WikiNearby

- Wordpress plugin -

Relazione del progetto d'esame per il corso
Progettazione e Produzione Multimediale [B003712]

Biagini Diego

Poggiani Alessio

Università degli studi di Firenze
A.A. 2019/2020

Introduzione

Il progetto che è stato realizzato è un plugin per il CMS Wordpress, che abbiamo scelto di chiamare WikiNearby.
Lo scopo di questo plugin è quello di offrire agli sviluppatori di un sito web incentrato sul turismo la possibilità di inserire un widget dentro a post o pagine Wordpress per visualizzare i luoghi di interesse culturale vicini ad un luogo geografico inserito a priori.

Per poter recuperare le informazioni circa i luoghi di interesse sarà consultato il database di articoli Wikipedia, questo è fatto attraverso una particolare API di Wikipedia, GeoData, la quale si occupa principalmente di organizzare certe pagine(per esempio di monumenti,musei, etc.) assegnandogli delle coordinate.
Oltre a questo permette di consultare informazioni sui luoghi vicini a una coordinata geografica inviata dall'utente.

Le tecnologie usate per la realizzazione di questo applicativo sono tutte quelle trattate a lezione, ovvero PHP e Wordpress API per realizzare l'interfacciamento tra utente Wordpress e plugin, nonchè HTML5 e CSS3 per la visualizzazione dell'applicativo e infine Javascript, JQuery e AJAX per rendere interattivo lo strumento e per recuperare le informazioni necessarie da Wikipedia.

Il lavoro è stato diviso principalmente in due parti:

- Realizzazione di una pagina "statica" che adempie al suo obiettivo a fronte di un luogo definito, cioè visualizzare i luoghi di interesse vicino ad esso
- Realizzazione dell'infrastruttura di plugin Wordpress che consente al proprietario di un sito di aggiungere molteplici luoghi, per poter poi visualizzare lo widget applicato su quel particolare luogo

Infine le due parti sono state fuse insieme per rendere il plugin funzionante.

Back End

Il back end si occupa principalmente di acquisire le informazioni su luoghi da parte dell'utente(proprietario del sito web), salvarli dentro il database di Wordpress e successivamente consultare i dati salvati per mostrare a chi visita il sito il widget front end.
Per individuare il luogo dove sarà mostrato l'applicativo è stato usato uno shortcode, inseribile in qualsiasi luogo in cui si voglia visualizzarlo(per esempio post, pagine, sidebar, etc.).

Salvataggio dei dati

Per conservare i dati dei luoghi inseriti dall'utente sono usate due classi:

- `Location`: mantiene le informazioni di un singolo luogo(latitudine, longitudine, nome del luogo, immagine del luogo, altre impostazioni) dentro l'array associativo `$loc_data`, il quale contiene anche un suo codice identificativo.
  Espone i metodi `display_table()`, usato per mostrare una riga della tabella sul pannello di controllo Wordpress che conterrà tutte le Location, `display()`, usato per mostrare l'applicativo quando necessario, e `set_id(id)`, per assegnargli un identificativo.

- `Saved_Locations`: mantiene un array di oggetti di tipo `Location` e offre i metodi CRUD per gestire questa collezione, inoltre offre un metodo, `print_locations_table()` per stampare una tabella contenente informazioni di tutte le Location.
  Per gestire la collezione viene registrato un id progressivo, `$prog_id`, che serve a identificare ogni singola Location nell'array `$locations`, questo è incrementato ad ogni inserimento di un nuovo luogo

Per registrare effettivamente questi dati sul database Wordpress è stata usata la Options API [1], in particolare le funzioni `add_option( option_name , object)`, `remove_option(option_name)`, `update_option(option_name , object)`, per aggiungere, rimuovere, aggiornare un dato, mentre la funzione `get_option(option_name)` per leggerlo.

Plugin Setup

Uno dei primi accorgimenti da prendere quando viene scritto un plugin Wordpress è quello di scrivere i metodi necessari alla sua attivazion/disattivazione e e successivamente registrarvi l'hook [2] appropriato.

Nel nostro caso vogliamo che quando il plugin è attivato, se non è presente l'oggetto `Saved_Locations` nel database allora esso sarà aggiunto.
Inoltre se l'utente decide di disinstallare il plugin sarà necessario rimuovere questo dato dalla memoria.

```
1 register_activation_hook( __FILE__, 'wikinearby_activate' );
2 register_deactivation_hook( __FILE__, 'wikinearby_deactivate' );
3 register_uninstall_hook( __FILE__, 'wikinearby_uninstall' );
4
5 // Check if option is in the DB, if not create it
6 function wikinearby_activate(){
7     $saved_locations = get_option('wikinearby_saved_locations');
8
```

```
9       if($saved_locations === false){
10          $sav = new Saved_Locations();
11                  add_option("wikinearby_saved_locations", $sav);
12          }
13  }
14
15  // Do nothing for now
16  function wikinearby_deactivate(){}
17
18  // Delete data
19  function wikinearby_uninstall(){
20          delete_option("wikinearby_saved_locations");
21  }
```

Menu pages

L'interfaccia utente principale del plugin è offerta tramite i cosiddetti menu[3], aggiunti nel seguente modo.

```
1  add_action( 'admin_menu', 'wikinearby_menu_page' );
2
3  function wikinearby_menu_page(){
4      add_menu_page('WikiNearby','WikiNearby','manage_options','wikinearby-menu',
5      'wikinearby_menu',plugin_dir_url(__FILE__ ).'assets/icon.png');
6      add_submenu_page( 'wikinearby-menu', 'Add new location', 'Add new location',
7       'manage_options', 'edit-location-submenu', 'edit_location_submenu');
8  }
```

Le funzioni che mostrano i due menu sono state scritte nei file `menu.php` e `edit_location_submenu.php`.

Il menu principale si occupa solamente di leggere le `Location` memorizzate e mostrare una tabella corrispondente ad esse(col metodo di `Saved_Locations` appropriato), rendendo possibile la modifica e eliminazione di singoli luoghi.
Il menu secondario invece viene usato quando l'utente ha la necessità di aggiungere o modificare una `Location`.
Esso offre un form in cui è possibile inserire i dati di un nuovo luogo, in particolare è anche possibile aggiungere le coordinate individuandole con una mappa(il file javascript corrispondente fa uso del plugin JQuery locationpicker [4]).

Questa pagina viene usata anche per modificare un luogo inserito precedentemente, per far questo deve essere inviata una richiesta GET con l'id del luogo da modificare come parametro(queste richieste sono normalmente eseguite premendo il bottone modifica di fianco a un luogo, nel menu principale).

In questo caso i campi del form saranno preimpostati ai valori precedenti alla modifica:

```
// Check if it came from the edit link
//if this is true initialize the fields to the given Location
$editing = false;
$selected_location = null;
if($_GET['id'] != 0) {
        $editing = true;

        $saved_locations = get_option('wikinearby_saved_locations');
        $selected_location = $saved_locations->get_location_by_id($_GET['id']);
        ...

}
...
//Location name
<input style="max-width:30ch" class="widefat" id="loc_name" name="loc_name" type="text"
value="<?php echo($editing ? $selected_location->loc_data['loc_name'] : 'Location'); ?>"
 required>
...
```

Aggiornamento delle informazioni salvate

Per eseguire le operazioni di creazione, rimozione, modifica dei luoghi sono state usate le post action di Wordpress [5].
Queste sono dei particolari tipi di hook che permettono di gestire richieste GET o POST all'interno del sito.
Un esempio del loro uso è il seguente:

```
function wikinearby_edit_location(){
    //Create new location
    unset($_POST['action']);
    $loc = new Location($_POST);
        $loc->set_id($_POST['id']);

    $saved_locations = get_option('wikinearby_saved_locations');
    if($saved_locations === false)
                add_flash_notice( __("Error"), "error", true );
    else{
        $saved_locations->update_location($_POST['id'], $loc);

        update_option('wikinearby_saved_locations', $saved_locations);

        add_flash_notice( __("Location modified succesfully"), "success", true );
    }
    wp_redirect(get_admin_url().'admin.php?page=wikinearby-menu');
}

```

```
20 add_action('admin_post_edit_location', 'wikinearby_edit_location');
```

In questo modo basta inviare una richiesta POST al percorso admin-post.php, specificando, oltre ai parametri della richiesta, un parametro aggiuntivo, `action = edit_location`.
Quando è ricevuta una richiesta del genere, è eseguita la funzione `wikinearby_edit_location`, che non fa altro che recuperare l'oggetto `Saved_Locations` e modificare la `Location` avente l'id passato come parametro, aggiornandola a un luogo con nuovi parametri.
Le altre azioni implementate in modo analogo sono:

- `wikinearby_add_location` : aggiunge una nuova `Location`
- `wikinearby_delete_location` : rimuove una `Location` dato il suo id
- `wikinearby_delete_all_locations`: rimuove tutte le `Location` e reinizializza l'oggetto `Saved_Locations`

In seguito al successo(o all'insuccesso) di ciascuna azione è inviata inoltre una notifica, attraverso il metodo `add_flash_notice`.
Questo si occupa di memorizzare delle notifiche da mostrare successivamente all'utente(attraverso l'azione `admin_notices`[6]); avendo cura di eliminarle una volta che sono state visualizzate.

Visualizzazione del plugin

Per consentire all'utente di inserire il plugin in una sezione del sito è usato il sistema degli shortcode Wordpress[7].
Tramite questa API è semplicemente necessario inserire un codice del tipo `[wikinearby id=1]` per utilizzare il plugin, dove id corrisponde all'id della `Location` nell'oggetto `Saved_Locations`.
Il codice che si occupa di tradurre questi codici nell'output voluto è il seguente:

```
1  function wikinearby_render_shortcode($atts = [], $content = null, $tag = ''){
2      //Normalize
3      $atts = array_change_key_case((array)$atts, CASE_LOWER);
4
5      $wikinearby_atts = shortcode_atts(
6          ['id' => '',] , $atts, $tag);
7
8      if(empty($wikinearby_atts['id']))
9          return 'Wrong shortcode';
10
11     //Find the correct location
12     $given_id = $wikinearby_atts['id'];
13
```

```
14      $saved_locations = get_option('wikinearby_saved_locations');
15      if($saved_locations === false)
16          return "ERROR";
17      else{
18          $found_loc = $saved_locations->get_location_by_id($given_id);
19          if($found_loc === null)
20                  return 'No location found';
21
22          ob_start();
23          $found_loc->display();
24          $output = ob_get_contents();
25          ob_end_clean();
26
27          return $output;
28      }
29  }
30
31  // Then add the shortcode
32  add_shortcode('wikinearby', 'wikinearby_render_shortcode');
```

Esso non fa altro che estrarre l'id voluto dalla notazione shortcode inserita dall'utente, ricercare tra le `Saved_Locations` se esiste un luogo con quell'id e in caso affermativo chiamare il metodo display di quell'elemento, salvando il risultato di questa visualizzazione in una variabile poi ritornata.
Quest'ultimo passaggio è realizzato attivando l'output buffering col metodo PHP `ob_start()`, che invece di mostrare a schermo il risultato di `$found_loc->display()` lo salva in memoria.

Front End

Bibliography

[1] W. Codex, *Options API*, https://codex.wordpress.org/Options_API.

[2] ——, *Hooks*, https://developer.wordpress.org/plugins/hooks/.

[3] ——, *Menu Pages*, https://codex.wordpress.org/Adding_Administration_Menus.

[4] Logicify, *JQuery location picker*, https://github.com/Logicify/jquery-locationpicker-plugin.

[5] W. Codex, *Post Actions API*, https://codex.wordpress.org/Plugin_API/Action_Reference/admin_post_(action).

[6] ——, *Admin Notices*, https://codex.wordpress.org/Plugin_API/Action_Reference/admin_notices.

[7] ——, *Shortcode API*, https://codex.wordpress.org/Shortcode_API.