February 2019 Features Update

Table of Contents

Global....3
- Updating the Page Appearance....3
- Scrolling Through Large Tables Using the Keyboard....4
- NameSpaces....5
- Templates....6
- Enhancements to Common Record Support....7
- Language Support....7
- JavaScript Object Notation (JSON)....8
- Database Normalization....8

Specific Changes....12
- Genealogy Entry Page....12
- General Blogs....12
- Person....12
- Family....12
- Birth....12

Bug Fixes....13

Illustration Index

Illustration 1: Modern Menu Button....3
Illustration 2: New Menu for German-Speaking Client....4
Illustration 3: The Table of Supported Countries with the new Scrolling Enhancements....5
Illustration 4: Page Apologizing because there is no Polish Version....8

Global

This report covers one year of changes to the services and implementation.

Updating the Page Appearance

The common appearance of the pages on the site was basically frozen almost 20 years ago and was based upon best practices of that time. However the rest of the world has moved on and it was more than time for this site to be refreshed. Two decades ago Microsoft introduced the menu button. That is instead of having all of the major menu items continuously displayed on every page, only a button was presented which the user could click in order to see the menus. Particularly as most users of technology switched to devices such as mobile phones and tablets it became important to use the space on the device more efficiently. The now ubiquitous menu button consists of a stack of three thick black lines. It is therefore sometimes likened to a stack of three hamburger patties. Clicking on the button displays the following:

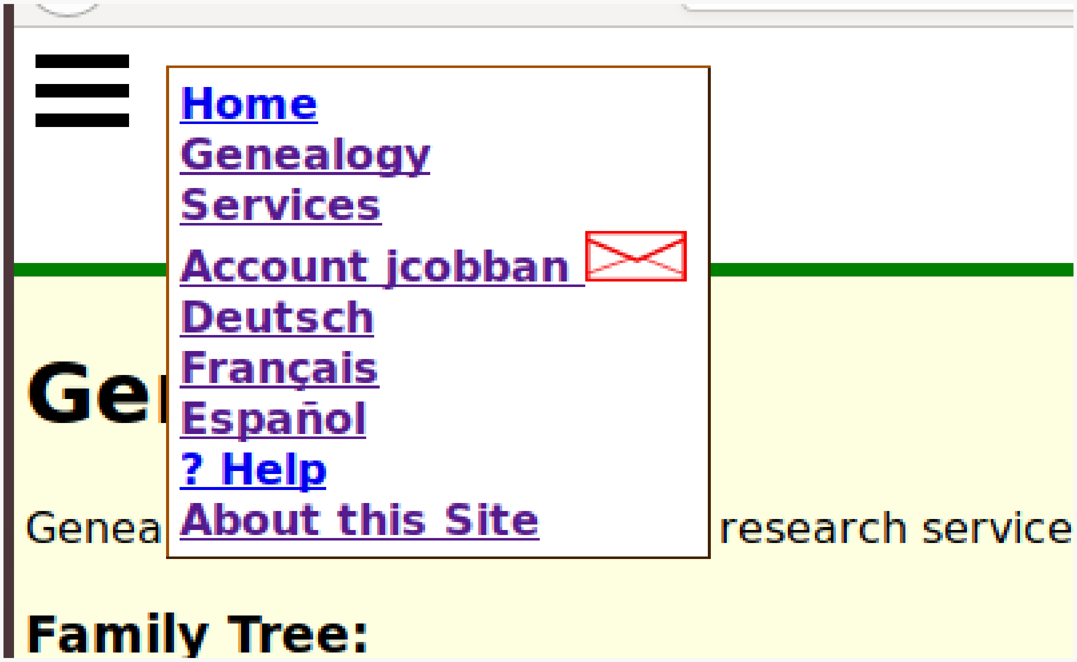


Illustration 1: Modern Menu Button

This compact menu combines information that was formerly displayed in three different parts of the page and provides the opportunity to display additional information which was not available using the old format.

1. The hierarchy of pages leading to the current page is displayed first.
2. After this comes the invitation to sign on to the site, or if you are already signed on the option to manage your account, and to view any messages which have been sent to you using the internal secure messaging system. The presence of such messages is now indicated by an icon representing an envelope.

3. If the current page is available in languages other than the current language you are given the option of selecting to see the information in one of the supported languages.
4. A link to a page which describes the overall features of the current page and how to exploit them.
5. A link to a page which describes the overall objectives of the site and explains the common features of all pages on the site.

The contents of this menu is customized to the language of communication chosen by the user.

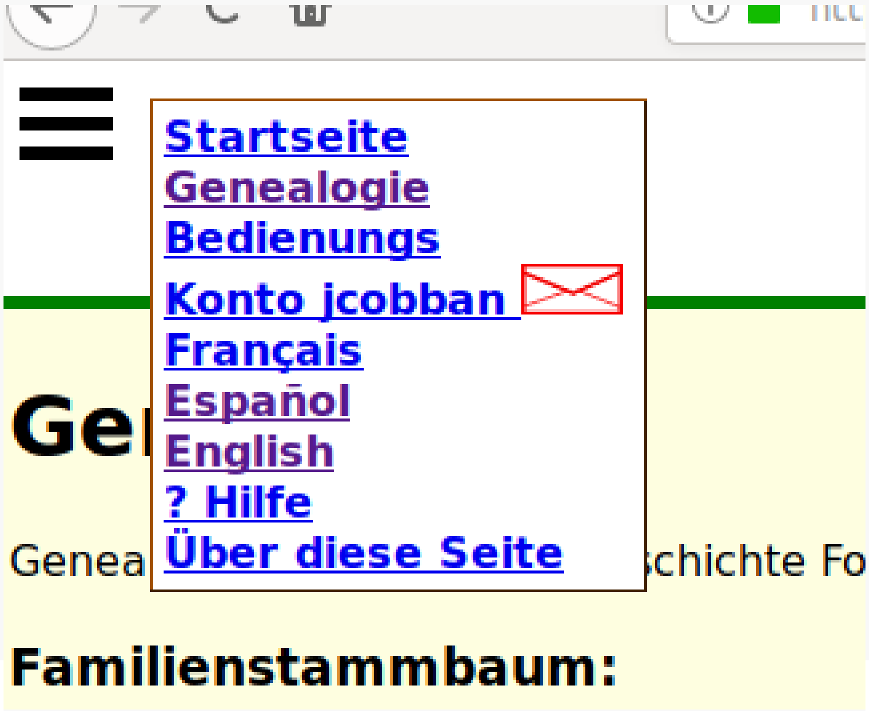
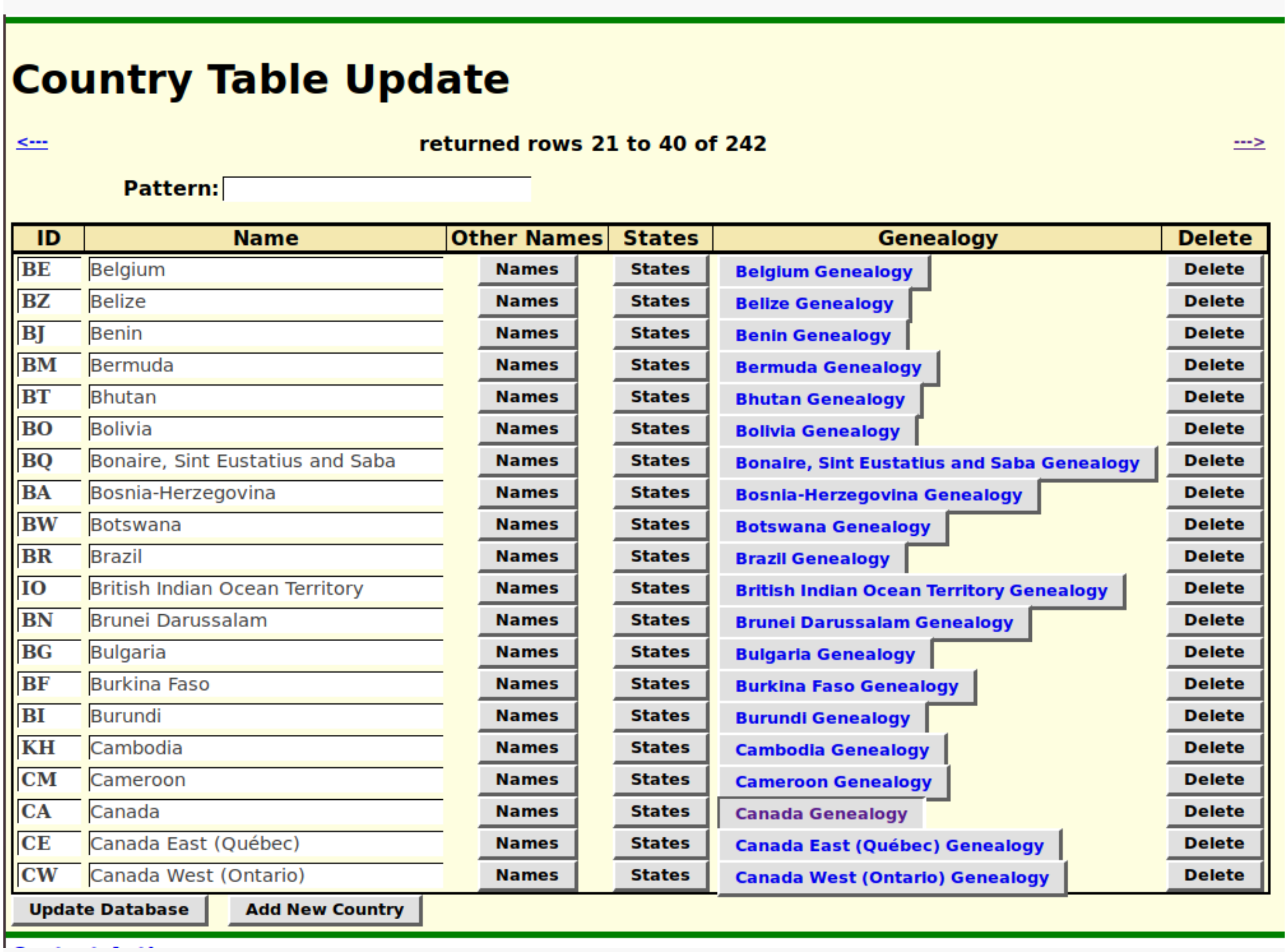


Illustration 2: New Menu for German-Speaking Client

Scrolling Through Large Tables Using the Keyboard

Where tabular information has a large number of rows the application generally displays only 20 rows at a time, which is a number which can be viewed on most devices, and displays forward and backward arrows above the table pointing to the left and right. Clicking on these arrows with the mouse, or tapping on them on a touch-sensitive device, takes you to the next or previous portion of the table. Holding the mouse over these arrows explains what they mean. As a convenience for people who like to keep their hands on the keyboard the PgUp and PgDn keys are now supported to move through tables. As I get to them the width of the forward and backward arrow lines are re-sized so they are never wider than the tables they describe.



Country Table Update

<--- returned rows 21 to 40 of 242 --->

Pattern:

ID	Name	Other Names	States	Genealogy	Delete
BE	Belgium	Names	States	Belgium Genealogy	Delete
BZ	Belize	Names	States	Belize Genealogy	Delete
BJ	Benin	Names	States	Benin Genealogy	Delete
BM	Bermuda	Names	States	Bermuda Genealogy	Delete
BT	Bhutan	Names	States	Bhutan Genealogy	Delete
BO	Bolivia	Names	States	Bolivia Genealogy	Delete
BQ	Bonaire, Sint Eustatius and Saba	Names	States	Bonaire, Sint Eustatius and Saba Genealogy	Delete
BA	Bosnia-Herzegovina	Names	States	Bosnia-Herzegovina Genealogy	Delete
BW	Botswana	Names	States	Botswana Genealogy	Delete
BR	Brazil	Names	States	Brazil Genealogy	Delete
IO	British Indian Ocean Territory	Names	States	British Indian Ocean Territory Genealogy	Delete
BN	Brunei Darussalam	Names	States	Brunei Darussalam Genealogy	Delete
BG	Bulgaria	Names	States	Bulgaria Genealogy	Delete
BF	Burkina Faso	Names	States	Burkina Faso Genealogy	Delete
BI	Burundi	Names	States	Burundi Genealogy	Delete
KH	Cambodia	Names	States	Cambodia Genealogy	Delete
CM	Cameroon	Names	States	Cameroon Genealogy	Delete
CA	Canada	Names	States	Canada Genealogy	Delete
CE	Canada East (Québec)	Names	States	Canada East (Québec) Genealogy	Delete
CW	Canada West (Ontario)	Names	States	Canada West (Ontario) Genealogy	Delete

Illustration 3: The Table of Supported Countries with the new Scrolling Enhancements

NameSpaces

In order to make the implementation of the site more portable all of the classes which implement portions of the functionality, particularly support for the various types of records which represent elements of the family tree and the original document transcriptions, have been repackaged into namespaces. The purpose of a namespace is to avoid conflicts over the names of resources so that the application code implementing the site can be embedded into other sites without requiring renaming. So, for example, the base class which represents the shared functionality of all records in the database is now identified as Genealogy\Record so that if the code is embedded into an application which already has a class named Record there will be no conflict. The classes which implement the functionality for reading and writing GEDCOM 5.5 files are in the namespace Genealogy\GedCom so, for example, the common functionality of a GEDCOM tag is implemented in the class Genealogy\GedCom\Tag.

To facilitate the use of the code implementing this site by others, and to permit other developers to contribute to the implementation of the site, the source code is now stored on https://github.com/jcobban/Genealogy

Templates

There have been a number of improvements to the support for templates:

- The common support for template internationalization, including displaying the appropriate apology for lack of support for a particular language, is now consolidated in the constructor for the class FtTemplate. Class FtTemplate, unlike its parent class Template, is specific to the genealogy application and understands the specific file naming conventions used by the application. If the FtTemplate constructor is called with a file name "...pagell.html" or "...dialogll.html" it recognizes that this is a master page/dialog template and behaves as it has in the past by simply passing it to the parent class Template. However if it does not match that pattern it is assumed to be the name of the page and language specific template for the current application script. It invokes the parent class with a file name "...pagell.html" or "...dialogll.html". If the requested page and language specific template exists it is made the MAIN substitution, otherwise the language code is replaced with "en" and the appropriate language specific apology text is added to `$warn`. See "Language Support" on page 7. This consolidates 16 lines of code that existed in every page specific script.
- The error reporting pages `/error/page404.php`, for incorrect page names, and `/error/page403.php`, for access rejection, now use class Template and templates have been created for English, French, Spanish, and German.
- Define constants for brace style insertions. This is to avoid an issue while editing the code of the template implementation that string constants containing brace characters were confused with the braces used by the programming language to delimit blocks of logic.
- Use linked list of strings for better performance in the innermost loop.
- The basic page templates now contain a <link> element to permit application scripts to define CSS style sheets within the <head> portion of the page.
- Move initialization of fields used in the FamilyTree page template to FtTemplate::customization
- Support passing the full text of a template to the constructor in place of a file name.
- Apply tag updates for $template->set(,,true) to permit manipulating tags within the value.
- Pass the full substitution text to TemplateTag::update.
- Add the definition of $GOOGLECC and $GOOGLEKEY in method FtTemplate::customization to identify authorization to use Google services such as Google Maps. The value assigned to $GOOGLEKEY is defined in the file `/includes/Genealogy/configDb.inc` along with the username and key used for accessing the SQL database. This facilitates defining a different authentication key for different sites since it is recommended that a specific Google key be restricted to a single site.
- Enhance method `$template->setFields` to accept an instance of a class implementing interface Iterator.
- Implement readonly ArrayAccess to tags by id to improve readability of application code. That is in place of `$template->getElementById($id)` the programmer can use the more compact `$template[$id]`. This also encourages the programmer to use

`$template[$id]->update($value)` rather than `$template->updateTag($id, $value)`.

- Permit the application script to define $BREADCRUMBS.
- Support requesting an alternate language page template. In place of the original single master template page.html there are now multiple `pagecc.html` templates so that the appearance of the page header and footer can be customized to the requested language.
- Include substitutions defined by $template->set in expansion of tags customized by $template->updateTag($id, $value).
- Implement the changes to the page layout described above in the derived class Genealogy\FtTemplate.
- If possible replace advertising <iframe> with the contents of the <body> of the advertising file.
- Selection of random advertisement is moved to Genealogy\FtTemplate::displayBegin from common.inc .

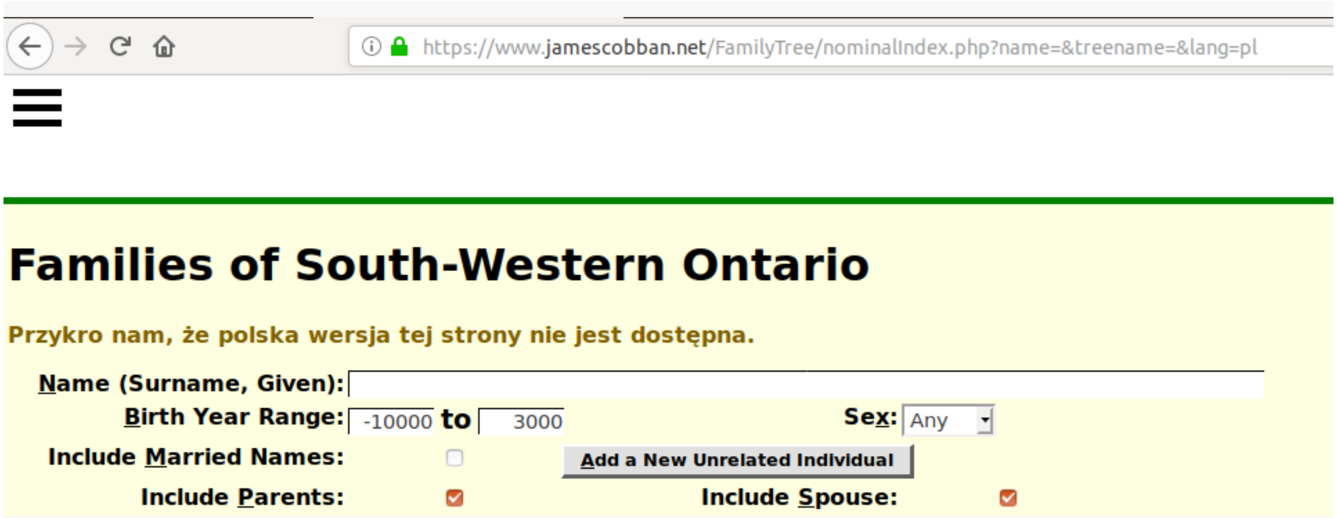
Enhancements to Common Record Support

There have been a lot of improvements to the common handling of database records:

- Using `foreach($record as $key => $value)` now includes both the fields from the database record and any extra pseudo-fields defined by $record->set.
- Add method setGetModeHtml which specifies whether values returned by method get are valid HTML.
- $this->msg and $this->getErrors() added to permit constructors to report invalid parameter values, for example bad password, without throwing an exception. Instead if an attempt is made to update or create the record the request is ignored with a warning. Throwing exceptions from constructors complicates the implementation of application scripts.
- Remove numeric code to text tables, in their place add member $record->template and method $template->setTemplate() to define translations for internationalization.
- Add support for multiple field keys to method $template->getId().

Language Support

If there is no language specific template for the current page for the language requested by the user, using the `lang=cc` parameter, the pages now try to apologize in the requested language for presenting the English version of the page. This adds an additional field, `sorry`, to the table of Languages which is administered through the Languages administration page /Canada/LanguagesEdit.php?lang=cc. On 13 Feb 2019 this option was defined for 11 languages.


https://www.jamescobban.net/FamilyTree/nominalIndex.php?name=&treename=&lang=pl
Add a New Unrelated Individual

Families of South-Western Ontario

Przykro nam, że polska wersja tej strony nie jest dostępna.

Name (Surname, Given):

Birth Year Range: -10000 to 3000 Sex: Any

Include Married Names: ☐

Include Parents: ☑ Include Spouse: ☑

Illustration 4: Page Apologizing because there is no Polish Version

JavaScript Object Notation (JSON)

One of the techniques for making web-sites more efficient and more dynamic is to permit the code which is run in the browser, which is written in a language usually called JavaScript, to interact with the server to obtain information which is used to update the appearance of the page, instead of the older way which required refreshing the page with a mixture or old and new information. In 1998 Microsoft introduced the XMLHTTP ActiveX control to support implementing a web interface to their Outlook electronic mail interface. This idea was quickly adopted by other browsers who standardized the name of the resource as XMLHttpRequest. This was popularized by Google through its use in services such as Gmail and Google Maps. This capability is used throughout this site. However since the requests are initiated by JavaScript and the results must be interpreted by JavaScript, XML is generally a cumbersome, although extremely flexible, way of describing the information which is sent from the server. In particular the ability in XML to transmit both data and attributes which describe how the data is to be interpreted is rarely useful. In practice the XML response must be converted into a JavaScript object in order for the application to act on it. Therefore it is more common now to simply pass JavaScript objects using the textual representation that is used in JavaScript programs to define objects. So gradually all of the code which is currently retrieving XML from the server is being replaced by retrieving JSON objects.

Database Normalization

The original database definition used for the site was largely borrowed from the commercial product I was using at the time on my personal computer/ This design has a few weaknesses arising our of the objectives of the original designers:

1. The design assumed that the database would be accessed only by a single user, since it was implemented using Microsoft Access on a Microsoft Windows personal computer. The design therefore assumed that the database did not need to be locked for exclusive use while making a series of changes which all needed to be completed before the database would be once again logically consistent.

2. Furthermore since there was only one application using the database, the application stored information obtained from the database in memory, since the contents of the database could not be modified without the application’s knowledge.
3. The original application supported Internationalization (I18n), but only for one language per system, whereas a shared service must support multiple languages at the same time.
4. Almost all of the tables use an arbitrary numeric index which has no meaning in and of itself except to uniquely identify a specific record. Not all implementations of the SQL standard for relational databases even support this sort of index because it is considered a poor design. As much as possible each record in each table should be identified by a unique key, or set of fields acting as a compound key, which has meaning.
5. The database design duplicates the same information in multiple fields in multiple tables. That makes the first assumption problematic, since all of the places where a particular piece of information must be recorded must all be changed “simultaneously” in order to maintain logical consistency.

A particular example of the last issue was that the name of an individual appeared in three different tables, probably with the expectation that this would improve performance since the application would not need to look further than the current record to get the name. Specifically:

1. The primary name of an individual appeared in the Person record, and in an instance of the Name record with the same IDIR and `Order`=0, and in every Family Record for that individual.
2. The married name of an individual appeared both in the Family record, and in an instance of the Name record with the same IDIR, IDMR, and `Order`=-1.

Part of the design flaw was, based upon the way the Name table is described in the original documentation, that it was originally intended that the Name table would be used only for alternate names, that is AKAs. If that had been true then there would have been less duplication.

The implementation is changed so that all references to the name of an individual, in all contexts, are to the copy of the name in a Name record, so that only the information in that record is maintained as part of the logical consistency of the database. Class methods which formerly returned or modified name fields in the Person and Family records now return or modify the value in the appropriate instance of the Name record. Furthermore the scripts that return the contents of Person and Family records in either XML or JSON format now present the name information as it appears in the associated Name record, and not as it appeared in the now obsolete fields within the Person and Family records. For example the JSON representation of a typical Family record now looks like:

```
{
"33477":
    {
        "idmr": "33477",
        "idirhusb":     "119621",
        "husborder":    "0",
        "husbprefmar":  "1",
        "idirwife":     "119622",
        "wifeorder":    "0",
        "wifeprefmar":  "1",
        "marriednamerule":      "1",
        "idms":         "1",
        "templetag":    "0",
        "taggroup":     "0",
        "remindertag":  "0",
        "notmarried":   "0",
        "nochildren":   "0",
        "alreadyused":  "0",
        "ppcheck":      "0",
        "added":        "20180423",
        "addedtime":    "0:42",
        "updated":      "20180423",
        "updatedtime":  "0:48",
        "id":   "34035",
        "private":      "0",
"husbpriname":{
        "idnx": 338543,
        "idir": 119621,
        "order":        0,
        "marriednamecreatedby": 0,
        "idmr": 0,
        "prefix":       "",
        "title":        "",
        "surname":      "Watson",
        "givenname":    "James William",
        "userref":      "",
        "akanote":      "",
        "preferredaka": 0,
        "treename":     ""
}
,"husbmarname":{
        "idnx": 338549,
        "idir": 119621,
        "order":        -1,
        "marriednamecreatedby": 2,
        "idmr": 33477,
        "prefix":       "",
        "title":        "",
        "surname":      "Watson",
        "givenname":    "James William",
        "userref":      "",
        "akanote":      "",
        "preferredaka": 0,
        "treename":     ""
}
,"wifepriname":{
        "idnx": 338544,
        "idir": 119622,
        "order":        0,
        "marriednamecreatedby": 0,
        "idmr": 0,
        "prefix":       "",
        "title":        "",
        "surname":      "Garner",
        "givenname":    "Dora Edith",
        "userref":      "",
        "akanote":      "",
        "preferredaka": 0,
        "treename":     ""
}
,"wifemarname":{
        "idnx": 338548,
        "idir": 119622,
        "order":        -1,
        "marriednamecreatedby": 2,
        "idmr": 33477,
        "prefix":       "",
        "title":        "",
        "surname":      "Watson",
        "givenname":    "Dora Edith",
        "userref":      "",
        "akanote":      "",
        "preferredaka": 0,
        "treename":     ""
}
,"married":{
        "ider": 0,
        "idir": 33477,
        "idtype":       20,
        "idtypemeans":  "Family",
        "idet": 69,
        "eventtype":    "married",
        "order":        0,
        "eventdc":      "002603192100000000",
        "eventd":       "26 Mar 1921",
        "eventsd":      19210326,
        "idlrevent":    10464,
        "eventloc":     "Glencoe, Middlesex, ON, CA",
        "desc": "",
        "gedtag":       "",
        "eventexclude": "N",
        "idar": "0",
        "description":  "",
        "sentenceoverride":     "",
        "qstag":        "N",
        "rgexclude":    "N",
        "kind": 0,
        "ldstempleready":       null,
        "preferred":    1
}

}
}
```

Specific Changes

Genealogy Entry Page

In addition to exploiting the new page layout and other features previously described, the main entry page to the Genealogy site now scrolls the main portion so that the page header and page footer are always visible. This is also done for the entry page for the Canadian censuses.

General Blogs

One day I was frustrated that the only support for rich-text documents on my Android tablet and cell-phone was the Microsoft Office 365 product. Yes this product comes without a license fee and permits the user to view Microsoft Office documents, but it is not Free Open-Source Software, FOSS, because it is licensed by Microsoft under a set of restrictions on what you can use it for, you cannot take the source code and enhance it to add functionality which the Microsoft marketing department has decided should be available only on their pricier services, and it only creates documents that use the proprietary and patent protected file formats defined by Microsoft, and not the Open Document Foundation file formats which come without a license or patent restrictions. Furthermore you cannot pass the license on to another user if you decide you do not need it any more, just as you cannot pass your license for any of Microsoft's products even if you can prove that you deleted all trace of the product from your computers. I wanted to be able to create rich-text documents which I could post on my site for anyone to see and comment on using any platform. So I extended the existing support for internal messaging and commenting on database records to include the ability to create documents which are not related to the Family Tree or any other aspect of the site. This facility is visible at General Blogs.

Person

The method `$person->getName()` redirects to `$person->getPriName()` and now implements internationalization of the birth and death dates. It also returns "`private`" if the current requestor does not have permission to access the private information about the Person.

Family

The method `$family->getName()` redirects to `$person->getPriName()` and now implements internationalization of the birth and death dates.

Birth

This class, which represents the transcription of a birth registration or certificate, is brought into line with the other class definitions by accepting an associative array of search parameters. The parameter defining the administrative domain, that is province or state, may now be an instance of Domain.

Bug Fixes

There are thousands of bugs fixed in the past year including.

- Correct the output of toXML where the primary index of the table is multiple fields.
- Report problem in debugPrepQuery if sqlParms contains an array value.
- Printing backtrace failed for recursion on parameters.
- method Address::mergeAddresses used the obsolete Address::deleteAddresses instead of RecordSet→delete()
- MySQL started objecting to assigning an empty string to an INT, so ensure INT fields are set to NULL to represent an empty string. MariaDB does not yet flag this as an error.