u3a Research Database

Background

The u3a research database is now found at u3aresearch.uk. The database and web interface resulted from a project begun in 2015 by members of Malling u3a in conjunction with the national Research and Shared Learning Committee. A work-in-progress version of the site, partially populated with data on known research and shared learning projects, was presented to the Research and Shared Learning Committee in November 2015 and after amendment and further population of data, presented to a meeting in London in March 2016. It was then officially launched with the domain u3aresearch.org.uk and the database and software for the interface moved to a server provided by u3a head office.

Projects featured

The categories of project (one of which must be selected when a new project is entered) are:

- u3a-led research
- u3a shared learning project
- External research with u3a assistance
- External research with u3a members or groups as research subjects
- Research about u3a

Projects can also be “tagged” as relating to one or more of a longer list of topics to assist searching and grouping. The list of topics can be expanded as necessary.

Addition and editing of project information

A user interface for logging in and editing has been a feature of the site from the beginning. The intention was for adding and editing to be done by regional and local representatives (u3a members) who would be in contact with projects in their area. This has happened to some extent but the response has been patchy and in practice most projects have been added by Jo Livingston, a member of the Research and Shared Learning Committee and a small number of local representatives. Around 40 u3a members have been granted editing credentials since 2016 but most of these are no longer active. The total number of projects registered on the database is now (February 2024) 1084.

Technical background

The original version of the site was built with a single file sqlite database, the original batch of data being imported via csv files from existing spreadsheets, with the code for the interface, both public pages and the user authentication and editing, being written in PHP using the PHP data object (PDO) for connection to the database and basic data representation. The design was database led, the process being first to structure the database to accommodate the information available and the anticipated uses (storage, searching, browsing, adding and amending) and then to construct the interface from the bottom up to enable public viewing, searching and browsing, and editing by authenticated editors. Apart from the PDO itself, the code included a single PHP class templating an object for each project; otherwise, the code was procedural rather than object-oriented.

The PDO was used for database connection and data representation as it offers greater

security than older methods and facilitated switching to a different database if desired with minimal changes to the PHP code.

There was some discussion about which database to use, one member of the team advocating use of a server-based rather than single-file database, but it was agreed to stick with sqlite at least for the earlier phases of the project, retaining the option to move to a server-base database at some stage in the future. This hasn't yet happened. The main limitation of sqlite would become apparent if the volume of use made it likely that concurrent commitments (resulting from editing or adding, not just reading) conflict. At the current or any likely foreseeable level of usage, this is highly improbable, and meanwhile the greater flexibility and transferability of sqlite provide strong arguments for staying with it.

Use of javascript has been very limited, within the philosophy of progressive enhancement, providing core functionality through server-side code and only, for example, highlighting of results of searches through client-side javascript. This remains the case with version 2 of the site, referred to below.

Developments since 2016

Since launch, the database has gradually been expanded by the addition of further projects. The main change in the public facing pages of the website has been the adoption of a responsive design enabling satisfactory viewing of the site on a much wider range of devices, including mobile phones, without switching users to a separate version of the site. This has been made necessary by the increasing use of mobile devices to view websites and made possible by developments in CSS and the ability of modern browsers to interpret html and css consistently.

The interface has worked successfully and proved resilient to achieve its modest initial objectives. The main challenge with maintenance has been keeping access to the server used by u3a HQ and needing to intervene when services or certificates have been allowed to lapse. Keeping the database alive has not been a priority for the staff at HQ, who have had much bigger challenges to their resources, especially during the last couple of years. After the hosting changed and then lapsed for the database files in 2023, I set up the alternative domain name u3aresearch.uk and used copies of the scripts that I had and a recently backed up copy of the sqlite file to set up the site again on the server on which it had been originally developed (the provider is alwaysdata, a company based in France that I have used for a number of years).

Because I had not had access to the files on the HQ server for most of 2023, I had a backlog of "backend" updates that I wanted to make, and could now do this. Rather than make the updates to the existing PHP code, though, I decided to take the opportunity to switch the interface to one using the Django development platform. This is partly due to a subjective preference to use Python (on which Django is based) as a general scripting/programming language rather than keep using PHP, when this database had become the only project on which I was using PHP. I feared it would become more difficult and inefficient to keep up to date with developments in PHP, particularly in respect of security. Objectively also, the admin interface provided as part of the Django platform is a good, reliable and secure fit for enabling the editing of the site. There were a number of reasons why I did not use Python and Django in 2015 which no longer applied:

- Difficulties with low-cost deployment through FastCGI, which was the only route I had

available at that time. Deployment through WSGI is now widely available, including within low-cost hosting services.
    - Python was going through a backwards-incompatible migration from v2 to v3 that caused complications within the language itself and for platforms like Django built on it.
    - Because of these issues and others, the long-term future of Django was not sufficiently clear. It is now more mature as a platform with well-established long-term support through its community (which has grown significantly since then) and the foundation set up to support it.

Another reason for originally using PHP was that I believed at the time it would be better for "succession planning" for someone else to take over the running of the site, as there would likely be more people with knowledge of PHP. On the other hand, though there would be a specific requirement for someone who knows Python and Django, that actually makes the "job description" very clear, and for anyone who meets that spec, the site as it is now set up would be very straightforward. It is simple in construction and sticks to mainstream conventions for use of the platform. Code written in PHP without a platform is inevitably more or less idiosyncratic depending on the original author (as the history of the Beacon project shows, though the Beacon code is many times more complex due to the much greater range of functionality).

When originally launching the database site I stressed that, for this project, the maintenance of the database was the most important issue. Setting up a public and user interface through a website is not trivial but could always be re-done, and anyone with the knowledge and skills to take over would likely have their own preference for how to do it. The exercise of migrating from PHP to Django is itself a demonstration of this.

As well as the ready-made admin interface, Django provides a robust basis for object-relational mapping so that, rather than starting with the database being manipulated directly through SQL queries, Python "models" can be used both for originally conceiving the data structures and also for manipulating and retrieving data used on the site. This is more efficient than using bespoke PHP code either through further bespoke classes and objects or through thinly wrapping SQL queries in PHP. The migration did involve an initial challenge where the database had already been constructed: this involved, as a one-time exercise, going backwards from the SQL structures to set up the Python models, but this was not particularly difficult and was in itself a useful learning exercise.

The intention always was that the research database would itself be a "shared learning" project, listed (recursively?!) as project 410 in the database, one of whose uses should be to share knowledge about the techniques used in its development.

Please let me know if you have comments or questions.

David Nicholls
23 February 2024