DEVELOPING A WORDPRESS PLUGIN

With the previous knowledge about database design, SQL and PHP, now we are able to develop a WordPress plugin. We have already developed the “Lexis Radar-Version 1.0”, and soon new functions and data will be added to make it a more powerful and useful plugin.

1. Basic Introduction to Lexis Radar

As it is shown in our proposal, Lexis Radar is a plugin aiming at providing explanations of different words in a WordPress blog, especially GRE vocabulary. (see: http://toyhouse.cc/group/lexisradar/page/proposal-of-team-lexisradar) In the later development of the plugin, we found that the usage of Lexis Radar should not be confined to GRE vocabulary. Every explanation that each blogger wishes to show to his or her visitors should be added to the database so that it can be displayed in the blog. The first version of Lexis Radar offers a database with GRE vocabulary and the morphs of each word. It also allows users to add their own data so that it is a fairly flexible plugin. Later we are also going to provide other kinds of database, such as images, URLs, etc.

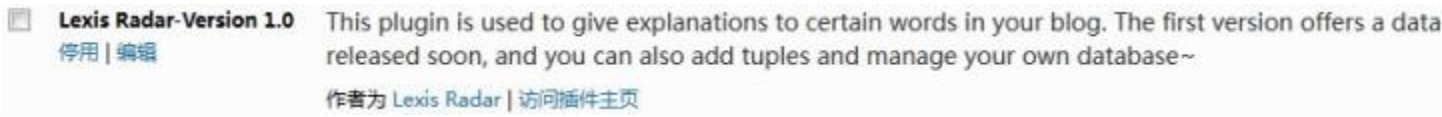
2. WordPress Plugin Development

It is easy to develop a WordPress plugin, since WordPress has already offered many examples and explanations in the API. Here I want to share some important part of the plugin development with you.

(1) File headers.

All WordPress should have file headers at the beginning of the main PHP file to illustrate the basic information about the plugin and the authors. This information will be showed at the plugin management interface of WordPress and thus it is required. Our file header and its effect are showed below:

```
<?php
/*
Plugin Name: Lexis Radar-Version 1.0
Plugin URI: http://LexisRadar.sinaapp.com
Description: This plugin is used to give explanations to certain words i
Author: Lexis Radar
Author URI: http://LexisRadar.sinaapp.com
*/
```



(2) Database Operations.

The entire tables you use in a plugin should be located in the default database of WordPress. After correctly deploy your database, we can easily operate them with the class *wpdb* offered by WordPress. It provides a class of functions of all database manipulations. We have to use the global variable $wpdb, an instantiation of the class, in our PHP file. Its function can basically satisfy all our need. The

following example shows two function of $wpdb: implementation of an insertion and return the result of selection.

```
//Insert the new tuple to vocabulary_list.
$add1="INSERT INTO ".$lexisradar_table_name." (GRE_ENG, GRE_CHN) VALUES ('".$word."','".$
$wpdb->query($add1);

//Get the inserted ID from vocabulary_list.
$get_id="SELECT max(GRE_ENG_ID) FROM vocabulary_list ";
$id=$wpdb->get_var($get_id);
```

(3) Edit the Menu of Plugin.

Noticing that the WordPress management interface as a menu of each plugin, which is important for the interaction with the users. Additionally, with these menus, users are able to better manage the data. Therefore, it is of great significance to edit such pages. WordPress offers an extremely easy way to achieve this. Simply call the functions named add_menu_page and add_submenu_page, then provide the title and the PHP files to be loaded, we create a menu for our plugin. Here is my example:

```
/*
 * Establish the menu for Lexis Radar.
 */
function lexisradar_menu()
{
    //Add a top level menu page.
    add_menu_page('Manage','Lexis Radar','manage_options','lexisradar/lexisradar-manage.php');

    //Add submenu pages.
    add_submenu_page('lexisradar/lexisradar-manage.php','Manage','Manage','manage_options','lexisr
    add_submenu_page('lexisradar/lexisradar-manage.php','Option','Option','manage_options','lexisr
}
```

(4) Connect to Plugin Hooks.

Many WordPress plugins accomplish their goals by connecting to one or more WordPress plugin hooks. It is like defining an event trigger for the page. If one action in WordPress is detected, a corresponding event in the plugin is triggered, thus certain goals are achieved. There are different hooks in WordPress, and appropriately use them we can implement the goal of our plugin.

Here is an example for our Lexis Radar. The core function of this plugin is to compare all the word in an article with the data in the database, and then determine the words in the database and prepare the explanation for these words for further uses. WordPress offers a hook called filter, which can modify the text before sending it to the browser. I used this hook in the following way:

```
/*
 * the_content displays the contents of the current post.
 * Reference: http://codex.wordpress.org/Function_Reference/the_content
 */
add_filter('the_content', 'add_lexisradar_words');
```

As it is showed, before the_content is executed, that is displaying the content of an article in a browser, WordPress will check if any hooks are assigned to this function. I wrote a function add_lexisradar_words to compare all the words in the article with the data, and it is found to registered to this function, thus add_filter is executed and the content to be displayed is modified by add_lexisradar_words. Other plugin hooks can be used in the similar way, and developing WordPress plugins becomes an interesting work.

(5) Use CSS and JavaScript.

To beautifully display the result of the searching result, we should apply CSS and JavaScript appropriately. Typically these modifications are embedded directly in the HTML file or PHP file, but there may be some minor mistakes that ruin the whole work, like the lecture by Shi Peiyu of this week. So as far as I'm concerned, using an external CSS or JavaScript file is a more safe way. WordPress offers safe ways to add such files to a WordPress generated page. The following is my usage of a JavaScript file to create a fade-in and fade-out effect.

```
//Include the javascript file.
wp_enqueue_script('jquery');
wp_enqueue_script('', $lexisradar_plugin_url.'javascript/scripts.js');
```

3. A First Shot of WordPress MU

Our first design of Lexis Radar is that when one wants to see the explanations provided by it, it can be achieved for all the blogs. However, it seems to be a little bit difficult since it is not possible to force all the bloggers to install this plugin. After discussing with Mr. Koo on Thursday, I came up with a relatively simple way. If many bloggers register on a same server, it is possible that we can SUGGEST them to install this plugin so that it can help readers better understand their articles. This requires the usage of WordPress MU.

Source: http://toyhouse.cc/profiles/blogs/personal-report-of-week-7-1