Final Project: Search Author Online

Zhan Xinyu, 517030910358
Wang Zhongye,
Xie Yichen,
Yue Ye

June 22, 2018

Contents

1 Introduction

1.1 Report Overview

1.2 Special Specification

2 Project Overview

2.1 Website Overview

2.2 Project Structure

2.2.1 Website Structure

Pages Our website mainly consist of 4 types of pages: *Home Page*, *Result Page*, *Information Pages* and *Stats Pages*.

Home Page The entrance of the website. Users can launch queries from here.

Result Page Presents the result of the query, either from the *Home Page* or from the search box in the navigation bar.

Information Page Displays the basic information of one entity (publications, coauthors, affiliations, conferences, etc).

Stats Page Contains the visualization graphs and charts, and recommendations on *Paper Stats*.

These pages are organized into the structure showed in the following chart:

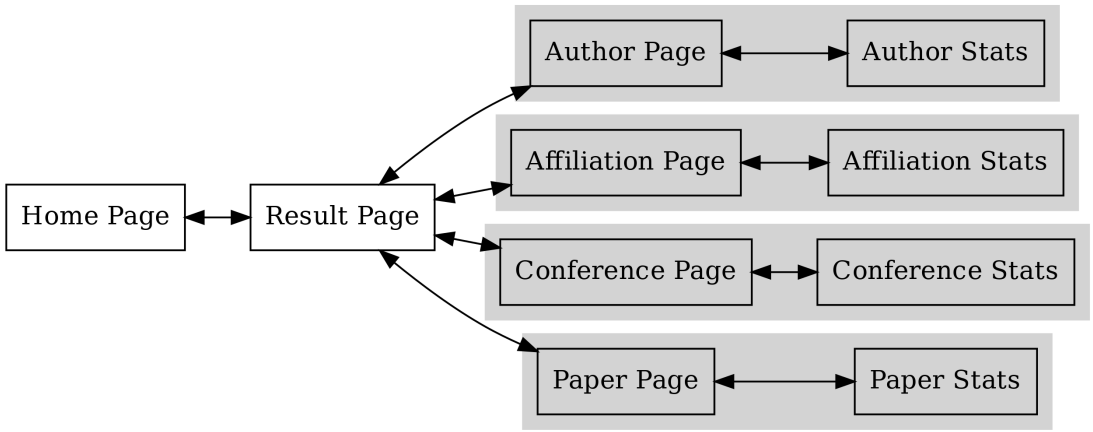


Figure 1: The structure of the website

As the chart shows, users will first visit our *Home Page*, input some keywords, and press Enter. The page will redirect to *Result Page*, and after user click one of the links present there,

it will jump to the *Information Pages*, which includes *Author Page*, *Affiliation Page*, *Conference Page* and *Paper Page*. From the navigation bar on the top of the *Information Pages*, users can direct to the *Stats Pages* corresponding to the *Information Pages*. All pages can return to the *Result Page* of the most recent query, and the *Home Page*.

2.2.2 Directory Structure

In this project we use Code Igniter framework. The framework features MVC design pattern and a bunch of utility and helper funcitons. As a result, we mainly write codes and The directory structure of out project is like following:

application This directory contains the php scripts that runs on the server and fulfill different jobs. These files mainly fall into three category: controllers, models and views, which follows the MVC design pattern.

controllers This directory contains the controllers, which is responsible for dealing with the requests and generating responses.

1. Lab.php
   Static Content Controller for all pages. Also provides dynamic content for text queries.
2. Autocomplete.php
   Dynamic Content Controller for all input box's auto completion.
3. Label.php
   Dynamic Content Controller for label extraction and paper recommendation.
4. Visual.php
   Dynamic Content Controller for visualization graphs and charts.
5. Pagin.php
   Dynamic Content Controller for pagination bar and corresponding js code.

models This directory main contains the models which interacts with our database.

1. Lab_model.php
   Data Provider for all text queries.
2. Visual_model.php
   Data Provider for queries from visualization graphs and charts.
3. Label_model.php
   Data Provider for label extraction and paper recommendation.

views The directory contains multiple sub-directories: Lab, Multi_pagin, Shared, Templates.

1. Lab
   This directory contains the "visible" part of the website. It has multiple subdirectories, containing **page frameworks** and **page contents** for different kinds of entities, for example authors or affiliations. Besides, it has two individual php files: Home.php, the *Home Page* of the website; Result.frame.php, the framework all-in-one search *Result Page*.

2. Multi_pagin
   This directory contains templates of javascript files used in the pagination system. The controller will process the php files and insert them into the html, and they will be working as ordinary js scripts.

3. Shared
   This directory contains the php files for the navigation bar in the top, and some legacy files.

4. Templates
   This directory contains the header and the footer. They will be embeded in every web page. *header.php* mainly includes the libraries, and *footer.php* add a banner in the bottom to show the credits.

assets This directory contains the css files and js scripts that will be sent to the client side and processed by the browser.

css This directory contains the cascading style sheets.

1. Theme.css
   This is the style sheet used by any page you can reach in the website. It defines the the color patterns, fonts and link styles.

2. Search_selection.css
   This is used by the *Home Page*. It defines the color patterns of the search boxes.

3. Visual.css
   This is the style sheet used in the visualization part, in particular the force directional graph.

js This directory contains the javascipt files.

1. Animation.js
   This script adds animations of the search boxes, so that users can collapse and un-collapse them as they like.

2. Authocomplete.js
   This script uses jquery to realize ajax search box autocompletion .

3. D3.layout.cloud.js, myplots.js
   These scripts defines the behavior of the visulization charts and graphs.

4. sidebar.js
   This script animates the side bar.

5. Pagin.js
   This script realizes the basic functionality of the pagin system. Itself may not be refernced frequetly as it only supports one pagin bar on one webpage, but it is also the prototype of the dynamic pagin system, which includes the full functionality of the pagin bar and content boxes.

Overview:

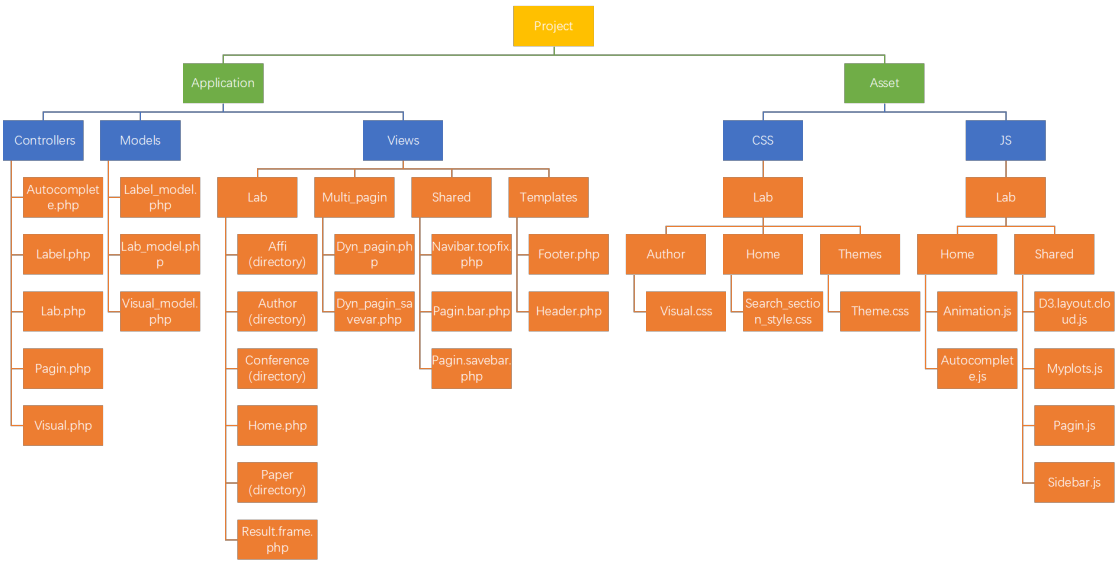
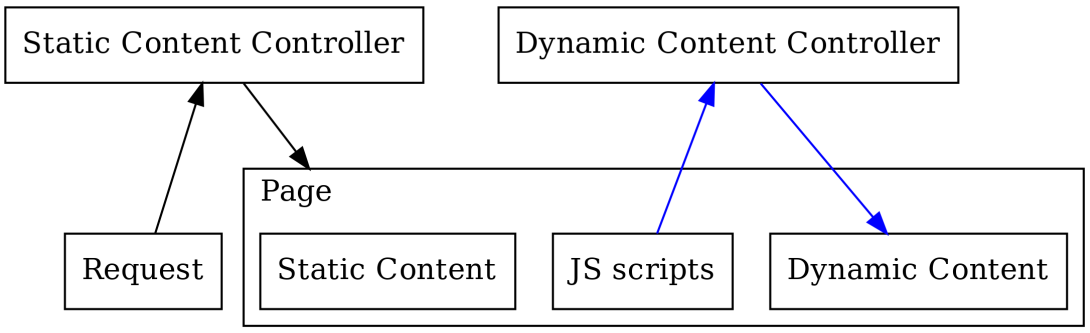


Figure 2: Directory Structure

2.2.3 Request Process Procedure: Page framework & Page Content

The *Result Page*, *Information Pages* and *Stats Pages* all involves many dynamic content loaded by js scripts from client side. To keep things simple, we design a standard and uniform request process procedure, and write some utilitiy functions for it. The procedure is showed as follows:

Figure 3: Request Process Procedure



As the chart shows, whenever the server receives a request, it will first be directed to the static content controller, and the controller will give back the web page containing the static content and the js scripts. We name these things as **page "framework"s**. Then the browser interprets

the js script, making new requests to the dynamic content controller. After the browser fetches data from the remote, it will present them in the page by ajax, without reloading the whole page. These data or html files are named **page "content"s**.

2.3 Project Organization

2.3.1 Workload Division

Zhan Xinyu Project structure; Website backend; Utility functions.

Wang Zhongye Webpage Design; Website frontend, including css and js; Visualization charts.

Xie Yichen Label Extraction; Page Recommendation.

Yue Ye Visualization backend; Slides.

2.3.2 Collaboration with Git and Github

To share code and synchronize

2.3.3 Third party packages and libraries

2.3.4 Website Deploy

3 Frontend Implementation

3.1 Overview

3.2 UI Design

3.3 Data Visualization

4 Functionality Implementation

4.1 Backend Implementation

In this section we mainly want to talk about how we present the content on server side to the browser running on the client side.

What we have come up with is a set of utility functions, which can be combined and assembled to realize the pagin systemm with which we can process the request and ship the paginated content to the browser.

Pagination Bar Generator

dyn_pagin.php

First of all, we need to write the js code to bind events on the elements. Realizing that there may be more than 1 pagination bar and more than 1 content fetcher in one webpage, we use id to distinguish them so that they can work independently without disturbing each other.

But problem comes: how can we generate the js scripts for different ids? Our solution is to use js script templates.

dyn_pagin.php is a js script template. Before the server send this file to the browser, the php interpreter will process it and fill the $prefix variable in the variable names. After doing so, it will turn into an html file with a well-functioning js script in it. Then the html file will be sent to the browser with the page framework together.

Following is the first part of the js script template. In this part, a function named <$prefix>_pagin_goto is defined, which launched a request with the local variables stored in the page.

```
function <?php echo $prefix; ?>_pagin_goto() {
  var table_target = <?php echo $prefix; ?>_content_fetch_url +
  "page=" + <?php echo $prefix; ?>_page +
  "&mode=" + <?php echo $prefix; ?>_field[0] +
  "&field=" + <?php echo $prefix; ?>_field[1];
  $.get(table_target, function(result){
    $("#" + <?php echo $prefix; ?>_content_handle).html(result);
  });

  var pagin_target = <?php echo $prefix; ?>_pagin_fetch_url +
  "handle=" + <?php echo $prefix; ?>_pagin_handle +
  "&page=" + <?php echo $prefix; ?>_page +
  "&startpage=" + <?php echo $prefix; ?>_startpage +
  "&endpage=" + <?php echo $prefix; ?>_endpage +
  "&width=" + <?php echo $prefix; ?>_width;
  $.get(pagin_target, function(result) {
    $("#" + <?php echo $prefix; ?>_pagin_handle).html(result);
  });
}
```

Following is the second part of the js script template. In this part, the click events are dynamically binded on the pagination bar elements and realizes the callback functions after the events happen.

The callback function mainly do three things: first extract the target page stored either in the local variables or in the attributes of the pagination bar items, then check the conditions and update the local variables, and finally launched requests to the remote.

```
// bind event: load content when the page is loaded
$(<?php echo $prefix; ?>_pagin_goto);

// bind click event: show the content of the first page when the button
    is pushed down
$("#" + <?php echo $prefix; ?>_pagin_handle).on(
'click',
'a#' + <?php echo $prefix; ?>_pagin_handle + '-begin',
function() {
  <?php echo $prefix; ?>_page = <?php echo $prefix; ?>_startpage;
  <?php echo $prefix; ?>_pagin_goto();
});

//...

// bind click event: show the content of the given page when the button
    is pushed down
```

```
16    $("#" + <?php echo $prefix; ?>_pagin_handle).on(
17    'click',
18    'a#' + <?php echo $prefix; ?>_pagin_handle + '-item',
19    function() {
20      <?php echo $prefix; ?>_page = $(this).attr('pagenum');
21      <?php echo $prefix; ?>_pagin_goto();
22    });
```

View: dyn_pagin_savevar.php

Similar to the js script template above, *dyn_pagin_savevar.php* is also a script template. It stores the local variables in the webpage.

```
1     <script>
2     var <?php echo $prefix; ?>_page = <?php echo $page; ?>;
3     var <?php echo $prefix; ?>_startpage = <?php echo $startpage; ?>;
4     var <?php echo $prefix; ?>_endpage = <?php echo $endpage; ?>;
5     var <?php echo $prefix; ?>_field = ["<?php echo $field['mode']; ?>", "<?
          php echo $field["field"]; ?>"];
6     var <?php echo $prefix; ?>_width = <?php echo $width; ?>;
7     var <?php echo $prefix; ?>_pagin_handle = "<?php echo $pagin_handle; ?>";
8     var <?php echo $prefix; ?>_pagin_fetch_url = "<?php echo $pagin_fetch_url
          ; ?>";
9     var <?php echo $prefix; ?>_content_handle = "<?php echo $content_handle;
          ?>";
10    var <?php echo $prefix; ?>_content_fetch_url = "<?php echo
          $content_fetch_url; ?>";
11    </script>
```

Controller: Lab.php

We have added a utility function to create pagination bars and load the js script templates. It works as an intermediate layer to realize the functionality in a more compact and cleaner way.

This function stores the paginination information(startpage, endpage, current page), style information(bar length, item number), id information(on which element the events will be binded) and route information(the url js will send request to).

```
1     require "Pagin.php";
2     \\...
3     private function _makeup_dyn_pagin(
4     $prefix, $mode, $field, $total, $nitem, $width,
5     $c_hdl, $p_hdl, $c_url, $p_url, &$data)
6     {
7       $data['prefix'] = $prefix;
8       $data['field'] = array(
9       "mode" => $mode,
10      "field" => $field);
11      $data['total_result'] = $total;
12      Pagin::_fill_pagenum(1,
13      ceil($total / $nitem),
14      $data);
```

```
15      $data['width'] = $width;
16      $data['content_handle'] = $c_hdl;
17      $data['pagin_handle'] = $p_hdl;
18      $data['content_fetch_url'] = $c_url;
19      $data['pagin_fetch_url'] = $p_url;
20
21      $this->load->view('multi_pagin/dyn_pagin_savevar.php', $data);
22      $this->load->view('multi_pagin/dyn_pagin.php', $data);
23    }
```

Besides the utility function to help create pagination bars and load script templates, we need more functions to provide the data the users are really interested in. Then we write a set of functions beginning with "view_", which returns the html files that will be dynamically loaded to the page.

Mainly there are two kinds of these, one is the "view_<entity>_table", which is used in the main panel, and the "view_<entity>_content", which is used in the sidebar.

Following is an example of authors:

```
1     public function view_author()
2     {
3       $data['title'] = "Author Page";
4       $data['author_id'] = $this->input->get('author_id');
5       $data['author_name'] = $this->Lab_model->get_author($data['author_id'])
            ['AuthorName'];
6       $data['item_url'] = get_cookie('query_url');
7
8       $this->load->view('templates/header', $data);
9       $this->load->view('shared/navibar.topfix.php', $data);
10      $this->load->view('lab/author/author.frame.php', $data);
11
12      // dyn pagin loading
13      $this->_makeup_dyn_pagin("main", "author_pub", $data['author_id'],
14      $this->Lab_model->get_author_pub_total_number($data['author_id']),
15      $this->res_per_page, 10,
16      "lab-author-main-panel" ,"lab-author-main-pagination",
17      base_url()."lab/view_author_content?",
18      base_url()."pagin/pagin_bar?", $data
19      );
20      $this->_makeup_dyn_pagin("affi", "author_affi", $data['author_id'],
21      $this->Lab_model->get_author_affi_total_number($data['author_id']), 5,
            5,
22      "lab-author-sidebar-af-items", "lab-author-sidebar-af-pagination",
23      base_url()."lab/view_author_content?",
24      base_url()."pagin/pagin_bar?", $data
25      );
26      $this->_makeup_dyn_pagin("conf", "author_conf", $data['author_id'],
27      $this->Lab_model->get_author_conf_total_number($data['author_id']), 5,
            5,
28      "lab-author-sidebar-cf-items", "lab-author-sidebar-cf-pagination",
29      base_url()."lab/view_author_content?",
30      base_url()."pagin/pagin_bar?", $data
31      );
32      $this->_makeup_dyn_pagin("coau", "author_coau", $data['author_id'],
```

```
33      $this->Lab_model->get_author_coau_total_number($data['author_id']), 5,
            5,
34      "lab-author-sidebar-ca-items", "lab-author-sidebar-ca-pagination",
35      base_url()."lab/view_author_content?",
36      base_url()."pagin/pagin_bar?", $data
37      );
38      $lbl_count = $this->Label_model->fetch_author_label($data['author_id'
            ],0,0)['num'];
39      $this->_makeup_dyn_pagin("labl", "auth_labl", $data['author_id'],
40      $lbl_count, 5, 5,
41      "lab-author-sidebar-lb-items", "lab-author-sidebar-lb-pagination",
42      base_url()."label/view_author_label?",
43      base_url()."pagin/pagin_bar?", $data
44      );
45      $this->load->view('templates/footer');
46    }
47
48    public function view_author_content()
49    {
50      $mode = $this->input->get('mode');
51      $author_id = $this->input->get('field');
52      if ($mode == "author_pub") {
53        $page = $this->input->get('page');
54        if($page===null || $page <= 0)
55        $page = 1;
56        $pagenum = ($page - 1) * $this->res_per_page;
57
58        $retrieve = $this->Lab_model->search_paper_of_author($author_id,
              $pagenum, $this->res_per_page);
59        foreach($retrieve as $index => $content) {
60          $retrieve[$index]['ConferenceName'] =
61          $this->Lab_model->get_conference($content['ConferenceID'])['
                ConferenceName'];
62          $retrieve[$index]['AuthorList'] = $this->Lab_model->
                search_author_of_paper($content['PaperID']);
63        }
64
65        $data['query_result'] = $retrieve;
66        $data['offset'] = $pagenum;
67        $data['AuthorListType'] = "link";
68        $data['NeedConference'] = true;
69        echo $this->load->view('lab/paper/paper.table.php', $data, true);
70      } else if ($mode == "author_affi") {
71        $page = $this->input->get('page');
72        if($page===null || $page <= 0)
73        $page = 1;
74        $pagenum = ($page - 1) * 5;
75
76        $retrieve = $this->Lab_model->search_affi_of_author(
77        $author_id, $pagenum, 5);
78        $data['query_result'] = $retrieve;
79        $data['offset'] = $pagenum;
80        echo $this->load->view('lab/affi/affi.content.php', $data, true);
```

```
81      } else if ($mode == "author_conf") {
82        $page = $this->input->get('page');
83        if($page===null || $page <= 0)
84        $page = 1;
85        $pagenum = ($page - 1) * 5;
86
87        $retrieve = $this->Lab_model->search_conf_of_author(
88        $author_id, $pagenum, 5);
89        $data['query_result'] = $retrieve;
90        $data['offset'] = $pagenum;
91        echo $this->load->view('lab/conference/conference.content.php', $data
              , true);
92      } else if ($mode == "author_coau") {
93        $page = $this->input->get('page');
94        if($page===null || $page <= 0)
95        $page = 1;
96        $pagenum = ($page - 1) * 5;
97
98        $retrieve = $this->Lab_model->search_coau_of_author(
99        $author_id, $pagenum, 5);
100       $data['query_result'] = $retrieve;
101       $data['offset'] = $pagenum;
102       echo $this->load->view('lab/author/coauthor.content.php', $data, true
              );
103     }
104   }
```

Views: <entity>.table.php & <entity>.content.php

As metioned above, these php files are part of the ”page contents”. Still, we use author as an example:

author.table.php (main panel)

```
1   <?php foreach($query_result as $index => $author_item): ?>
2   <?php if ($index != 0) { ?>
3     <hr>
4     <?php } ?>
5   <div class="panel-item-title-0 panel-dark-body" >
6   <?php echo $offset + $index + 1; ?>.  
7   <a href="<?php echo base_url().'lab/view_author?author_id='.$author_item
        ['AuthorID']; ?>" >
8   <?php echo ucwords($author_item['AuthorName']); ?>
9   </a>
10  </div>
11  <div class="panel-item-content-0 panel-dark-body" >
12       
13  Author ID: <?php echo $author_item['AuthorID']; ?>
14    -  
15  Paper Published: <?php echo $author_item['PaperNum']; ?><br>
16       
17  Major Affiliation: 
18  <?php if ($author_item['AffiliationName'] != null) { ?>
```

```
19      <a href="<?php echo base_url().'lab/view_affi?affi_id='.$author_item['
            AffiliationID']; ?>" >
20      <?php echo ucwords($author_item['AffiliationName']); ?>
21      </a>
22      <?php }else{ echo "--"; }?>
23    </div>
24
25    <?php endforeach; ?>
```

author.content.php (side bar)

```
1   <?php foreach($query_result as $index => $author_item): ?>
2   <div class="list-group-item" >
3   <a href="<?php echo base_url().'lab/view_author?author_id='.$author_item
        ['AuthorID']; ?>"
4   ><?php echo ucwords($author_item['AuthorName']); ?></a>
5   </div>
6   <?php endforeach ?>
```

We can see that these are just ordinary html files. As a result, they can be directly inserted to the existing webpage by jquery.

4.2 Visualization Implementation

4.3 Label Extraction

4.4 Paper Recommendation

5 Future Work

6 Conclusion