

Troggle: a novel system for cave exploration information management

Aaron Curtis, Cambridge University Caving Club
aaron.curtis@cantab.net

Why cavers need effective data management

Cave exploration is more data-intensive than any other sport. The only way to “win” at this sport is to bring back large quantities of interesting survey, and possibly photos or scientific data. Aside from the data collection requirements of the game itself, setting up a game (an expedition) of cave exploration often involves collection of personal information ranging from dates available to medical information to the desire to purchase an expedition t-shirt.

If an expedition will only happen once, low-tech methods are usually adequate to record information. Any events that need to be recorded can go in a logbook. Survey notes must be turned into finished cave sketches, without undue concern for the future expansion of those sketches.

However, many caving expeditions are recurring, and managing their data is a more challenging task. For example, let us discuss annual expeditions. Every year, for each cave explored, a list of unfinished leads (which will be called “Question Marks” or “Qms” here) must be maintained to record what has and has not been investigated. Each QM must have a unique id, and information stored about it must be easily accessible to future explorers of the same area. Similarly, on the surface, a “prospecting map” showing which entrances have been investigated needs to be produced and updated at least after every expedition, if not more frequently.

These are only the minimum requirements for systematic cave exploration on an annual expedition. There is no limit to the set of data that would be “nice” to have collected and organized centrally. An expedition might collect descriptions of every cave and every passage within every cave. Digital photos of cave entrances could be useful for re-finding those entrances. Scans of notes and sketches provide good backup references in case a question arises about a finished survey product, and recording who did which survey work when can greatly assist the workflow, for example enabling the production of a list of unfinished work for each expedition member. The expedition might keep an inventory of their equipment or a catalog of their library. Entering the realm of the frivolous, an expedition might store mugshots and biographies of its members, or even useful recipes for locally available food. The more of this information the expedition wishes to keep, the more valuable an effective and user-friendly system of data management.

Evolution data management on the CUCC Expeditions to Austria

Over 32 years, CUCC has developed methods for handling such information. Refinements in data management were made necessary by improved quantity and quality of survey; but refinements in data management also helped to drive those improvements. The first CUCC Austria expedition, in 1976, produced only Grade 1 survey for the most part (ref Cambridge Underground 1977 report). In the 1980s, the use of programmable calculators to calculate survey point position from compass, tape, and clinometer values helped convince expedition members to conduct precise surveys of every cave encountered. Previously, such work required hours of slide rule or log table work. On several expeditions, such processing was completed after the expedition by a FORTRAN program running on shared mainframe time. BASIC programs running on personal computers took over with the release of the BBC Micro and then the Acorn A4. In the 1990s, Olly Betts and Wookey began work on “Survex”, a program in C for the calculation and 3-D visualization of centerlines, with intelligent loop closure processing. Julian Todd's Java program “Tunnel” facilitated the production of attractive, computerized passage sketches from Survex centerline data and scanned hand-drawn notes.

Along with centrelines and sketches, descriptions of caves were also affected by improvements in data management. In a crucial breakthrough, Andrew Waddinton introduced the use of the nascent markup language HTML to create an interlinked, navigable system of descriptions. Links in HTML documents could mimic the branched and often circular structure of the caves themselves. For example, the reader could now follow a link out of the main passage into a side passage, and then be linked back into the main passage description at the point where the side passage rejoined the main passage. This elegant use of technology enabled and encouraged expedition members to better document their exploration.

To organize all other data, such as lists of caves and their explorers, expedition members eventually wrote a number of scripts which took spreadsheets (or comma separated value files, .CSV) of information and produced webpages in HTML. Other scripts also used information from Survex data files. Web pages for each cave as well as the indexes which listed all of the caves were generated by one particularly powerful script, make-indxa4.pl . The same data was used to generate a prospecting map as a JPEG image. The system of automatically generating webpages from data files reduced the need for repetitive manual HTML coding. Centralized storage of all caves in a large .CSV file with a cave on each row made the storage of new information more straightforward.

Another important element of this system was version control. The entire data structure was stored initially in a Concurrent Version System (CSV) repository, and later migrated to Subversion (SVN). Any edits to the spreadsheets which caused the scripts to fail, breaking the website, could be easily reversed.

However, not all types of data could be stored in spreadsheets or survey files. In order to display descriptions on the webpage for an individual cave, the entire description, written in HTML, had to be typed into a spreadsheet cell. A spreadsheet cell makes for an extremely awkward HTML editing environment. To work around this project, descriptions for large caves were written manually as a tree of HTML pages and then the main cave page only contained a link to them.

A less obvious but more deeply rooted problem was the lack of relational information. One table named `folk.csv` stored names of all expedition members, the years in which they were present, and a link to a biography page. This was great for displaying a table of members by expedition year, but what if you wanted to display a list of people who wrote in the logbook about a certain cave in a certain expedition year? Theoretically, all of the necessary information to produce that list has been recorded in the logbook, but there is no way to access it because there is no connection between the person's name in `folk.csv` and the entries he wrote in the logbook.

The only way that relational information was stored in our csv files was by putting references to other files into spreadsheet cells. For example, there was a column in the main cave spreadsheet, `cavetab2.csv`, which contained the path to the QM list for each cave. The haphazard nature of the development of the "script and spreadsheet" method meant that every cave had an individual system for storing QMs. Without a standard system, it was sometimes unclear how to correctly enter data.

What's different about Troggle?

Troggle is indeed another in a long series of paper-based and computerized utilities CUCC has developed to make our expedition run more smoothly. Yet, Troggle is revolutionary. For the first time, we have a unified system which can manage any type of expedition data, and understands the relationships between the data. It is the first time we have a system which we can recommend to other expeditions.

Using a relational database, Troggle represents familiar expedition concepts such as the cave, the person, and the logbook entry in tables. All of the links which exist between these concepts in real life, such as the fact that logbook entries are always written by people and can sometimes refer to caves, exist in the database as foreign keys between models. A slightly simplified graph of the models used and linkages between them is shown in figure 4.

Troggle: seen from the driver's seat

All of the data in the Troggle database is created, viewed, and edited through a web browser. However, by no means does Troggle require the internet to function. A Troggle server¹ is a lightweight program which can be run on any computer with Linux, Mac, Windows, or many other operating systems. If run on an expedition computer connected to a wired or wireless LAN, it could be simultaneously used on the expedition computer as well as any number of laptops expedition members may have brought with them. Troggle can also be run on a dedicated server used as an expedition webpage handling high volumes of traffic. Webpages based on similar technology can serve as many as 500,000 pages an hour². For this discussion, let us assume that Troggle is hosted by a server at `www.cavingexpedition.com`.

Troggle's URLs are set up so that things are easy to get at. There is no need to send the browser to a certain file as you would with a conventional webpage; in fact the file you are looking for is not written until you ask for it. If I wanted to access a person page for Joe Smith, I could type in any of the following, and they would bring me to his profile page:

- `www.cavingexpedition.com/person/joe_smith`
- `www.cavingexpedition.com/people/joe_smith/`
- `www.cavingexpedition.com/person/JoE_sMith`
- `www.cavingexpedition.com/people/Joe Smith`
- `www.cavingexpedition.com/person/JoeANYtHiNg123!@#HERESmith`

If I typed

- `www.cavingexpedition.com/person/joe`

then Troggle would present me with a list of all of the people named joe. For a list of all people, I would go to

1 either the Django development server or any webserver with `mod_python` set up to work with Django

2 For more about scalability of the Django platform which Troggle uses, see <http://www.davidcramer.net/other/43/rapid-development-serving-500000-pageshour.html>

www.cavingexpedition.com/person/ or www.cavingexpedition.com/people/. Similar flexibility exists for looking up expeditions, caves, logbook entries, and other objects.

A new expedition member's first encounter with Troggle might be when uses it to sign up for the expedition, by going to www.cavingexpedition.com/register/. Troggle uses the familiar 'activation email' strategy of verifying user email addresses where a user is sent an activation key link which they click on or paste into their browser. Once registered, the user is asked to fill out information which the expedition needs about them: what dates they will be available, next of kin, medical conditions, etc. The user is also asked if they have attended previous expeditions, in which case they will be asked for their name and the existing information about them will be attached to their user account. Once confidential information such as medical conditions is stored in the database, access to it is restricted to certain users.

Instantly, this becomes useful information for the expedition leader, who is planning the expedition and wants to know how many people will be present for each day of the expedition. When the user submits their available dates, the calendar for that expedition (figure 1) appears updated. In actual fact, nothing is updated except for a cell in the database's Person table, because the calendar does not exist in a file; it is generated dynamically each time a user loads it.

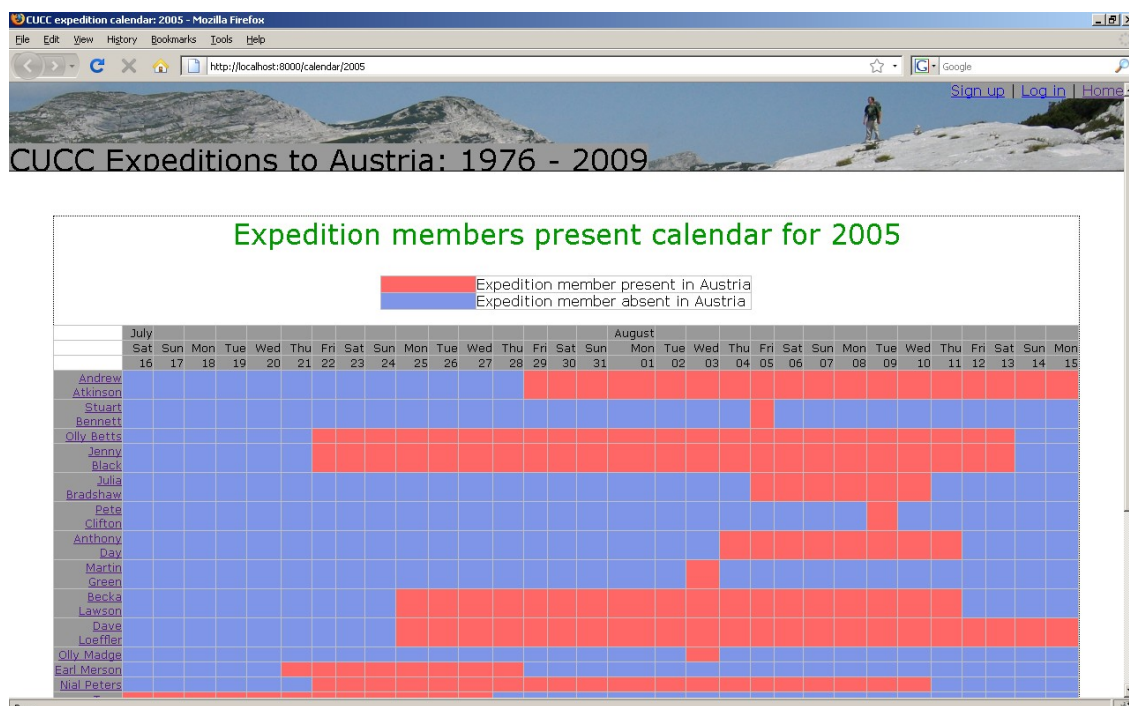


Figure 1: The automatically updated expedition calendar. For past expeditions such as 2005 (shown) dates present for people were inferred from the existence of logbook entries, and are therefore very approximate. From 2009 on, dates will be those specified by the expedition members themselves when signing up for the expedition. Clicking on a person's name links to information about each person.

On the expedition itself, Troggle is used to record each trip. In CUCC's case, an expedition member will typically spend four or five days at top camp caving before she hikes back down to base camp. When she arrives in base camp, she enters information about the trips she was on: who was on each trip, where the trip went, dates, which going leads (QMs) are no longer going, which new going leads need to be added to the database, and whether the trip brought back any new survey. The information she adds immediately becomes accessible from many different parts of the site. If she writes a trip description, it shows up in the logbook for that year (figure 2). On the page for the cave she visited, her trip will appear in the list of trips to that cave, and her QMs appear in the list of dead and going leads, respectively.

Now the survey work begins. If she stated that her trip produced survey, Troggle will send her to its Virtual Survey Binder, www.cavingexpedition.com/surveys/ (figure 3). This page guides her through the expedition workflow which will eventually end in integrating her new information into the final survey. Our caver can take a break for a beer or five at any point during this process, and Troggle will keep track of what needs to be done. For each year, such as www.cavingexpedition.com/surveys/2008, it displays a table with each section of survey and what workflow stages have been completed.

Expedition CUCC expo2008 - Mozilla Firefox

http://www.cavingexpedition.com/expedition/2008

CUCC Expeditions to Austria: 1976 - 2009

CUCC expo2008: 2008-07-18 - 2008-08-11

Logbook entries

Date	Title	Author	Place
2008-07-18	Journey to Austria - The Van limps to Austria	Kathryn Hopkins	Journey to Austria
2008-07-18	Journey to Austria - From Artic to Austria	Nial Peters	Journey to Austria
2008-07-23	Hauchhole - rigging	Jess Stirrups	Hauchhole
2008-07-23	204 - High hopes rigging	Edvin Deadman	204
2008-07-24	204 - High hopes pushing	Martin Jahnke	204
2008-07-24	204 - Walk to top camp and Keith's first trip to 2	Keith Curtis	204
2008-07-25	204 - Keith's second underground trip	Keith Curtis	204
2008-07-26	Journey to Austria - Trip to Expo	Steve Jones	Journey to Austria
2008-07-27	204 - Tourist trip down 204E	Kathryn Hopkins	204
2008-07-28	204 - Chocolate Salty Balls pushing	Kathryn Hopkins	204
2008-07-28	Tunnocks - 07-49A	Becka Lawson	Tunnocks

Caver	From	To
Edvin Deadman	2008-07-18	2008-08-03
Jess Stirrups	2008-07-23	2008-08-04
Frank Tully	2008-07-26	2008-08-07
Nial Peters	2008-07-18	2008-08-11
Steve Jones	2008-07-26	2008-08-07
Ollie Stevens	2008-07-28	2008-08-09
Martin Jahnke	2008-07-24	2008-08-04
Tony Rooke	2008-07-18	2008-07-24
Julian Todd	2008-07-26	2008-08-05
Pete Harley	2008-07-18	2008-07-30
Becka Lawson	2008-07-28	2008-08-09
Keith Curtis	2008-07-24	2008-07-25
Olly Madge	2008-07-30	2008-08-03
Serena Povia	2008-07-30	2008-08-07
Mark Shinwell	None	None
Eeva Makiranta	2008-08-09	2008-08-11
Aaron Curtis	2008-07-24	2008-08-04
Kathryn Hopkins	2008-07-18	2008-08-03
Djuke Veldhuis	2008-07-18	2008-08-04

Figure 2: The expedition index page, with links to pages for related logbook entries, caves, and people.

CUCC Expeditions to Austria: 1976 - 2009

Choose a year: 2008

✓ SURVEY PROGRESS

TABLE

Choose a wallet number: 2008#24

✓ SCANNED NOTES

SURVEY FILE

PRINTED CENTRELINE

✓ SCANNED PASSAGE

SKETCH


TUNNEL XML FILE

ADD TO MAIN SKETCH

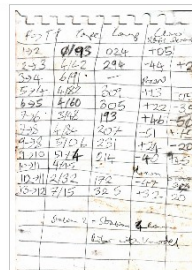
Survey progress table for 2008

	1	2	3	4	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	28	32	33	
Notes	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Survey file																												
Plans	✓	✓	✓	✓	✓							✓	✓	✓								✓	✓	✓	✓	✓	✓	
Elevations	✓	✓	✓	✓	✓																		✓	✓	✓	✓	✓	✓

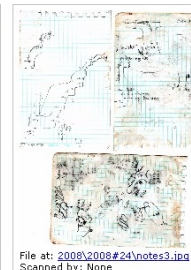
Scanned notes for 2008#24.



File at: 2008\2008#24\notes1.jpg
Scanned by: None
On: None




File at: 2008\2008#24\notes2.jpg
Scanned by: None
On: None



File at: 2008\2008#24\notes3.jpg
Scanned by: None
On: None

✓ Add a new scanned notes page.

Scanned plan sketch files for 2008#24.



✓ Add a new scanned sketch.

Figure 3: Troggle's "Virtual Survey Binder." Check marks show items which the database knows are present in the file structure. Different items can be toggled on and off using the left hand navigation menu

Troggle: under the hood

What makes all of this magic possible? Troggle is a “Django project.” This means it is a set of data models, views, and templates (also parsers to convert from the legacy data structure) to be used with Django, a web framework for the programming language Python. Django represents all of the data models as tables in a relational database and automatically creates and sends SQL commands to control that database. (Currently, Troggle is being developed on MySQL, but switching to another database backend is as simple as changing a variable in a configuration file).

When a user requests a URL, and it is handled by Django (if using Apache, `mod_python` passes the request to Django, other webservers have analogous modules), Django checks the incoming URL against a list of python regular expressions. Troggle's flexibility with URLs comes from the fact that we have written very loosely matching regular expressions. Once Django finds a regular expression which returns a match for the URL, it runs a python function (a “view”) which takes the URL, draws relevant information from the database, and uses it to fill in a template written in the django template language, producing a HTML page for display user.

Database access is facilitated by the powerful Django Object Relational Mapper (ORM), accessed through a Python application programming interface (API). Each data model, such as the “Cave”, “Person”, or “Logbook Entry” model, is accessible as a Python object with useful methods that allow powerful searching, editing, and saving of the database from python. Here are some lookup scenarios, ranging from the simple to complex, with the necessary python code.

- Get a list of all people who have been on the expedition and store it as `result`

```
result=People.objects.all()
```
- Get a list of all people who were on the 2004 expedition and store it as `result`

```
result=Expedition.get(year=2004).people_set.all()
```
- Get the text of the logbook entry with “intrepid voyage” in the title and store it as `result`

```
result=LogbookEntry.objects.get(title__contains="intrepid voyage").text
```
- Get a list of all photos taken in a cave whose official name begins with “Stein” in the year 2000, and store it as `result`

```
result=Photo.objects.filter(cave__official_name__startswith="Stein")
    .filter(date__year=2000)
```

Of course, the API is not accessible through a web browser, and Troggle users should never need to query the database directly using the commands in the example above. However, the API is what drives the views and templates and is therefore used whenever a page is loaded.

Tailoring Troggle to your needs

Three aspects of Troggle's design make it highly customizable. First, all paths and URLs are relative to variables in a settings file. Second, because Troggle takes advantage of Django's powerful template inheritance, one HTML file and one css file control the appearance and style of the entire website. Third, Troggle's modular structure allows the addition of alternate or new components.

The locations of files and other aspects of in system setup are set as variables for using the file “`localsettings.py`”. This file also contains all of the information necessary to access the database Troggle will use. Changing the variable `DATABASE_ENGINE` allows the use of different database systems; Postgresql, Mysql, Sqlite3, and Oracle databases are supported. The settings which allow troggle to send email, necessary to verify users who are setting up accounts, are also in the same file, and allow any STMP mail server to be used. This allows troggle to send emails from almost all commercial email services, as well as progressive free email services such as Gmail, because these email services provide STMP access.

To customize the appearance of the Troggle website as presented to users, it is generally sufficient to modify the file `base.html`, and the Cascading Style Sheet which it references. In Django terms, all of the other templates “extend” `base.html`. The other templates only serve to replace “blocks” in `base.html` with content. Table FIXTHIS shows the various blocks with a description of what they represent. The location of `base.html` and all other template files is specified in `localsettings.py`, in the variable `TEMPLATE_DIRS`, which is a tuple of strings that are absolute paths to the directory or directories where templates are stored. The stylesheet used by `base.html` is stored under the “`css`” directory of the path specified in `MEDIA_ROOT` and must be served at the URL specified in `MEDIA_URL`.

Block name	Purpose
title	The title of the page, to be displayed at the top of the browser
head	Holds elements that need to be inside the HTML <head> tag, such as metadata and scripts
logininfo	This is used to display “welcome,” “log in,” or “sign up” links. Should not be replaced by “extending” templates.
nav	Used for child templates, such as the “virtual survey binder,” that include a lefthand menu
contentheader	The header for the actual information being displayed
content	The actual information being displayed
footer	Content at the footer of the page. Should not be replaced by “extending” templates.

Table 1: Template blocks specified in base.html , which provide the framework for inserting content into Troggle's web interface.

Custom modules are easy to add to Troggle. For example, CUCC's troggle installation has a folder called `Parsers` which contains scripts for importing data from our legacy data structure (the system of csv tables described above) into the database using the Django API. If your expedition has a similar data structure to import from, these parsers can be replaced with your own. One of the parsers, `survex.py`, imports from Survex files. To use survey data in other formats, such as Compass or CMAPS, a new parser would need to be written.

Acknowledgements

Troggle began as a joint effort between Martin Green, Julian Todd, and myself. For years, there had been discussion regarding a new paradigm for data management on the expedition, and it was Martin who introduced the idea of a Django project. Julian Todd's web programming and scraping experience was invaluable for many of the views and the logbook parsing. Logbook formats for each year of the expedition are different, but thanks to Becka Lawson's hard work cleaning up and standardizing past logbook files and Julian's year by year parser coding, we now have over a decade of trips in the database. I would also like to thank Vic Brown and Andrew Waddington for their helpful accounts of the data management techniques they used on early CUCC expeditions, going back all the way to 1976.

