Extending the Navigation

for Adding Custom Features to ***Delightful Labor***

A ***Delightful Labor*** White Paper
January 30th , 2015

Table of Contents

Overview

This white paper describes the method for adding custom user navigation to ***Delightful Labor***. By extending the base navigation system, developers have the ability to link to features that have been developed to support custom requirements of their non-profit or NGO.

This paper refers to versions of ***Delightful Labor*** released on or after January 30th, 2015.

It is assumed that the reader has familiarity with php, mySQL, object-oriented programming, and the codeIgniter framework. That being said, Be Not Afraid!

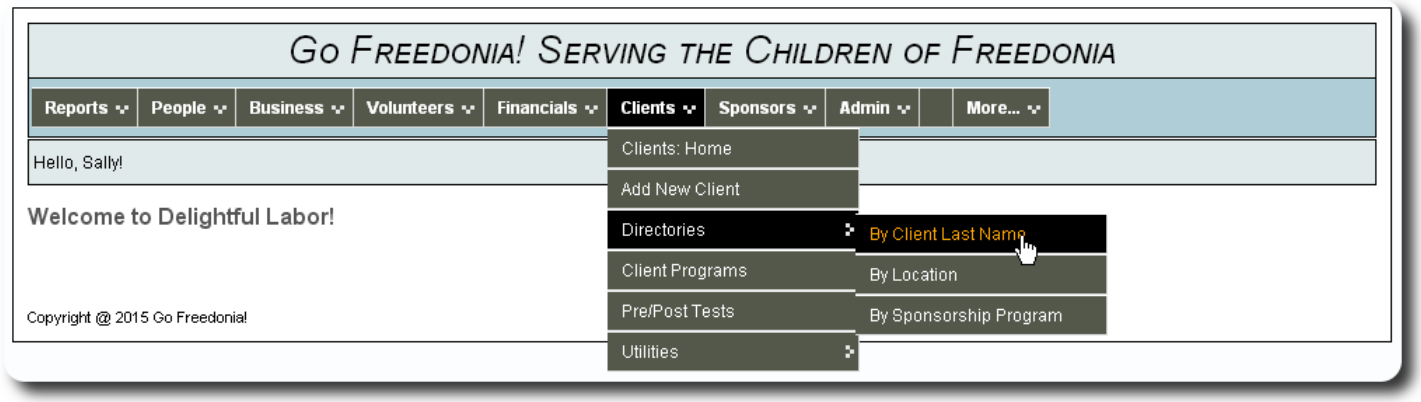
Base Navigation System

Delightful Labor provides base user navigation through three entities:

- CSS-based drop-down tabs at the top of the pane
- A "Home" menu for each functional area
- Bread crumbs near the top of each window

Drop-Down Tabs

The primary navigation for ***Delightful Labor*** is through the drop-down tabs at the top of all forms:



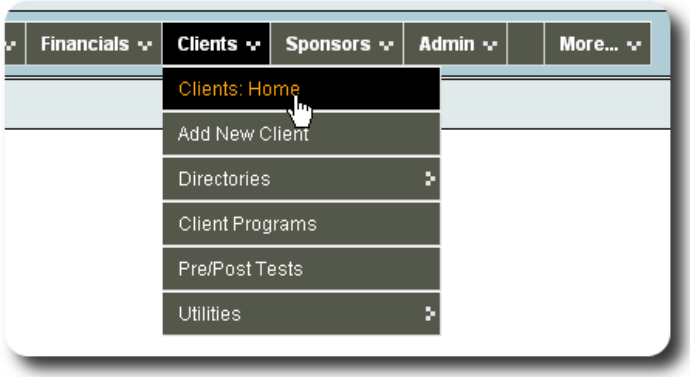
The drop-down menus are implemented via style sheets developed by Live Web Initiatives (`http://www.lwis.net/free-css-drop-down-menu`). The tabs are created in the model file `application/models/mnav_brain_jar.php`

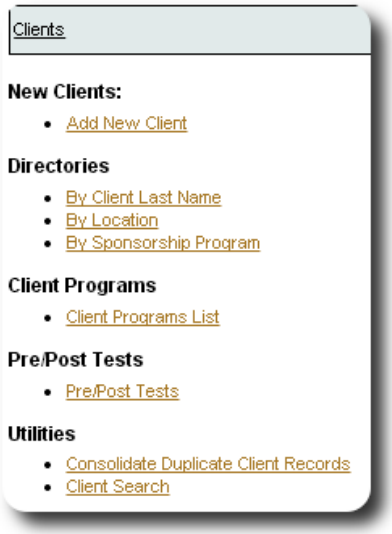
An historical note: ***Delightful Labor*** originally used a spiffy java script-based navigation system developed by "Brain Jar" (`http://www.brainjar.com/`), and worked with every browser in the known universe, except for some versions of Internet Explorer. For the greatest compatibility, we changed the navigation to use a strictly CSS-based navigation.

"Home" Menu

Each top-level tab contains a "Home" link. The home page contains static links to all the features available through the drop-down entries for a given functional area.

This is useful for users who have difficulty navigating the drop-down menus, or for users who wish to see all the available options at one time.





Here is an example of the "Home" page for client features.

The home pages are managed by the simple controller

```
application/controllers/main/menu.php
```

and by views in

```
application/views/main
```

Bread Crumbs

Breadcrumbs are the links near the top of the panel that help provide context for the user, as well as a way to back-track to a higher level in the program.

The method for adding breadcrumbs to your controllers is described below. All breadcrumbs used by ***Delightful Labor*** are made from stone-ground whole grains and are guaranteed to be 100% gluten-free.

The URL and CodeIgniter

It is useful to understand how codeIgniter parses the URL. CodeIgniter examines the URL to determine:

- the path within your controller directory
- the name of the controller module
- the entry point (or function name)
- parameters to be passed to the entry point

For example, with the URL

```
index.php/volunteers/event_date_shifts_add_edit/addEditShift/3/28
```

CodeIgniter will look in directory

```
application/controllers/volunteers
```

for file

```
event_date_shifts_add_edit.php
```

then pass control to routine

```
addEditShift($lEventDateID, $lShiftID)
```

with `$lEventDateID == '3'` and `$lShiftID == '28'` (note that parameters are passed as strings).

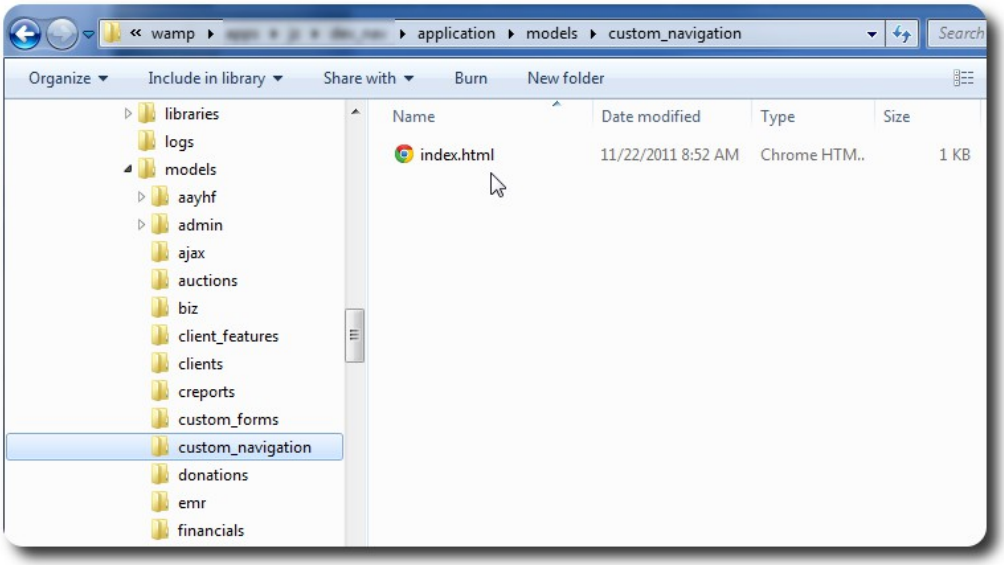
Details can be found on the codeIgniter web site at

```
http://www.codeigniter.com/user_guide/helpers/url_helper.html
```

Adding Custom Navigation

Drop-Down Tabs

To add your own custom drop-down tags, place one or more model php file in directory `applications/models/custom_navigation`.



Initially the folder will be empty, with the exception of the `index.html` file.

The `index.html` File

You will find the `index.html` file in every application directory in ***Delightful Labor***. This file prevents bad guys from browsing your directory contents.

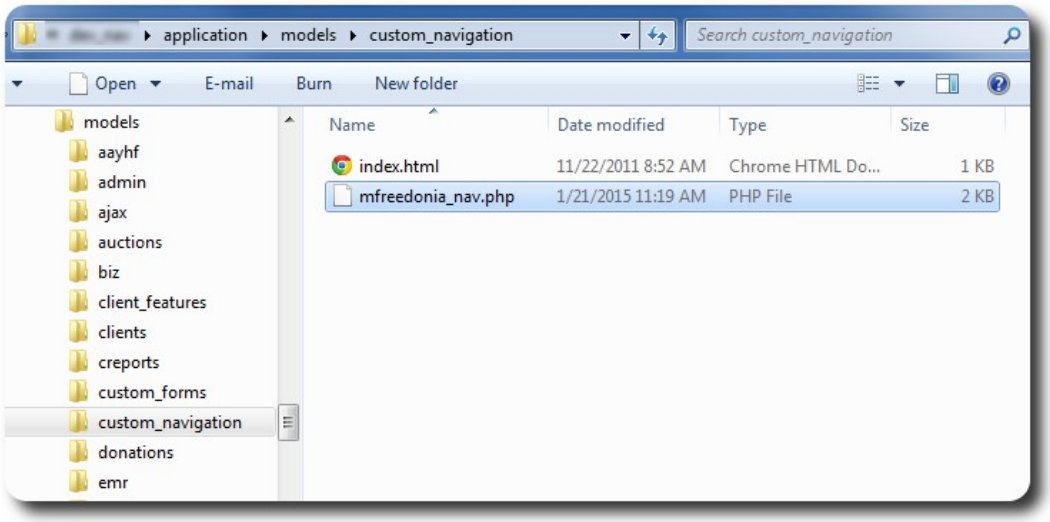
The index.html file typically has the following contents:

```
<html>
   <head>
      <title>403 Forbidden</title>
   </head>
   <body>
      <p>Directory access is forbidden.</p>
   </body>
</html>
```

The Model File

When a user initially logs in, ***Delightful Labor*** looks for models in the `application/models/custom_navigation` folder. If files are found, when ***Delightful Labor*** is preparing the navigation it loads models and adds the string returned from `your-model-name::strNavigation` to the navigation menu.

Here is a sample model navigation file:



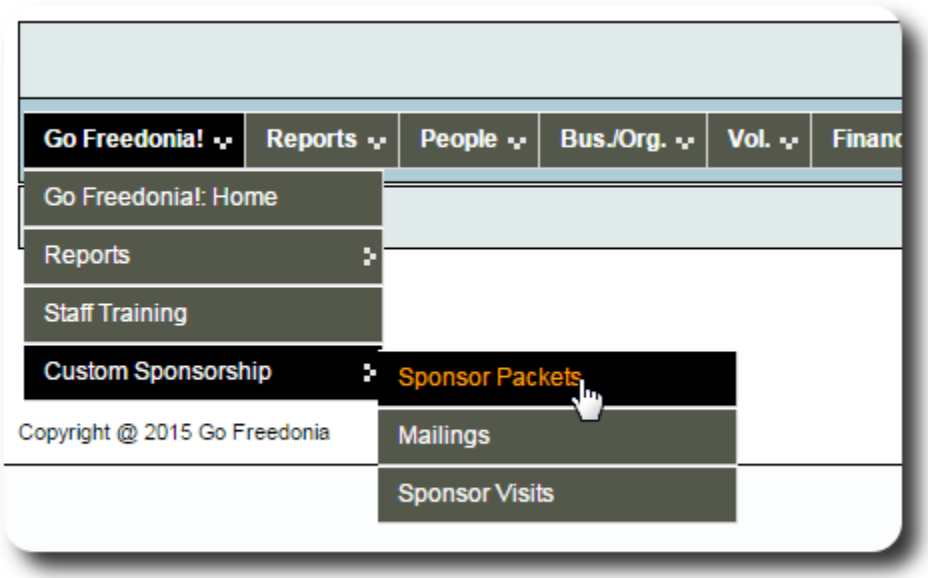

```
<?php

   // the class name must match the file name (excluding the ".php"
class mfreedonia_nav extends CI_Model{

   function __construct(){
      parent::__construct();
   }

   function strNavigation(){
   //---------------------------------------------------------------------
   // the function name must be "strNavigation" - it returns a string
   // with your custom navigation
   //---------------------------------------------------------------------
      $navData =
         '<li class="dir">  // use class="dir" for navigation that will have child links
             Go Freedonia!
             <ul>
                <li>'
                   .anchor('freedonia/home/home_menu/',  'Go Freedonia!: Home').'
                </li>
                <li class="dir">
                   Reports
                   <ul>
                     <li>'
                        .anchor('freedonia/reports/client_review/opts/',  'Client Review').'
                     </li>
                     <li>'
                        .anchor('freedonia/reports/management_rtps/opts/',
                                                              'Management Reports').'
                     </li>
                   </ul>
                   <li>'
                      .anchor('freedonia/staff/training/opts/',  'Staff Training').'
                   </li>
                </li>
                <li class="dir">
```

```
                    Custom Sponsorship
                    <ul>
                       <li>'
                          .anchor('freedonia/sponsorship/packets/opts/',  'Sponsor Packets').'
                       </li>
                       <li>'
                          .anchor('freedonia/sponsorship/mailings/opts/',  'Mailings').'
                       </li>
                       <li>'
                          .anchor('freedonia/sponsorship/visits/opts/',  'Sponsor Visits').'
                       </li>
                    </ul>
                 </li>
              </ul>
           </li>';

      return($navData);
   }
}
```

Notes:

- the class name must match the file name (for example, the class within `mfreedonia_nav.php` must be `mfreedonia_nav`) and must extend the `CI_Model` class
- to ensure compatibility, use only lower-case file names
- the entry point name is `strNavigation()` and returns a string with your navigation information.
- if you receive an error, such as "`Unable to locate the model you have specified: mfreedonia_nav`", check your spelling of the file name and model, and make sure the model extends the `CI_Model` class.
- after making changes to the navigation, users must log out and log back in to access the changes.

Here are the results from the above navigation extension:



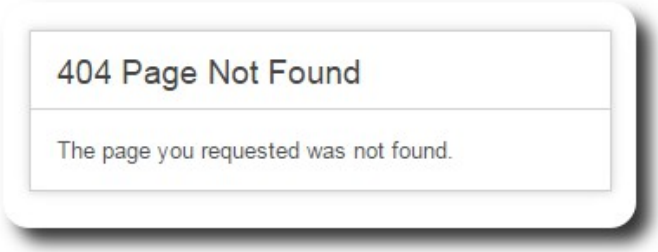
"Home" Menu

In the above example, the first menu is "`Go Freedonia: Home`". To create a home menu, we will add a simple controller file and a view file.

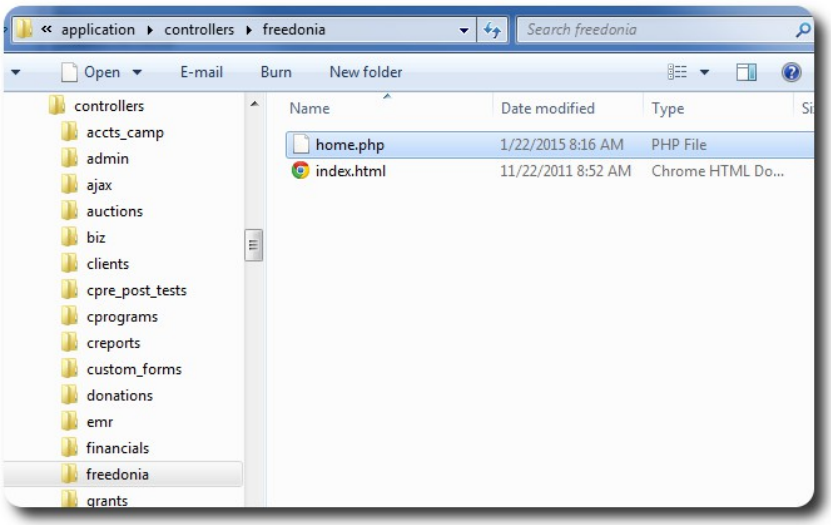
The Controller File

In our above example, if we were to click on:

(which corresponds to the statement `anchor('freedonia/home/home_menu/', 'Go Freedonia!: Home')`), we would receive this message:



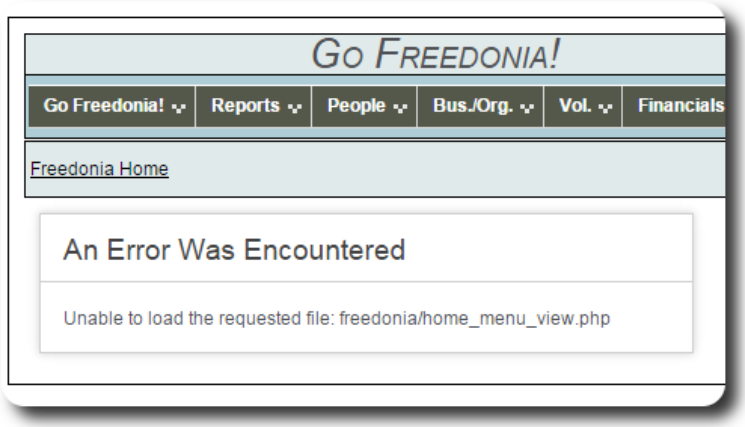
because we have not yet created our controller file. We now create file `home.php` in folder `application/controllers/freedonia`.



The controller file looks like this:

```
<?php

// the class name corresponds to the file name (home.php), as well as the
// segment in the URL
class home extends CI_Controller {

   function __construct(){
      parent::__construct();
      session_start();
      setGlobals($this);
   }

   function home_menu(){
   //-----------------------------------------------------------------------
   // controller to display the Freedonia "home" page. Note how the
   // function name corresponds to the segment in the URL
   //-----------------------------------------------------------------------
         // create the array that will pass information to our view file
      $displayData = array();

         // this will become our browser tab label
      $displayData['title']        = CS_PROGNAME.' | Freedonia';

         // this will become our "bread crumb"
      $displayData['pageTitle']    = anchor('freedonia/home/home_menu',
                                            'Freedonia Home', 'class="breadcrumb"');

         // this loads the navigation system
      $displayData['nav']          = $this->mnav_brain_jar->navData();

         // we specify the name of our view file, which will create the HTML that
         // is ultimately displayed
      $displayData['mainTemplate'] = 'freedonia/home_menu_view';

         // pass the display information to our view
      $this->load->vars($displayData);

         // finally, load the template that will display our home page
      $this->load->view('template');
   }
}
```

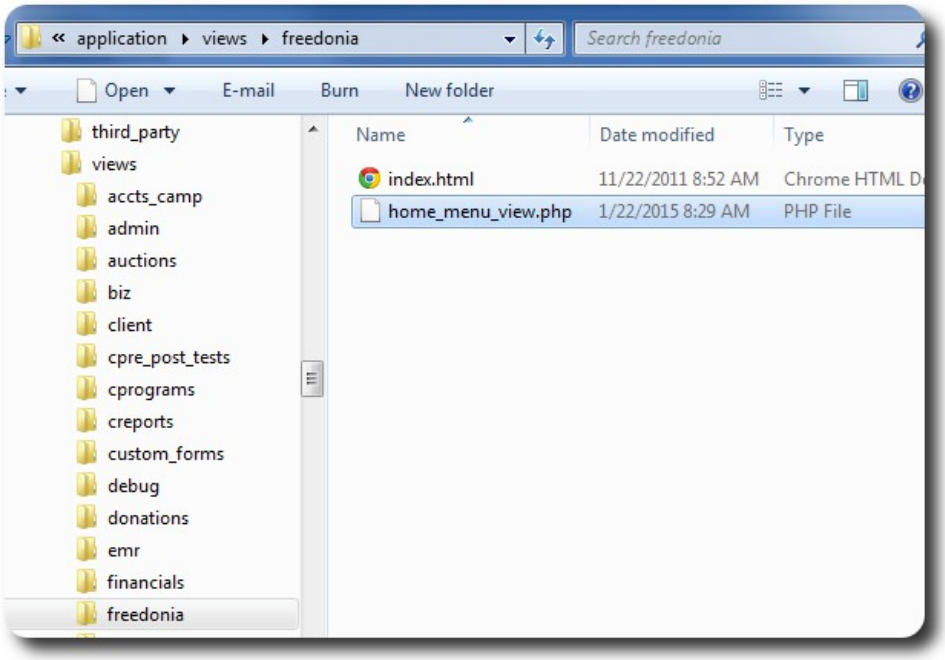
We now click on the `Freedonia: Home` link, and here is what we see:



Wow! A lot of stuff happened! We see our navigation, browser tab, and bread crumbs. But we still have an error - we haven't created our view file yet.

The View File

We now create our view file, as specified in our controller file. We create `home_menu_view.php` in folder `application/views/freedonia`:



The view file contains:

```
<?php
      //--------------------------------------------------------------------------------
      // We now create our html that will be displayed on our form
      //--------------------------------------------------------------------------------
      // echoT is a Delightful Labor utility that trims strings and then "echo's" them,
      // to reduce network bandwidth
      //--------------------------------------------------------------------------------
   echoT('
      <br>
      <b>Go Freedonia!</b>
      <ul>
          <li>'
             .anchor('freedonia/home/home_menu/',  'Go Freedonia!: Home').'
          </li>
          <li>
             <b>Reports</b>
             <ul>
               <li>'
                  .anchor('freedonia/reports/client_review/opts/',  'Client Review').'
               </li>
               <li>'
                  .anchor('freedonia/reports/management_rtps/opts/',
                                                          'Management Reports').'
               </li>
             </ul>
             <li>'
                .anchor('freedonia/staff/training/opts/',  'Staff Training').'
             </li>
          </li>
          <li>
             <b>Custom Sponsorship</b>
             <ul>
                <li>'
                   .anchor('freedonia/sponsorship/packets/opts/',  'Sponsor Packets').'
                </li>
                <li>'
```

```
                .anchor('freedonia/sponsorship/mailings/opts/',  'Mailings').'
            </li>
            <li>'
                .anchor('freedonia/sponsorship/visits/opts/',  'Sponsor Visits').'
            </li>
        </ul>
    </li>
</ul>');
```

Now when we click our link, we see our home page:

Bread Crumbs

Bread crumbs are added in the controller file before control is passed to the view. Here is an example of an extended bread crumb used to add or edit a client's status:

```
$displayData['pageTitle']  =
                anchor('main/menu/client', 'Clients', 'class="breadcrumb"')
          .' | '.anchor('clients/client_record/view/'.$lClientID, 'Client Record',
                                                          'class="breadcrumb"')
          .' | '.anchor('clients/client_rec_stat/viewStatusHistory/'.$lClientID,
                                        'Status History',   'class="breadcrumb"')
          .' | '.($bNew ? 'Add ' : 'Edit ').'Status';
```

This results in the following:

Appendix

References

- Brain Jar: Experiments in Web Programming (`http://www.brainjar.com/`)
- CodeIgniter (`http://www.codeigniter.com/`)
- Free CSS Drop-Down Menu Framework (`http://www.lwis.net/free-css-drop-down-menu/`)

About Delightful Labor

Delightful Labor is a free, open source project created by Database Austin. ***Delightful Labor*** helps non-profits manage their contacts, donations, sponsorships, client programs, silent auctions, and volunteers.

Delightful Labor is available from SourceForge at
`https://sourceforge.net/projects/delightfullabor`

The ***Delightful Labor*** user's guide and contact information is available at
`http://www.delightfullabor.com/`