USING AJAX , PHP, MYSQL AND JAVA FOR INTEGRATION OF WEB 2.0-METHODS IN ONLINE-GENEALOGY

Papadakis George[1], Schleusener Heinz[2]

[1]Department of Electrical and Computer Engineering,National Technical University of Athens,
[2]School of Process Science and Engineering,Technische Universität Berlin

Abstract

Genealogy is one of the most contemporary applications of Web 2.0 as there is an increasing number of newly developed web sites that deal with it. This trend traces back to the growing number of persons that are interested in their family history and have gathered lots of related material and photos. Thus, there is an urgent need for organizing, documenting and managing all this data in an easy and useful manner.

The prevailing format for genealogical data, e.g. important dates of individuals as well as of families, is the GEDCOM (**Ge**nealogical **D**ata **CO**mmunications File Format) file format. This format was defined by the "Church of Jesus Christ of Latter Day Saints“(LDS Church) in 1984 but is still under continuous development, with the latest version being based on XML structures. Though very common, it indisputably has some drawbacks, one of which is its complex text structure, which clearly shows its age in the current era of relational databases. Moreover, GEDCOM files are usually overwhelmed with a considerable amount of unimportant information.

The main goal of our approach is therefore to parse GEDCOM files (using PHP), retaining only the most significant information that is next inserted in a relational database (maintained using MySQL). The stored data is then available for display, i.e. for depicting the corresponding family trees through the use of the PHP graphics library. Furthermore, we used server and client side languages, namely AJAX and PHP, to create a user interface with typical Web 2.0 behaviour, as we intent to publish the outcome of our work in the WWW. We also make use of an open-source Java library that implements some machine learning algorithms so as to ensure data integrity in our database, i.e. to avoid duplicate data (an individual that is stored twice for example). The resulting application will be initially published on a website with genealogical and historical content for a region located in former Austria-Hungary.

Keywords

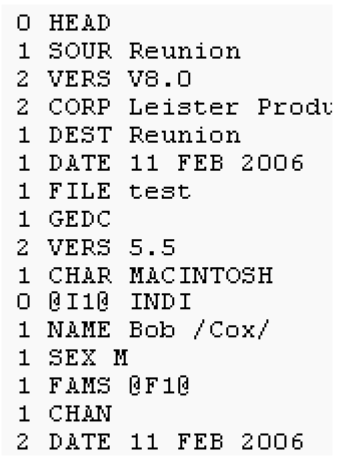
Web 2.0, Ajax, PHP, MySQL, Java, Genealogy, Gedcom, Family Tree

1. Introduction

The goal of this project was to develop an application that facilitates the maintenance and visualization of family trees. In this context, we initially had to survey the numerous pre-existing efforts in this popular field. Then we went on to decide which of their features are indispensable to a new approach to genealogy management and which new features could add more value to our application. In the following paragraphs we present the key points of our implementation step by step.

2. Necessary Information

The prevailing format for managing family trees is undoubtedly the GEDCOM file format (GEnealogical Data COMmunications file format), introduced by the „Church of Jesus Christ of Latter Day Saints" (LDS Church) in 1984. Though fairly successful, this format has some serious shortcomings we tried to overcome. More specifically, it entails a huge amount of information, a large part of which is actually useless, while simultaneously failing to capture efficiently some important information, for example divorces. The complicated and outdated structure of GEDCOM files is reflected in the following image.



```
0 HEAD
1 SOUR Reunion
2 VERS V8.0
2 CORP Leister Produ
1 DEST Reunion
1 DATE 11 FEB 2006
1 FILE test
1 GEDC
2 VERS 5.5
1 CHAR MACINTOSH
0 @I1@ INDI
1 NAME Bob /Cox/
1 SEX M
1 FAMS @F1@
1 CHAN
2 DATE 11 FEB 2006
```

Figure 1. An excerpt from a GEDCOM file, representative of its unwieldy structure

In this context, we initially had to specify which pieces of information are indeed worth of being maintained by a genealogy application. After a thorough examination of the GEDCOM format, we reach the following conclusions:

- every <u>individual</u> participating in a family tree should be represented by the following information:
  1. name, consisting of name prefix, given name, nickname, surname prefix, surname and suffix
  2. date, time and place of birth
  3. date, time and place of christening
  4. date, time and place of death
  5. sex
  6. occupation

7. education
8. nationality
9. religion

- every <u>family</u> consists of the following fields:
  1. husband
  2. wife
  3. date, time and place of the ceremony of the marriage
  4. date, time and place of the ceremony of the divorce (if there is one)
  5. its children along with the date, time and place of adoption (for those that are adopted)

Every individual as well as family can also be associated with one more pictures, meaning that a photo album for each entity is supported. Moreover, they are associated with the person that submitted or changed the corresponding information as well as the date this information was last changed.

We also have to stress that of the above fields, only the name of an individual is required to contain real information while the others can in most cases be left blank. This is so due to the fact that it is indisputably difficult to acquire so much information for people that lived hundreds of years ago.

3. Structure of Application

Based on the above mentioned conclusions and design decisions, we first went on to design a relational database where all the necessary information will be stored. To be more specific, the database comprises of 9 tables in total as well as a bunch of views, each one corresponding to a blood relationship. The DBMS environment we used for the maintenance of the database is MySQL, version 5.0.45. The schema of our database is depicted in the figure 2.

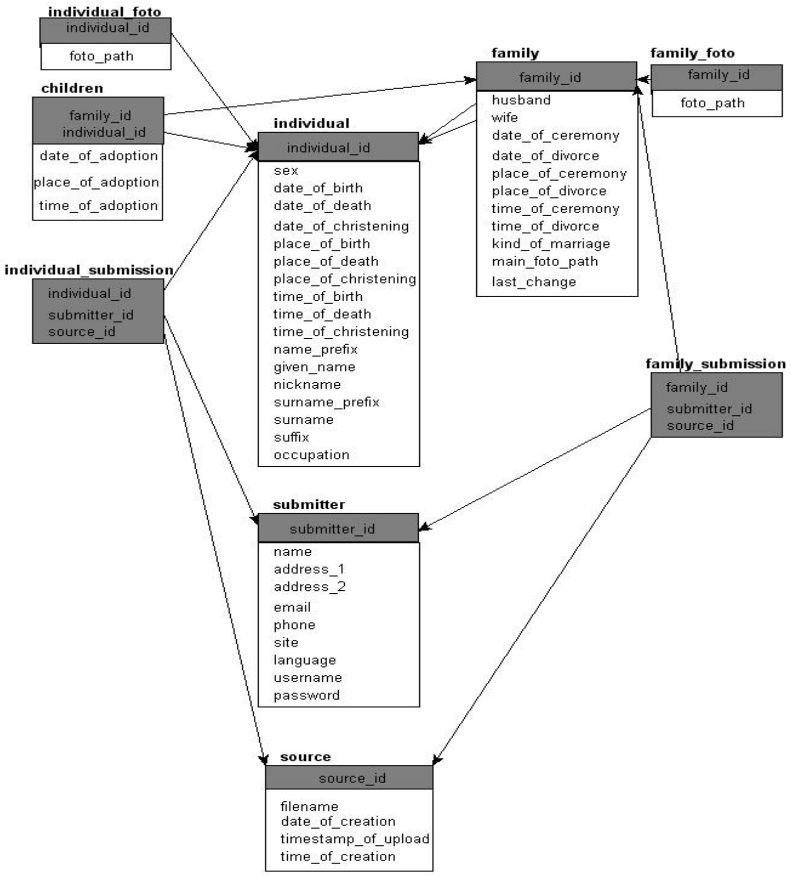


Figure 2. The relational schema of our database.

For ease of interaction with the database, we also created a package in PHP responsible for the maintenance the aforementioned information. Analytically, this package consists of classes that create objects modelling the tables in the database. Thus, for each table there exists a class having the fields of the table as properties as well as methods that set, retrieve and store the associated information.
We then went on to specify the capabilities our application. More specifically, a certified user, meaning a person that possesses an application account, is given the options depicted in the following picture (fig. 3):

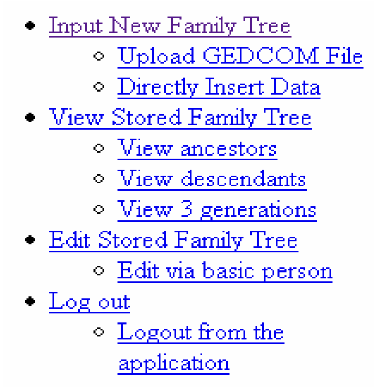


Figure 3. Basic menu options of the application

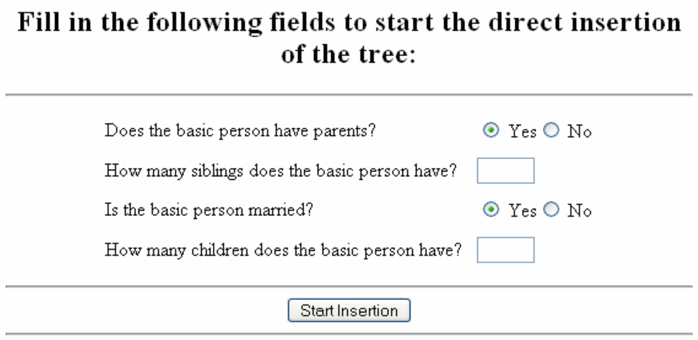
These functions are analyzed in detail in the following:

- **Insert a family tree**

The insertion can be accomplished in two different ways:

1. *By uploading a GEDCOM file*, that is a file already created by another (desktop) application. For the implementation of this first option, we had to develop a **GEDCOM parser**. This parser detects the information used by our application from a GEDCOM file based on the corresponding tags, while taking special care for the pictures associated with an individual or family. PHP objects are afterwards created based on the gathered information and used for storing the data.

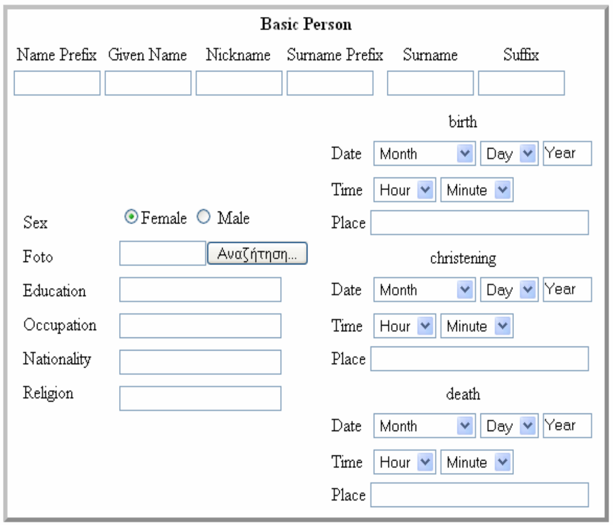
2. *Directly, by filling in the fields of a form with the available information*. To do so, the user has first to specify some information about the basic person, such as the number of her siblings (if information about her family is going to be stored) as well as the number of her children (as long as she is married). This is simply accomplished by filling in and submitting a small form, depicted in the next figure, which then is replaced by a more complicated one containing all the necessary fields for every individual and marriage (figures 5 and 6 respectively). After submitting this second form, a JavaScript function is called to verify the correctness of the given information. To be more specific, special care is taken for the validity of the given dates and for the names of the individuals. No other field needs to be checked. Following the JavaScript control, a PHP script is called to process the form, namely to create the corresponding objects that are then used to store the information.



Fill in the following fields to start the direct insertion of the tree:

Does the basic person have parents?	[✓ Yes] No
How many siblings does the basic person have?	
Is the basic person married?	[✓ Yes] No
How many children does the basic person have?	

Figure 4. Required Data before inserting directly a new family tree



Basic Person

Name Prefix	Given Name	Nickname	Surname Prefix	Surname	Suffix

Sex	[✓ Female] Male
Foto	
Education	
Occupation	
Nationality	
Religion	

birth

Date	Month Day Year
Time	Hour Minute
Place	

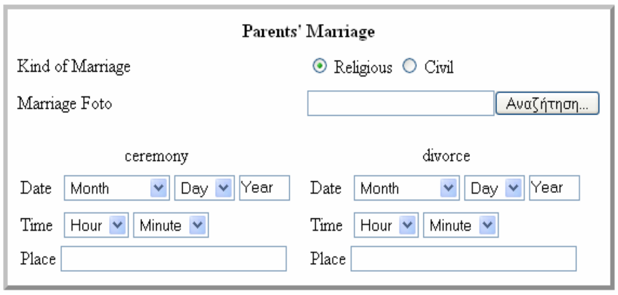
christening

Date	Month Day Year
Time	Hour Minute
Place	

death

Date	Month Day Year
Time	Hour Minute
Place	

Figure 5. Data that can be specified for an individual while inserting directly a new family tree



Parents' Marriage

Kind of Marriage	[✓ Religious] Civil
Marriage Foto	Αναζήτηση...

ceremony		divorce	
Date	Month Day Year	Date	Month Day Year
Time	Hour Minute	Time	Hour Minute
Place		Place	

Figure 6. Data that can be specified for an marriage while inserting directly a new family tree

- **View a family tree**

As fig. 3 demonstrates, there are three options for displaying the information stored in the database:

1. Display only the ancestors of a person
2. Display only the descendants of a person
3. Display the parents, siblings, marriage and children of a person (*3 generation family tree*)

In the following figures some illustrative family trees for each kind are displayed (the basic person is always depicted with a purple rectangle, while cyan and pink rectangles denote men and women respectively).

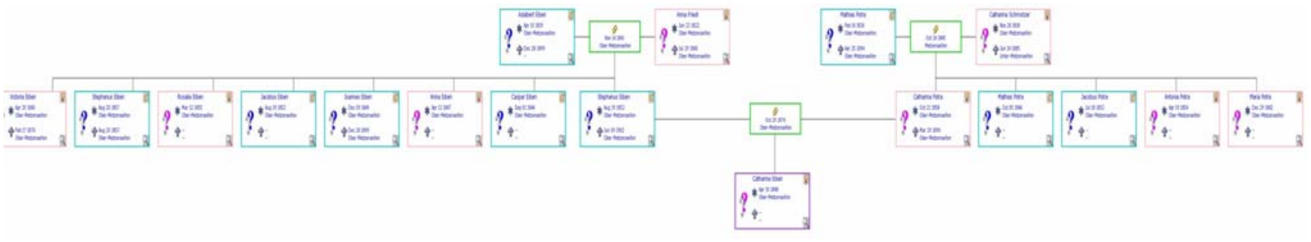


Figure 7. Ancestor family tree

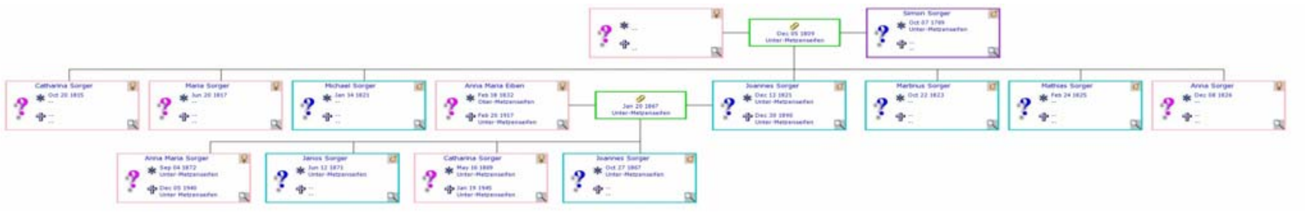


Figure 8. Descendant family tree

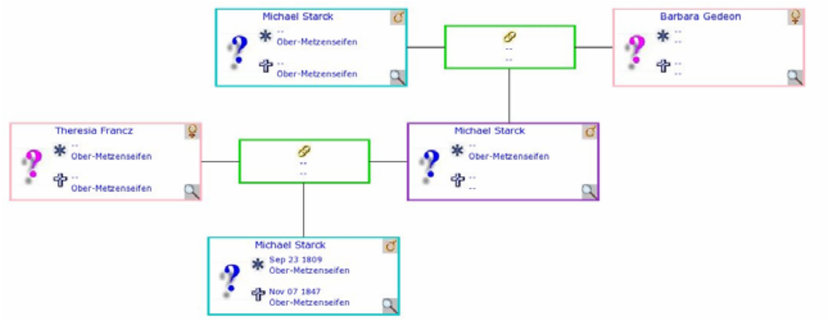
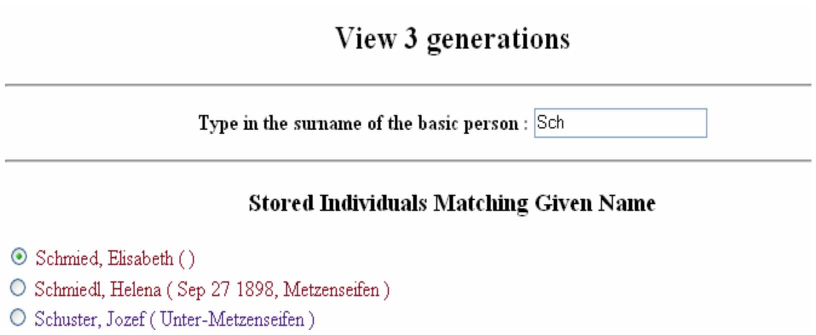


Figure 9. 3 Generation family tree

All these options are implemented relatively in the same way:

The user has initially to choose the basic person, that means the person to which all the others are related. This is done in the following, simple manner:

An input field is presented to the user, where she can type the initial letters of the basic person's surname. A JavaScript function goes then on to asynchronously fetch from the database (utilizing the AJAX technology) all the names matching the letters typed so far. These names are then presented in a list to the user, enabling him to choose the desired one without having to type in the whole name. It is worth mentioning, that the names are coloured according to the sex of the individual (blue for men, red for women) and presented along with their corresponding birth information (date & place). Thus, it is easy for the user to distinguish between persons with exactly or almost the same name. This first stage of choosing the basic person is illustrated in the next screenshot.



View 3 generations

Type in the surname of the basic person : Sch

Stored Individuals Matching Given Name

[✓ Schmied, Elisabeth ( )]
Schmiedl, Helena (Sep 27 1898, Metzenseifen)
Schuster, Jozef (Unter-Metzenseifen)

Figure 10. Choosing the basic person

After choosing the desired person, the user is presented with an image depicting the corresponding family tree. This image consists of rectangles representing individuals and families connected with lines hinting their relationship (married individuals or children of a marriage). These rectangles contain only a summary of the information pertaining to the corresponding entities, namely the name, sex, birth and death info for individuals and ceremony info for marriages.

These images are *actually hmtl maps*, meaning that some parts of it are clickable. Therefore, when the user places the cursor either over a magnifying glass icon or the name of a person, it turns to a hand cursor implying that these areas are active.

In the first case, a new page is presented to the user displaying all information pertaining to the selected entity (either individual or marriage) along with the associated pictures (a kind of photo album).

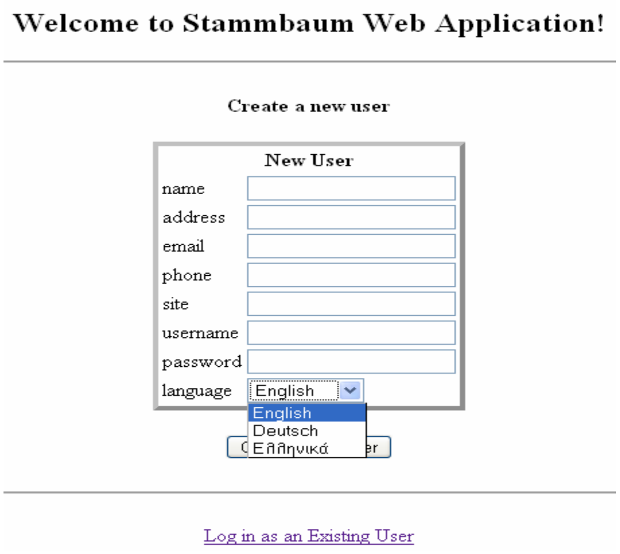
In the second case, the person corresponding to the selected name becomes the basic person. That is, the page reloads displaying a new picture corresponding to new basic person, *thus providing a useful way of going around the family tree of a person*.

- **Edit a family tree**

This option gives the user the chance to edit the stored information pertaining to a person or marriage, so as to either correct wrong data or add new information acquired after the initial insertion of a family tree. To do so, the same form depicted in the above figure 10 is presented to the user so that she can select the basic person just by typing the first few letters of his surname. Following the selection of the basic person, a form is displayed containing all the necessary fields for every individual and marriage related to the basic person (see figures 5 and 6 for illustrative examples of this form). These fields are already filled with the stored information, so that the user normally has to correct or fill only a small numbers of fields.

When the user finishes processing and submitts this form, the fields are verified by a Javascript function and then passed to a PHP script that updates the database.

Apart from the aforementioned capabilities, we should also stress that our application already *supports three languages* (namely English, German and Greek) for ease of use. The user chooses the preferred language while signing up, as it is depicted in the following screenshot.


Welcome to Stammbaum Web Application!

Create a new user

New User

name	
address	
email	
phone	
site	
username	
password	
language	English

Figure 11. Choosing the language of the application

4. Future Work

In the future, we plan to add some more functions to the application, the most important among them being *the implementation of record linkage techniques*. The term „**record linkage**“ refers to the task of finding duplicate entries, that is entries that refer to the same entity (an individual in our case). Important research has already been conducted in this field and some quite effective techniques for detecting duplicate data have been developed. A fairly rich collection of them is implemented in an open source JAVA library, named **SecondString** [1], which we plan to integrate in our application. More information on these methods can be found in article [2].

We also plan to enrich the interface of our application so as to provide the user with additional, useful information regarding her family tree, for example life expectancy of its members, the average couple's age difference and the percentage of divorces.

Acknowledgements

We would like to give special thanks to our colleague Theodoros Kyriakidis for his valuable help in developing this application and of course to DAAD for financing this cooperation.

References

1 http://secondstring.sourceforge.net/

2. Cohen W., Ravikumar P., Fienberg S.: A Comparison of String Distance Metrics for Name-Matching Tasks.In: Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI-2003) Workshop on Information Integration on the Web. (2003)