionality which allows a user to upload bulk cave updates. This is essential for testing the bulk uploading as there are many errors that can occur in a bulk file generated or edited outside of the KID system. For instance in a CSV upload a record in the file may have unbalanced quotes, a field may be missing or an extra field may have been inserted. For each possible error that could occur is a test file with that error. During the test validation each of these files is loaded and the test suite should flag an error.

Source Code Management

All the KID software is managed by a SCM (Source Code Management) system. We are using Subversion (http://subversion.tigris.org) which is rapidly becoming the most widely used SCM in the open-source world. It is a "work-alike" replacement for the older SCM system called CVS.

A SCM system allows provides several benefits; it allows many developers to work collaboratively on the same code, tracks which changes have been made by which developers. tracks history of changes and it allows one to revert to any previous version. For instance, I can work on the KID code on my laptop whilst on the train for a few days and later connect to the ASF server and commit my changes to the test KID.

Backups

Backups of the data and the KID software are critical in case there is a hardware problem with our server. I have setup a backup script that runs automatically as a cron job at about 4 am Sydney time each week. This sends a copy of the current and software to a site in the United States (kindly provided by the Informatics Commission of the IUS). Also included with the software and data is documentation to assist a System Administrator in a recovery procedure.

The following is done each time the backup script runs:

1. Backup files more than 6 months old are removed so the server will not fill up.
2. The current KID software is copied to a backup directory if no current backup file exists.
3. The latest installation guides and disaster recovery documents are also copied to the backup directory.
4. The Update Home Page is replaced with a temporary page preventing new updates and checkouts. The script needs to stop and restart the web server as the Update Home Page will be cached in memory. This may result in the ASF Web site being "down" for a few seconds.
5. A dump of the entire database is performed, compressed and placed in the backup directory.
6. The Update Home Page is restored.
7. Logs of the above operations are written to the backup directory.
8. A checksum is generated for the backup files.
9. The script then securely copies all these backup files to the remote servers.
10. Finally an email is sent to the KID System Administrator to indicate if the backup was successful or not.

An example of the email that the KID System Administrator receives is below.

```
Subject: ASF KID Backup successful
From: kid@www.caves.org.au
To: mikel@speleonics.com.au
Date: 1 Jan 2007 04:00:44 +1100

Hi
The KID backup script at caves.org.au successfully made a backup of database kid on 2007.01.01.
There is a checksum asf-kid-data-v1.32-2007.01.01.cksum with the data file that can be used to
check that data transfer was successful.

Admin, caves.org.au
```

Security

There are two main security concerns for the ASF server. First the KID contains some sensitive information such as records of aboriginal remains in some caves or, in the future, exact location information. We would not like this information to be stolen. Secondly is that we don't wish the server to be compromised and a trojan installed or an extra login account created. Such trojaned servers are refered to by crackers as being "0wned". They can be used to mount subsequent attacks against other servers such as government or military sites.

Security was considered by the web programmer during the coding stages and a number of measures taken in the KID code. For instance, all user input into the KID is filtered to guard against SQL injection attacks. Additionally the following policies are adhered to on the ASF server:

- Minimal services are run on the ASF Server.
- The server is kept up-to-date with security patches.
- Intrusion detection systems are used. These compute multiple checksums for important files on a system.
- All login access is via secure shell version two and the use of public/private keys.

Future Directions

Generate Information for use on PDAs

Already work has started to generate summary information for all caves in an area for use on PDAs by updaters. The idea is that an updater could request the KID to generate summary information, similar to the simple and standard search format, but for all caves in an area and not just a small range of caves. This data would be zipped into a single file which the updater could download and transfer to their PDA. There would be no logo or large headings as the format would be designed to be used on the small screens of PDAs. Such information would be useful in the field, particularly if exact location information was included.

Karst Index Books in PDF Format

The last hard copy version of the KID was the "Australian Karst Index 1985" [6]. The cost of publishing today would preclude such a publication however one could produce of PDF book of karst features for a State. They would be similar in format and layout for each State, thus forming a collectable series, and could be produced every six months.

Internationalisation

The software currently uses the English language. Future versions of the software should be internationalised so that it can be easily adapted by other countries. The KID already has excellent separation of the Perl code from the HTML markup so redesigning the user interface for other speleo groups is not too difficult. However there are areas of code such as error messages which will present some problems, especially for countries that use non Latin character sets.

The Next Generation KID

Software ages fast and the KID is no exception. There are already technologies that would be useful in the KID such as AJAX (Asynchronous Javascript and XML) and it's likely that we will wish to add some GIS capabilities and connectivity to other databases in the near future. Also as updaters from more clubs use the KID they will develop ideas on how the software and the updating process could be improved. New ideas can't implemented as soon as users would like as we have limited funds and volunteer time however we should certainly think about how and in what direction the KID will evolve – or what we will replace it with.

References

[1] GNU General Public Licence, http://www.gnu.org/licenses/licenses.html

[2] Netcraft, http://news.netcraft.com

[3] Informatics Commission of the International Union of Speleology, http://www.uisic.uis-speleo.org

[4] LaTeX Project Site, http://www.latex-project.org

[5] LaTeX Wikipedia entry, http://en.wikipedia.org/wiki/LaTeX

[6] Australian Speleological Federation Inc., “Australian Karst Index 1985”, Edited by Peter G. Matthews

ISBN 0 9588857 0 2

Author Contact Information:

Michael Lake
Thornleigh NSW 2120
Sydney, Australia
email: MikeL@speleonics.com.au