Assignment 2, Part B: Event Logging

(5%, due 5:00pm Wednesday, October 25th)

Overview

This is the second part of a two-part assignment. This part is worth 5% of your final grade for IFB104. Part A which preceded it was worth 20%. This part is intended as a last-minute extension to the assignment, thereby testing the maintainability of your solution to Part A and your ability to work under time pressure. If you have a neat, clear solution to Part A you will find completing Part B much easier. For the whole assignment you will submit only one solution, containing your combined response to both Parts A and B, and you will receive one grade for the whole 25% assignment.

Goal

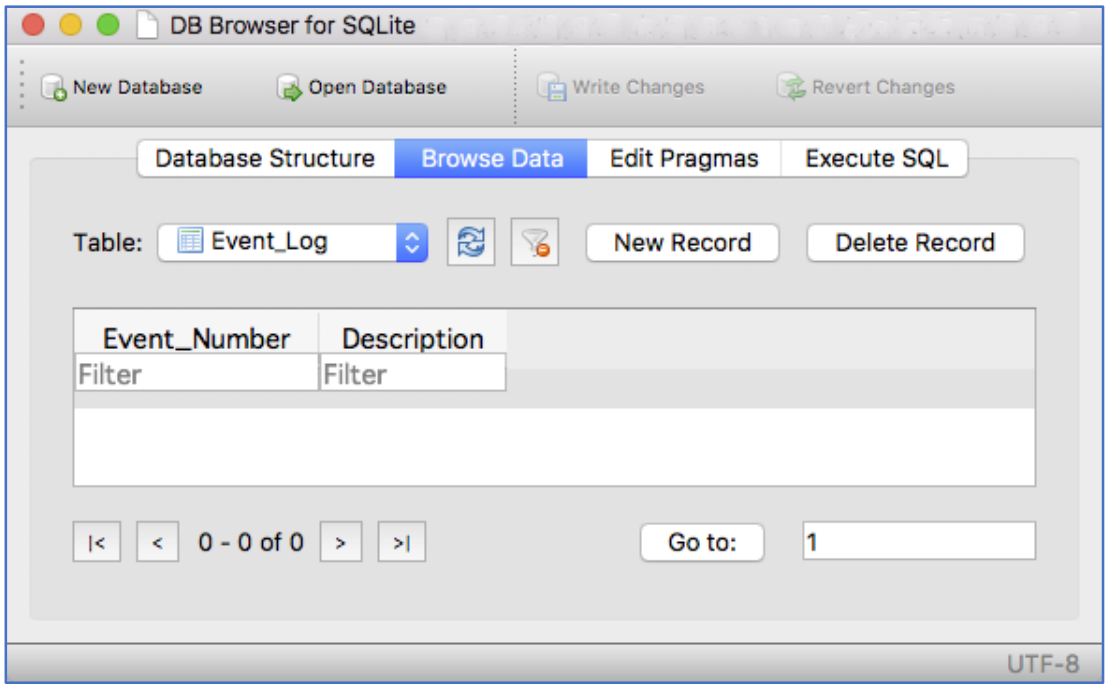
In Part A of this assignment you built a significant "IT system" for maintaining an archive of news saved from the Internet. In practice, a common feature of IT systems is the ability to log events that occur during their execution. Such logs serve many purposes, such as helping system maintainers diagnose faults or detect inappropriate use of the system.

In this extension you will create your own "event logging" capability which records information about the way your program is used. This information is to be stored in a relational database, for later perusal using a tool such as the *DB Browser for SQLite*. Your solution must have the following features:

- Some form of interactive widget, or collection of widgets, must be added to your program's GUI to allow the user to control whether or not events should be logged, and to see whether or not event logging is currently enabled.
- As long as this function is activated by the user, your program must save a record of all user interactions with the GUI in a supplied SQLite database, `event_log.db`. A copy of this database accompanies these instructions. It contains a single table, `Event_Log`, which has two columns, `Event_Number` and `Description`. Your program must insert descriptions of the GUI events initiated by the user into this table for as long as the event logging feature is switched on. You should assume the database table exists but is empty when your program is started.

Example

Supplied with these instructions is an empty SQLite database called `event_log.db`. If you open it in the *DB Browser for SQLite* or an equivalent tool you will see that it contains a single table with two columns, as shown below.



In the instructions for Part A we used a demonstration program which managed a news archive sourced from the ABC News' Queensland web site. Now we extend its capabilities by adding an extra feature to the GUI for controlling logging of events in the database. Below is the updated GUI, showing the new widget, which in this case is a checkbox, although other widgets could be used to do the same job, such as on/off pushbuttons and a label.

Initially event logging is switched off by default. Now assume that the user decides to view the top ten news stories for Sunday, September 17th. The user extracts these stories from the archive but this action is not logged because event logging is turned off. However, at this point the user enables event logging and presses the button to display the extracted news stories in the web browser.

Both of these events will be recorded in the database. At this point the user can see the news as it was on that day, part of which is shown below. (At that time the football season was coming to a climax!)

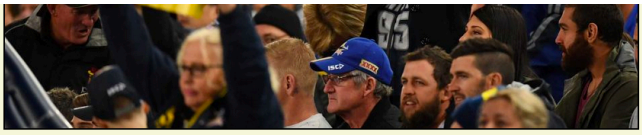
ABC News Archive (Queensland)

Sun, 17 Sept 2017

ABC

News source: http://www.abc.net.au/news/feed/50990/rss.xml
Archivist: Grimly Feendish

1. NRL crowds outstrip AFL as Giants semi attracts lowest finals attendance since WWI



The crowd at the Eels v Cowboys semi-final dwarfed the one that turned out just down the road to see GWS win its AFL semi-final against the Eagles.

Full story: http://www.abc.net.au/news/2017-09-17/nrl-wins-battle-of-the-crowds-over-afl/8954190
Dateline: Sun, 17 Sep 2017 08:29:31 +1000

---

2. Tourists flock to Geographe Bay for whale watching

Whales in Geographe Bay off the WA coast.

Full story: http://www.abc.net.au/news/2017-09-17/whales-in-geographe-bay/8952210
Dateline: Sun, 17 Sep 2017 08:09:04 +1000

Tiring of this old news the user then

1. presses the button to download the latest news from the Internet and store it in the archive,
2. selects the latest news and presses the button to extract the top ten "latest" stories from the archive, and
3. presses the button to display the extracted stories in the default web browser.

All of these events are also logged. The user then studies the news for the current day (which was Friday, October 20th when these instructions were written, towards the end of the "big wet" of 2017), part of which is shown below.

News source: http://www.abc.net.au/news/feed/50990/rss.xml
Archivist: Grimly Feendish

---

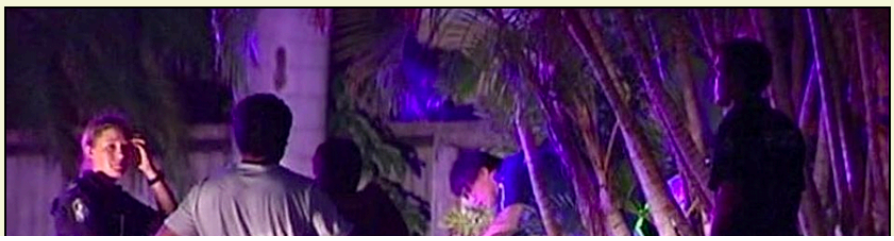
1. A school bus with students on board drives through flooded section of road near Bundaberg.

Video has been released of a school bus with several students on board driving through flood waters.

Full story: http://www.abc.net.au/news/2017-10-20/school-bus-drives-through-flood-waters/9071034
Dateline: Fri, 20 Oct 2017 16:00:55 +1100

---

2. Police need targeted training to handle people with mental illness: coroner

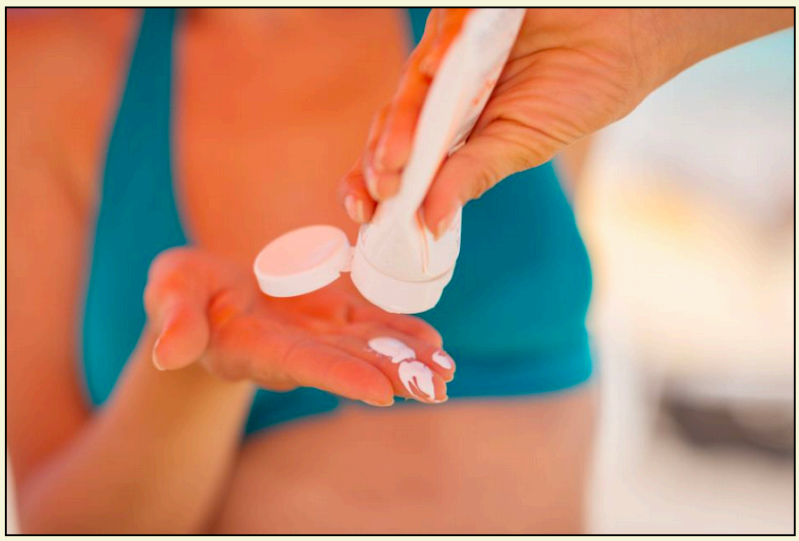


Nowhere is the increasing popularity of street art more apparent than in Australia's regional cities and towns.

Full story: http://www.abc.net.au/news/2017-10-20/street-art/9065066
Dateline: Fri, 20 Oct 2017 11:20:12 +1100

---

7. Sunscreen safety myths fuelling fears against use 'by nearly half of Australians'



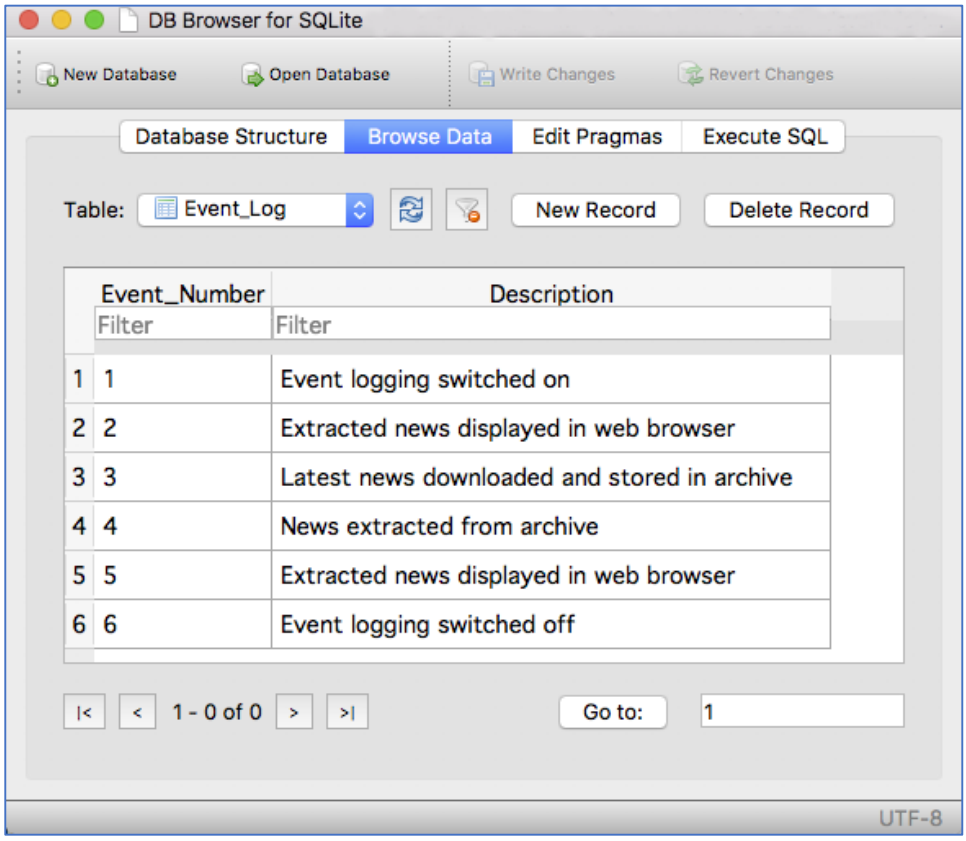
Many Australians fail to apply sunscreen on a daily basis because they don't think it's safe to use that frequently, research shows.

Full story: http://www.abc.net.au/news/2017-10-20/sunscreen-safety-myths-fuel-fears-against-use-australians/9068768
Dateline: Fri, 20 Oct 2017 11:07:22 +1100

Finally, the user switches off event logging (which itself is an event that is logged), and can easily tell that this has been done due to the empty checkbox.

The user could, of course, turn logging back on again, at which point interactions with the GUI would continue to be logged.

Later we can use a database tool to inspect the events that were logged. For this particular example the resulting database table appears as follows.



Notice that all the button presses while the event logging capability was activated have been recorded in the database, along with a number reflecting their order of occurrence. (Q: Why did we need to have an "event number" column when the rows are already numbered by the *DB Browser* on the left? A: Because the order of rows in a database table is not significant. If

we sort the rows alphabetically by the "description" we would no longer be able to identify the order in which the events occurred!)

Your task is to extend your solution to Part A of the assignment with an event logging capability equivalent to that above. The details of what "descriptions" you write to the log depend on the particular choice of widgets you used in creating your GUI.

Note that not all user interactions can be "logged". In our demonstration solution we used a Listbox widget for which the action of selecting an item from the list updates the current selection but does not produce an "event" in the GUI. However, the Button and Checkbox widgets we used can have "commands" associated with their use, and these can be logged as events in the database. In general, you must log all events associated with widgets whose activation can result in execution of a "command" or function.

Requirements and marking guide

To complete this task you are required to further extend the provided `news_archivist.py` template file with your solution to Part B, on top of your solution to Part A, to provide an event logging capability equivalent to that illustrated above.

The extension for Part B must satisfy the following criteria. Marks available are as shown.

- **Adding an "event logging" widget or widgets to the GUI (2%).** Your program must provide some simple, intuitive feature to allow the user to turn the event logging capability on and off. This could consist of a single widget as shown in the example above or some other combination of suitable widgets. Importantly, for privacy reasons, the user *must be able to tell at all times whether or not event logging is activated*.
- **Implementing the "event logging" functionality (3%).** For as long as the "event logging" feature is switched on your program must record all significant events occurring in the GUI in the database table `Event_Log`. Each record must consist of an integer event number and a textual description of the event. A record must be made in the database whenever the user activates any widget in the GUI which can have a "command" or function associated with it. Your Python code should assume that the necessary SQLite database already exists in the same folder as your Python program and its single table is empty when the program begins.

You must complete this part of the assignment using only basic Python features and the `sqlite3` module.

Supporting material

Accompanying these instructions we have provided a copy of the necessary SQLite database, `event_log.db`. It contains a single table, `Event_Log`, which has two columns, `Event_Number` and `Description`. As supplied the database table is empty. Your solution should assume that this database and its table *already exists in the same folder as your Python program*. Your GUI does not need to create the database or the table. Before you begin you should confirm that you can open this database with the *DB Browser for SQLite* or a similar database tool.

Development hints

- It should be possible to complete this task merely by *adding* code to your existing solution, with little or no change to the code you have already completed.
- The SQLite statements needed to complete this task are quite simple. Similar examples can be found in the relevant lecture demonstrations and workshop exercises.
- If you are unable to complete the whole task, just submit whatever parts you can get working. You will receive *partial marks* for incomplete solutions.

Deliverable

You should develop your solution by completing and submitting the provided Python template file `news_archivist.py` together with a folder containing an "archive" of at least seven previously-downloaded web documents. These documents must be stored in a folder called `InternetArchive`. Your submission can also include one or more *small* GIF files needed to support your GUI interface, in the same folder as your Python script, but **no other image files**. In particular, your "archive" folder may contain HTML/XML source code documents only, **no image files**. **Do *not* submit a copy of the SQLite database. We will use our own copy to test your solution.**

1. Complete the "statement" at the beginning of the Python file to confirm that this is your own individual work by inserting your name and student number in the places indicated. **Submissions without a completed statement will be assumed to be someone else's work.**
2. Complete your solution by populating your "archive" with downloaded web documents and developing your Python code. You must complete your solution using **only the modules imported by the provided template**. You do not need to use or import any other modules or files to complete this assignment.
3. Zip up your (a) Python program, (b) GIF files needed to support its GUI, and (c) the folder of archived web documents. **Upload the zip file** to Blackboard as a **single submission**.

Apart from working correctly your Python and HTML code must be well-presented and easy to understand, thanks to (sparse) commenting that explains the *purpose* of significant elements and *helpful* choices of variable, parameter and function names. **Professional presentation** of your code will be taken into account when marking this assignment.

If you are unable to solve the whole problem, submit whatever parts you can get working. You will receive **partial marks for incomplete solutions**.

How to submit your solution

A link is available on Blackboard under Assessment for uploading your solution file before the deadline (5:00pm on Wednesday, October 25th). Note that you can submit as many drafts of your solution as you like. You are strongly encouraged to *submit draft solutions* well before the deadline as insurance against computer and network failures. Students who encounter problems uploading their solutions to Blackboard should contact the IT Helpdesk (ithelpdesk@qut.edu.au; 3138 4000) for assistance and advice.