**FamilyTreeWebsiteBuilder**

A utility to produce HTML web pages, showing a family tree, from a GEDCOM format file.

**Overview**

This utility allows the user to produce HTML web pages, showing a family tree, from a GEDCOM format file.

An example is at <https://familyhistoryrecords.000webhostapp.com/>

It produces trees for up to 4 level-1 persons, e.g. the user’s grandparents. These level-1 persons are identified by their surname only, to protect their privacy in case they are still living. Other persons are shown with surname, forename(s) and year of birth.

Where a person has associated documents, the name has a hyperlink to a documents index page, listing the documents, and each document name has a hyperlink to a page showing an image of the document.

Each tree page shows the level-1 person with their parents (level-2), grandparents (level-3) and great-grandparents (level-4), where known.

Where the level-4 person has known ancestors, a hyperlink is provided to take the user to the next level tree which starts with the level-4 person.

Each tree page has hyperlinks to the top-level of the other 3 trees.

Modules

* FamilyTreeWebsiteBuilder.pyw

The main module. This module presents the user with a Windows user interface, allowing them to edit parameters, process a GEDCOM format file or build the website.

* params.py

This module presents the user with a user interface to edit parameters. It contains the parameter variables used by other modules and the following subroutines:

*get\_params* reads parameter values from params.txt.

*save\_params* writes parameter values to params.txt.

*edit\_params* presents the user with a user interface to edit parameters.

Parameters are:

|  |  |
| --- | --- |
| GED File | The name of the GEDCOM format file to be processed. The file should be in the location where the utility runs. |
| Page Size | Parameter not used in FamilyTreeWebsiteBuilder. |
| Initial Family | Parameter not used in FamilyTreeWebsiteBuilder. |
| Title Page Required | Parameter not used in FamilyTreeWebsiteBuilder. |
| Contents Required | Parameter not used in FamilyTreeWebsiteBuilder. |
| Index Required | Parameter not used in FamilyTreeWebsiteBuilder. |
| Document Images Required | Parameter not used in FamilyTreeWebsiteBuilder. |
| Website Path | Set to the full path of the folder to contain the web pages and containing an ‘images’ folder containing document images, e.g. if images are in C:\website\images, set this parameter to C:\website. Note: each document image must have a filename with the name and date of birth of an individual in the report, e.g. the filename of an image of the Birth Index Register document for Mary Jane Swales, born in 1885, should be ‘Swales, Mary Jane b. 1885 – BIR.jpg’. BIR is an image type. See ImageTypes.txt. |
| Title Line 1 | Parameter not used in FamilyTreeWebsiteBuilder. |
| Title Line 2 | Parameter not used in FamilyTreeWebsiteBuilder. |
| Title Line 3 | Parameter not used in FamilyTreeWebsiteBuilder. |
| Title Line 4 | Parameter not used in FamilyTreeWebsiteBuilder. |
| Country to remove 1 | Parameter not used in FamilyTreeWebsiteBuilder. |
| Country to remove 2 | Parameter not used in FamilyTreeWebsiteBuilder. |
| Person1 | The ID of the 1st person in the 1st family tree to be built |
| Person2 | The ID of the 1st person in the 2nd family tree to be built |
| Person3 | The ID of the 1st person in the 3rd family tree to be built |
| Person4 | The ID of the 1st person in the 4th family tree to be built |

* process\_ged\_file.py

This module contains the *process\_ged\_file* subroutine that reads the GEDCOM file specified as a parameter and produces Individuals.txt, Families.txt and Children.txt containing lists of individuals, families and children, with attributes.

* individuals.py

This module contains the ‘individuals’ list and the following subroutines/functions:

*read\_individuals* reads Individuals.txt into the ‘individuals’ list.

write\_individuals writes the ‘individuals’ list to Individuals.txt.

*get\_person\_name* returns the name of the individual (id passed as a parameter) in the form ‘surname, forename(s)’.

*get\_birth\_date* returns the year of birth of the individual (id passed as a parameter) in form ‘b. yyyy’.

*get\_family\_where\_child* returns the id of the family where the individual (id passed as a parameter) is a child.

* families.py

This module contains the ‘families’ list and the following subroutines/functions:

*read\_families* reads Families.txt into the ‘families’ list.

*write\_families* writes the ‘families’ list to Families.txt.

*get\_father\_id* returns the id of the husband in the family where the individual (id passed as a parameter) is a child.

*get\_mother\_id* returns the id of the wife in the family where the individual (id passed as a parameter) is a child.

* process\_document\_images.py

This module processes the document image files in the \images folder within the website folder identified in params.

It first reads the list of document types from File\_Type\_Order.txt into a ‘file\_type\_order’ list.

It then scans the ‘images’ folder for jpg files, and for each one it obtains the sort order from the ‘file\_type\_order’ list and writes the individual’s name, order and file type to a ‘records’ list.

It then sorts the ‘records’ list and then iterates through the sorted list. When a new individual is found or the end of the list is reached it closes the previously opened HTML file showing the list of documents (if any). It then opens a new one for the new individual and writes the header information including the individual’s name. An example file name is ‘List Hunter, Horace b. 1884.html’

It then writes the document type, including a hyperlink to the HTML file displaying the document.

It then opens and writes the HTML file displaying the document. An example file name is ‘Doc Hunter, Horace b. 1884 - BIR.html’

* create\_tree.py

This module contains the *create\_tree* subroutine that builds the website based on the parameters and information in Individuals.txt, Families.txt and Children.txt.

*create\_ tree* calls *write\_tree\_page* for each of the persons defined in params.

*write\_tree\_page*  creates a ‘persons’ list then calls *add\_person* 15 times, once for each position on the tree page. Each time it passes the number of the person to be added (e.g. Person 2), and the number of the person that the person is a parent of (e.g. Person 1), and whether the person is the father or mother. Persons are arranged on the page in the following way:

Person 4

Person 3

Person 5

Person 2

Person 7

Person 6

Person 8

Person 1

Person 11

Person 10

Person 12

Person 9

Person 14

Person 13

Person 15

*write\_tree\_page*  then removes the forenames and year of birth from the person’s details if the person is at the top-level of the tree.

It then creates an HTML file to show the tree page. The filename is ‘Tree’ followed by the level and names of the first person, e.g. ‘Tree4 Hunter, John.html’. It then writes spaces, connecting lines, person names and year of birts and links to the next level tree from the top of the page to the bottom, i.e. Person 4, then Person 3 etc. Spaces and connecting lines are written via subroutine calls. Person names are written using the *write\_person* subroutine.

It then writes the hyperlinks to the 4 top-level trees, then closes the file.

If person 4, 5, 7, 8, 11, 12, 14 or 15 has a next level tree, *write\_tree\_page* is called for each individual, to create the tree.

*add\_person* adds a person to the ‘persons’ list. If the person being added is person 1 (passed as a parameter), a dummy entry is first added and the new entry is created with the individual ID passed as a parameter. If the person being added is person 2-15, the new entry is created with the father or mother of the person that is in the child person record (passed as a parameter). The ‘persons’ record is populated with the individual’s ID, name, date of birth, a ‘more’ indicator saying if the individual has a known parent and a ‘link’ indicator saying if a list of documents relating to the individual exists.

*write\_person* writes the person’s name and birth year (both passed as parameters) to the open HTML file. This includes a hyperlink if the person has an associated document list (‘link’ indicator passed as a parameter). If a link to the next level tree is required (‘more’ indicator passed as a parameter), the link is written.