Desenvolupament plugin de cerca avançada per a wordpress

Índex

Introducció

El wordpress és un software open source desenvolupat originalment per a la creació de blogs.

En els darrers anys la potència i flexibilitat del wordpress, ha esdevingut popular, no només per al desenvolupament de blogs, si no també per al desenvolupament de webs complertes.

El wordpress s'ha convertit en molt més que un software per a la creació de blogs, de fet s'ha convertit en una plataforma estàndard de publicació per a milions de llocs web en tot el mon. És una plataforma robusta, fàcil d'utilitzar i extremadament extensible.

Moltes de les característiques que ens ofereix el wordpress són molt avançades respecte a les que ens ofereixen altres solucions en el mercat actual, però el sistema de cerca que inclou deixa molt que desitjar. Les funcions de cerca que ens ofereix són massa simples per a un problema molt complicat que és trobar el contingut que cerquem en el nostre lloc web.

Tot i que les solucions que ens dona el wordpress, són bones per cercar articles amb paraules exactes, no va més enllà i es queda realment coix davant de les necessitats normals de qualsevol lloc web.

No permet fer cerques complexes amb operands lògics, no cerca en els tags ni en les categories, i els resultats de cerca es mostren ordenats per data sense possibilitat de modificar el criteri. Perdem per tant l'opció de cercar els articles més rellevants en funció de la cerca.

Aquest projecte pretén donar solució a aquestes mancances i crear un plugin per a wordpress que doti al wordpress d'un bon sistema de cerca.

Objectius

Els objectius d'aquest projecte són els de crear i explicar com crear un plugin de cerca per a wordpress que doti al worpress d'un bon sistema de cerca.

Les característiques que haurà de tenir aquest nou plugin seran les següents:

- Possibilitar cerques avançades tipus google (caràcters comodí i operadors lògics)

- Cerca en tot els elements de contingut del WP (articles, pàgines, comentaris, meta-data, contingut d'altres plugins, no només als articles

- Definir diferents criteris de cerca (rellevància...)

- Permetre restringir la cerca a categories i pàgines concretes

- Crear un motor de cerca per millorar la eficàcia i velocitat del cercador

- Dotar al plugin d'un sistema de configuració on l'usuari pugui modificar els diferents paràmetres del motor de cerca, com els pesos dels elements de contingut alhora de calcular la rellevància

Anàlisi

Sistema de cerca del wordpress

Quan s'executa una cerca en el Wordpress tots els resultats es retornen en base a la data de publicació. Això només inclou els articles. Per tant perdem les pàgines, les categories i els tags, que són la resta d'elements de contingut del wp. De tota manera el problema principal, és que el Wordpress ignora la rellevància de les paraules claus a la cerca, i es clar, això fa que el sistema de cerca pràcticament serveixi per ben poc.

Si un article s'ha publicat fa molt de temps, la probabilitat de trobarlo amb el sistema de cerca actual, és pràcticament nul·la. Per altra banda el wordpress te molta facilitat per ser indexat per els cercadors web com google, i al final resulta molt més eficient trobar un article del nostre blog a través de google que no des del cercador intern.

Per exemple si cerquem "Premis Internacionals", el motor de cerca de wp cerca la coincidència exacta, i per tant no trobaria els resultats amb tan sols la paraula "Premis".

El sistema com podem veure és molt arcaic comparat amb la majoria de sistemes actuals.

Probablement en les properes versions s'arreglarà, però ara mateix és un del punts més fluixos d'aquest gestor de continguts.

El Wordpress pot modificar la seva aparença de forma fàcil a traves dels temes. Aquests, són els que gestionen la visualització dels resultats de cerca i la caixa del cercador.

Dins de cada carpeta de tema de wordpress hi ha els fitxers que donen la vista a la part de cerca. Els fitxers que ens importen són el "search.php" ,que conté la plantilla general per als resultats de cerca., i "searchform.php", que conté el codi php que fa la cerca i el formulari de cerca, que bàsicament inclou un <input type='text'> i un botó de submit..

Aquest és el fitxer que normalment s'inclou a la part superior o lateral de les pàgines per dotar-les de cercador. Per a fer-ho el wordpress ens facilita la funció "get_search_form()" que no fa res més que carregar els continguts del fitxer searchform.php.

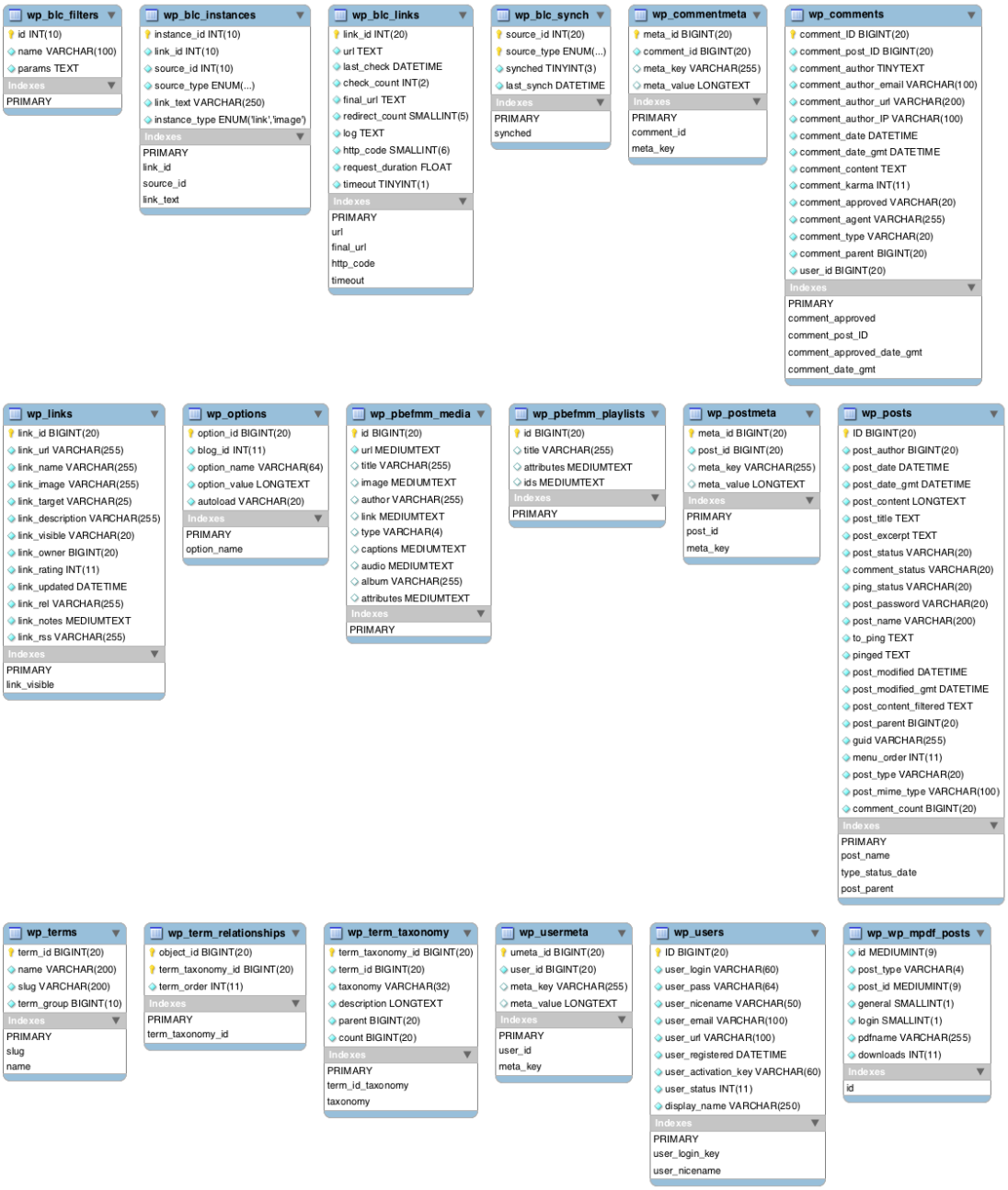
Aquesta fa una query simple i retorna la informació tal i com ja hem dit.

Anàlisi de camps i estructura de bbdd

La estructura de bbdd de wordpress utilitza el gestor de base de dades MySQL.

MySQL suporta indexació i recerca full-text, tot i que actualment el cercador per defecte de wordpress no n'aprofita el potencial.

A continuació veiem l'esquema del model de dades que em extret de l'estructura actual.



Tal i com veiem en l'esquema, la taula en la que es centra el cercador actual, ja que té gairebé tota la informació de base, és la taula wp_posts. Els camps

actuals de cerca són: post_name, post_title, post_content, y post_author. Després d'analitzar-los veiem que a excepció del post_name que si que està marcat com a índex de la taula, la resta no tenen cap optimització per la cerca.

Un dels potencials del mysql respecte a les cerques, és la funcionalitat de fulltext search, que ens facilita el poder realitzar consultes complexes/semàntiques amb càlculs de rellevància, de forma molt àgil. En podem veure les seves característiques a la web de referència de mysql http://dev.mysql.com/doc/refman/5.0/es/fulltext-search.html

Els índexs FULLTEXT poden usar-se només amb taules MyISAM ; poden ser creats des columnes CHAR , VARCHAR , o TEXT

Tal i com podem veure en el esquema del model de dades que hem realitzat, les taules utilitzades ja són del tipus MyISAM;que és el tipus necessari per poder implementar el fulltext search en el mysql, però tot i que existeixen indexs en diferents camps, cap dels camps dins del nostre objectiu de cerca utilitza el tipus d'index FULLTEXT, i per tant ens adonem de la manca de potència del cercador i estructura actual.

La resta de taules i camps, no intervenen en el sistema de cerca actual. Tot i que la taula wp-comments també conté contingut succeptible de ser cercat com ja dèiem en anterioritat.

Anàlisi de plugins de cerca existents

Com ja hem dit, si el worpress destaca per alguna cosa, és per ser extremadament extensible. Existeixen milers de plugins a internet que doten al wordpress de funcionalitats extres. Per tant la nostra primera tasca ha estat revisar quins plugins que millorin el sistema de cerca existeixen i quines són les seves característiques.

S'ha fet un petit "benchmarking" dels plugins que ja existeixen actualment i que intenten donar resposta a la gran mancança actual del wp en quan al seu cercador. Les conclusions han estat, que tot i haver molta oferta, no hi ha cap opció que realment aconsegueixi dotar al wp d'un cercador complert i funcional.

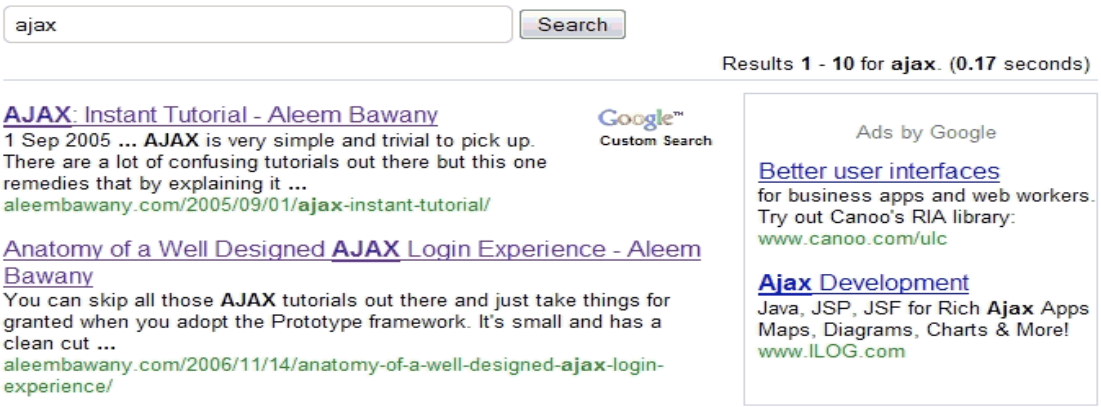
A continuació podem veure els que s'han analitzat i que han ajudat a acabar de definir les característiques del nostre plugin.

WordPress Sphinx Search Plugin

Aquest plugin utilitza el motor de cerca Sphinx per aconseguir resultats de forma ràpida per a sites en wp.

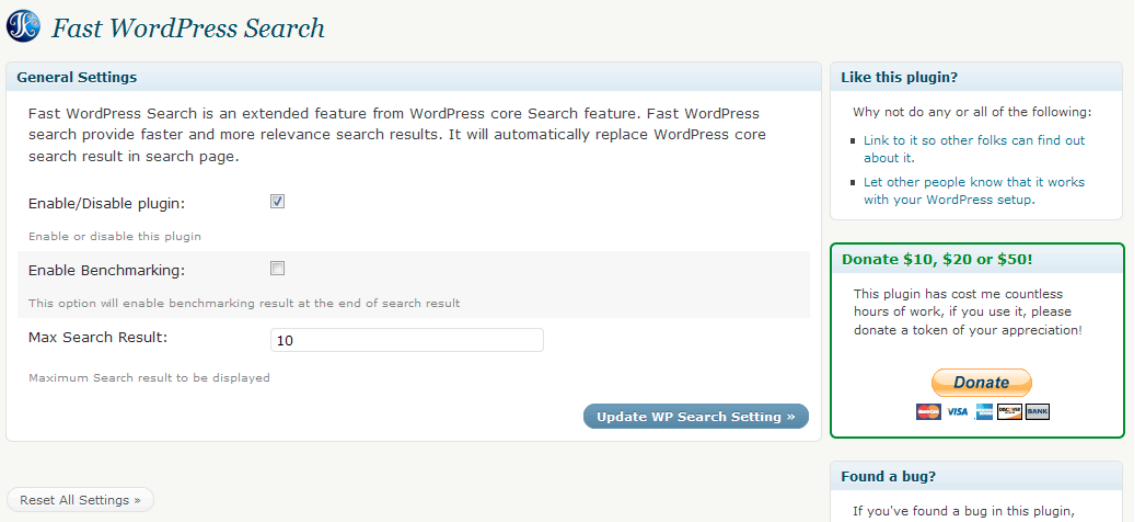
Google Custom Search Plugin

Aquest plugin directament reescriu la cerca per defecte del wordpress per el cercador de google customitzat per a webs.



Fast WordPress Search

Aquest plugin intenta de forma molt simple reemplaçar el motor de cerca del wp intentant optimitzar el temps de cerca, però sense afegir cap funcionalitat especial.

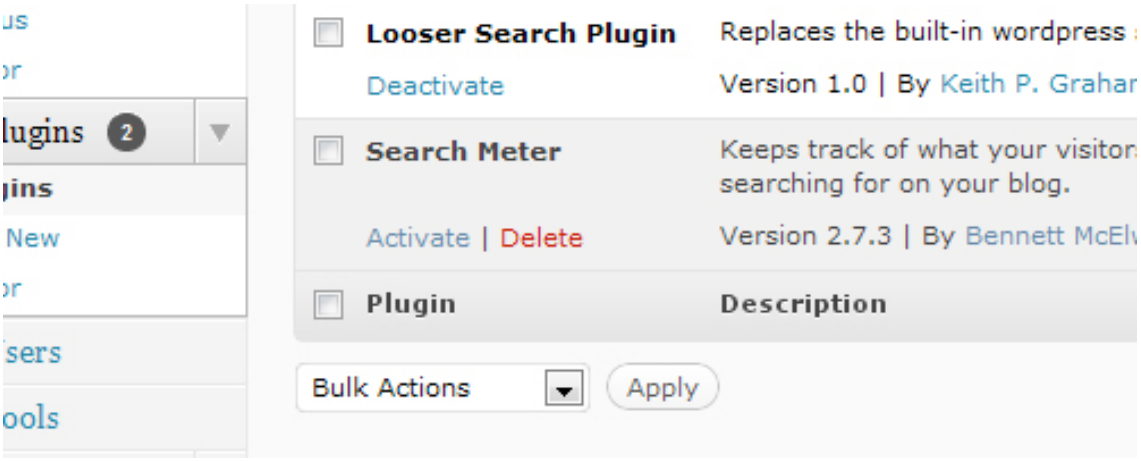


Enhanced Search Form

Per defecte el cercador estàndard de wp només té un únic camp de cerca, aquest plugin ens permet tenir múltiples opcions i realitzar cerques avançades.

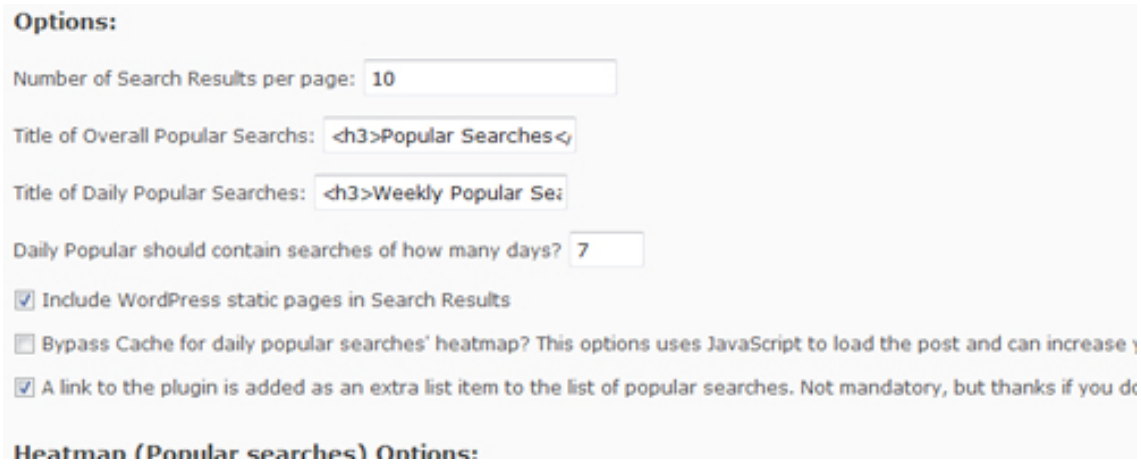
Looser Search Plugin

Aquest plugin crea molta expectativa amb el que diu que fa. Modifica els processos del cercador per defecte del wp per a poder cercar pàgines i articles. Te en compte el nombre d'ocurrències de la cerca, i sembla millorar el funcionament del cercador per defecte, però no acaba de funcionar correctament.



Better Search

Com el seu nom anuncia, aquest plugin pretén millorar la cerca per defecte del wp. La veritat és que ho aconsegueix, ja que te en compte la rellevància de les paraules clau, i pot cercar dins dels tags i les categories. Algunes de les seves característiques s'han tingut en compte per crear el nostre plugin.



Disseny proposat

A partir de l'anàlisi inicial, i d'algunes de les pràctiques que ja utilitzen alguns plugins de cerca, es proposa dissenyar un sistema que sense tocar l'estructura base del worpress, aprofiti el full text search del gestor de base de dades amb que treballa el wordpress MYSQL. Per altra banda es requereix un sistema configurable que permeti càlculs de rellevància en funció dels camps on es trobin els resultats i que alhora permeti excloure categories i pàgines concretes.

Creació de noves taules amb contingut indexat per full text search

La idea, és crear dues taules noves a la base de dades del wp, una per la informació relacionada amb els articles i pàgines, i una altra per la informació relacionada amb els comentaris que continguin tota la informació indexada mitjançant índexs fulltextsearch,per tal de poder ser atacades en el moment de la cerca amb les funcionalitats d'aquest sistema d'indexació.

S'ha decidit que siguin dos taules, perquè la estructura de continguts dels articles/pàgines es força diferent a la dels comentaris i per tant es requereixen camps diferents.

En el moment de la instal·lació aquestes taules estaran buides, i per tant es requerirà crear un procés de "regeneració d'índex" que permeti indexar el contingut actual en les taules de cerca.

Un cop generat el primer índex, aquestes taules, s'aniran actualitzant dinàmicament a mida que els usuaris vagin editant o entrant nou contingut. Per fer-ho es crearan diversos hooks que captin els events necessaris.

Adaptació i indexament de contingut

Com es va detectar analitzant l'estructura de la base de dades de wordpress, el contingut sobre el que volem cercar, esta dispersat per diferents taules i d'aquí la dificultat del wordpress en tenir una cerca efectiva.

En el procés de creació del nostre índex, es recopila tota aquesta informació de les diferents taules d'on volem obtenir el contingut, s'agrupa, s'adapta i s'inserta en les nostres taules de índex.

Aquest procés és repeteix per a cada alta, edició o baixa de contingut, o bé si l'usuari des de les opcions de configuració demana una regeneració de índex, que bàsicament recorre tot el contingut del wp en aquell moment per a tornar a crear l'índex.

Opcions de configuració

Un dels objectius del projecte, era que l'usuari pogués controlar la funcionalitat del cercador en quan a pesos de cada un dels camps de contingut, i alhora poder excloure certes pàgines o categories en el moment de la cerca. En la següent imatge es pot veure com s'han estructurat aquestes opcions.

Opcions de configuració del cercador

Opcions de rellevància

Edita la rellevància dels elements de cerca.

Modificar la rellevància dels elements farà que s'hagi de regenerar el index

Rellevància autor comentari 1

Rellevància contingut comentari 1

Rellevància url autor comentari 1

Rellevància autor article 1

Rellevància categoria article 1

Rellevància contingut article 1

Rellevància resum article 1

Rellevància metadata/camps afegits article 1

Rellevància nom url article 1

Rellevància tags article 1

Rellevància titol article 1

Opcions generals cercador

Si es modica qualsevol opció de les següents s'ha de regenerar l'index

Pàgines a excloure *(Identificadors de articles separats per coma)*

Cateories a excloure 7 *(Identificadors de categories separats per coma)*

Marcar el text trobat als resultats ☐

Guardar

Indexació

Hi ha **361** elements indexats.

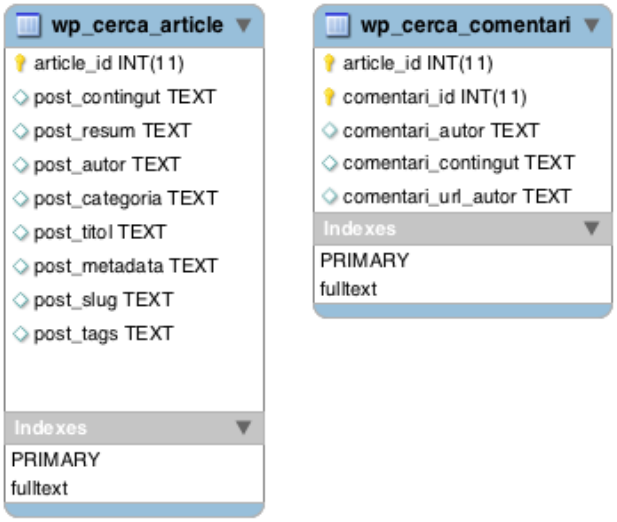
cerques fetes amb aquest index: 14

Regenerar index!

Camps i càlcul de la rellevància

Els índex fulltextsearch de per si ens donen la opció de calcular la rellevància per a una cerca concreta. Com hem vist en les opcions de configuració, el nostre plugin permetrà donar prioritat/rellevància a cada un dels possibles camps de contingut. Per poder fer això, a banda d'utilitzar el full text search i aplicar-li el factor de correcció que ens indiqui l'usuari, caldrà adaptar l'estructura de la base de dades per a que cada un dels nostres índex s'ajusti a aquests camps.

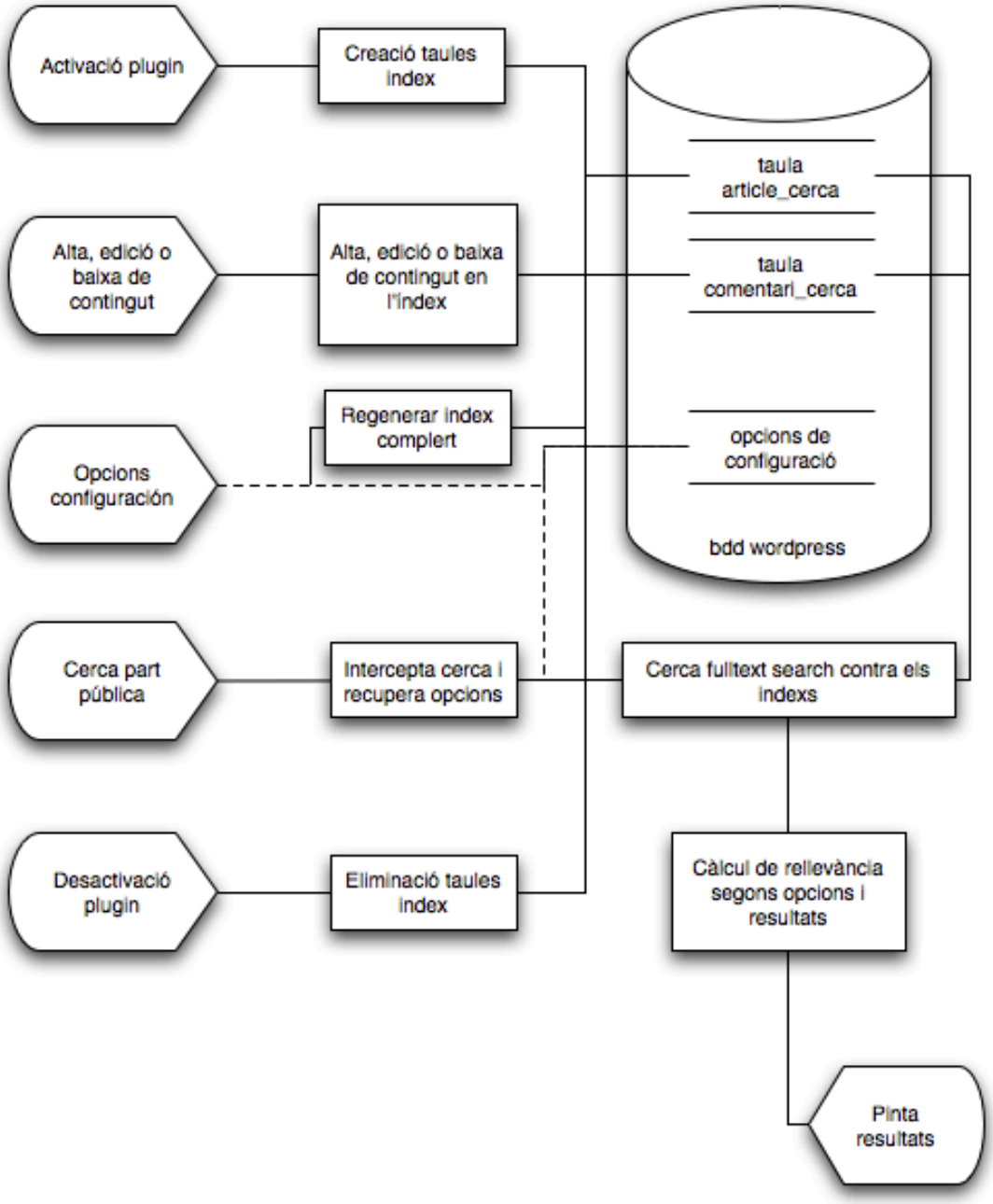
El disseny del nou model de dades necessari per a indexar el contingut i donar resposta a les necessitats és el següent, on tots els camps de text són índexs fulltext.



La cerca

Quan els usuaris de wordpress realitzen una cerca, aquesta s'intercepta de forma totalment transparent a l'usuari, per enlloc d'utilitzar la cerca per defecte del wordpress s'utilitzi la creada per el plugin, s'apliquen els càlculs de rellevància i les opcions d'exclusió de continguts i es retornen els resultats, també de forma transparent, a la pàgina de resultats que es segueix controlant totalment per l'usuari des del tema.

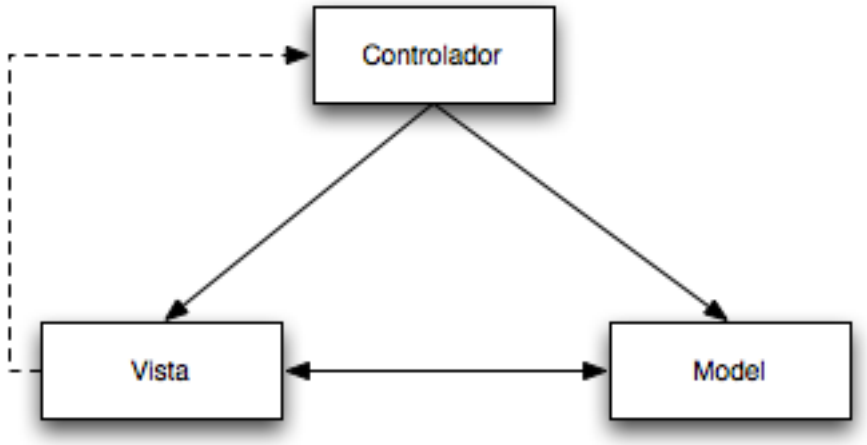
Esquema funcional



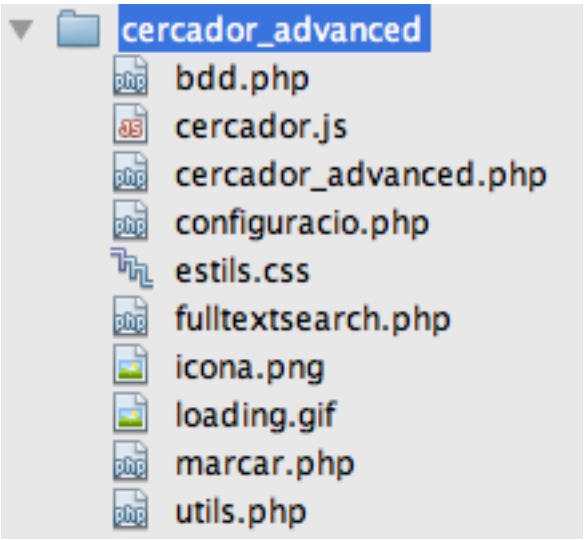
Desenvolupament

Estructura MVC

Tot i la simplicitat d'estructura a nivell de codi del projecte, s'ha intentat seguir el patró de desenvolupament estàndard Model Vista Controlador (MVC), per a possibles futures extensions. De tota manera, cal dir que és un seguiment no estricte.



La estructura de fitxers utilitzada en el projecte és la següent:



On el nostre controlador seria el fitxer "cercador_advanced.php", els models els fitxers "fulltextsearch.php" i "bdd.php", i la vista de la pàgina de configuració el fitxer "configuració.php". La resta de fitxers actuen a nivell de recursos com estils, imatges, funcions utils i scripts de client.

Els codi font

Cercador_advanced.php

```
<?php

/*
 * Plugin Name: Cercador advanced 1.0
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: Plugin que intenta donar funcionalitat de cerca extra al wp aprofitant
les possibilitats del fulltext search de mysql
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)

  This program is free software; you can redistribute it and/or modify
  it under the terms of the GNU General Public License, version 2, as
  published by the Free Software Foundation.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

 */

/**
 * Classe cercador advanced :P
 * */
class CercadorAdvanced {

    //variables que em controlen l'inici i la fi d'un post
    var $loop_comensat = false;
    var $loop_acabat = false;
    var $darrer_post = null;
    var $num_posts = 0;

    /**
     * Constructor
     * inicialitzem tot
     * */
    function CercadorAdvanced() {

        //registrem les variables
        add_action('admin_init', array(&$this, 'registra_opcions'));
        //afegim la pàgina d'opcions a l'administrador
        add_action('admin_menu', array(&$this, 'cercador_afegir_opcions'));
        //accions a executar quan s'activa el plugin
        register_activation_hook(__FILE__, array(&$this, 'activar_plugin'));
        //accions a executar quan es desactiva el plugin
        register_deactivation_hook(__FILE__, array(&$this, 'desactivar_plugin'));
        //afegeix l'enllaç a les ocpions directament a la pàgina de plugins
        add_action('plugin_action_links_' . basename(dirname(__FILE__)) . '/' .
basename(__FILE__), array(&$this, 'opcions_plugin'), 10, 4);

        //afegim el nostre script
        add_action('wp_print_scripts', array(&$this, 'cercador_advanced_scripts'));


        // Accions per a controlar quan es modifica/crea/esborra un post o un comentari
        add_action('save_post', array(&$this, 'guarda_post'), 10, 1);
        add_action('delete_post', array(&$this, 'esborra_post'), 10, 1);

        // afegim el control de cerca , es crida quan es realitza una cerca
        add_action('pre_get_posts', array(&$this, 'modifica_cerca'), 10, 1);


        add_filter('wp_set_comment_status', array(&$this, 'actualitza_comentari'), 10,
2);
        add_filter('edit_comment', array(&$this, 'actualitza_comentari'));
```

```
        //registrem la funcio que es cridara per regenerar el index
        add_action('wp_ajax_reindexa', array(&$this, 'reindexa'), 10);
    }

    /**
     * Funció que registra crea les opcions al wp
     */
    function registra_opcions() {

        //rellevancies
        register_setting('cercador_advanced', 'comentari_autor'); //rellevancia
comentari autor
        register_setting('cercador_advanced', 'comentari_contingut'); //rellevancia
comentari contingut
        register_setting('cercador_advanced', 'comentari_url_autor'); //rellevancia
comentari url autor
        register_setting('cercador_advanced', 'post_autor'); //rellevancia autor post
        register_setting('cercador_advanced', 'post_categoria'); //rellevancia categoria
post
        register_setting('cercador_advanced', 'post_contingut'); //rellevancia contingut
post
        register_setting('cercador_advanced', 'post_resum'); //rellevancia resum post
        register_setting('cercador_advanced', 'post_metadata'); //rellevancia metadata
post
        register_setting('cercador_advanced', 'post_slug'); //rellevancia slug post
        register_setting('cercador_advanced', 'post_tags'); //rellevancia tags post
        register_setting('cercador_advanced', 'post_titol'); //rellevancia titol post
        //exclusions
        register_setting('cercador_advanced', 'excloure_pagines'); //pagines a excloure
separades per ,
        register_setting('cercador_advanced', 'excloure_categories'); //categories a
excloure separades per ,
        //numero de cerques
        register_setting('cercador_advanced', 'num_cerques'); //numero de cerques
realitzades
        //marcar el textos trobats en el contingut
        register_setting('cercador_advanced', 'marcar_text');

        //posem valors per defecte si estan buits
        if (!get_option(comentari_autor))
            update_option('comentari_autor', 1);
        if (!get_option(comentari_contingut))
            update_option('comentari_contingut', 1);
        if (!get_option(comentari_url_autor))
            update_option('comentari_url_autor', 1);

        if (!get_option(post_autor))
            update_option('post_autor', 1);
        if (!get_option(post_categoria))
            update_option('post_categoria', 1);
        if (!get_option(post_contingut))
            update_option('post_contingut', 1);
        if (!get_option(post_resum))
            update_option('post_resum', 1);
        if (!get_option(post_metadata))
            update_option('post_metadata', 1);
        if (!get_option(post_slug))
            update_option('post_slug', 1);
        if (!get_option(post_tags))
            update_option('post_tags', 1);
        if (!get_option(post_titol))
            update_option('post_titol', 1);

        //if(!get_option(excloure_pagines)) update_option('excloure_pagines', 1);
        //if(!get_option(excloure_categories)) update_option('excloure_categories', 1);

        if (!get_option(num_cerques))
            update_option('num_cerques', 0);

        //if(!get_option(marcar_text)) update_option('marcar_text', 1);
    }

    /**
     * Creem l'opció en el menu administrador
     * */
    function cercador_afegir_opcions() {
        add_menu_page('Cercador', 'Cercador', 'administrator', basename(__FILE__),
array(&$this, 'pagina_principal_admin'));
```

```
    }

    /**
     * Funció que es crida quan s'activa el plugin
     * crea un motor de cerca buit
     * */
    function activar_plugin() {

        include_once dirname(__FILE__) . '/bdd.php';
        include_once dirname(__FILE__) . '/fulltextsearch.php';

        $bdd = new bdd();
        $bdd->init();

        $motor = new FullTextSearch();
        $motor->genera_motor();
    }

    /**
     * Funció que es crida quan es desactiva el plugin
     * elimina qualsevol rastre de motor de cerca
     * */
    function desactivar_plugin() {

        include_once dirname(__FILE__) . '/bdd.php';
        include_once dirname(__FILE__) . '/fulltextsearch.php';

        $bdd = new bdd();
        $bdd->esborrar();

        $motor = new FullTextSearch();
        $motor->esborra_motor();
    }

    /**
     * Mostra les opcions del plugin
     * */
    function opcions_plugin($links) {
        $enllas = '<a href="admin.php?page=' . basename(__FILE__) . '">' . __('Opcions')
. '</a>';
        array_unshift($links, $enllas);
        return $links;
    }

    /**
     * Afegeix el javascript que necessitem
     * */
    function cercador_advanced_scripts() {
        wp_enqueue_script('cercador_advanced', path_join(WP_PLUGIN_URL,
basename(dirname(__FILE__))) . '/cercador.js', array('jquery-form'));
    }

    /**
     * Funció que cridem quan es guarda un post
     * Actualitza l'index amb la informació d'aquest post
     * */
    function guarda_post($id) {
        global $wpdb;

        // Recullo el post complert
        $post = $wpdb->get_row("SELECT {$wpdb->posts}.*,{$wpdb-
>users}.user_login,{$wpdb->users}.user_nicename,{$wpdb->users}.display_name FROM {$wpdb-
>posts} LEFT JOIN {$wpdb->users} ON {$wpdb->posts}.post_author={$wpdb->users}.ID WHERE
{$wpdb->posts}.ID=" . $id);

        // Si no es una revisió
        if ($post->post_type != 'revision') {

            include_once dirname(__FILE__) . '/fulltextsearch.php';
            include_once dirname(__FILE__) . '/utils.php';

            $motor = new FullTextSearch();
            $motor->borra($post->ID);

            if (!$this->exclos($post)) {
                //no esta exclos l'hem d'indexar
                //preparem l'autor
                $camp = $post->display_name;
```

```
                $camps['post_autor'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_autor'));
                //preparem les categories
                $camp = get_the_category_list(',', '', $post->ID);
                $camps['post_categoria'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_categoria'));
                //preparem el contingut del post
                $content = apply_filters('the_content', $post->post_content);
                $content = apply_filters('the_real_content', $content, $post);
                $camp = str_replace(']]>', ']]&gt;', $content);
                $camps['post_contingut'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_contingut'));
                //preparem el resum
                $camp = apply_filters('the_excerpt', $post->post_excerpt);
                $camps['post_resum'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_resum'));

                //agafem els camps meta i els posem separats per espai
                $idem = get_post_custom($post->ID);

                if (count($idem) > 0) {
                    $camp = '';
                    foreach ($idem as $camp_meta => $valor_meta) {
                        $camp .= ' ' . implode(' ', $valor_meta);
                    }
                }

                $camps['post_metadata'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_metadata'));
                //agafem el nom de la pagina
                $camp = str_replace(get_bloginfo('home'), '', get_permalink($post->ID));
                $camps['post_slug'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_slug'));
                //agafem els tags
                $noms_tags = array();
                $tags = get_the_tags($post->ID);
                if ($tags) {
                    foreach ((array) $tags AS $tag) {
                        $noms_tags[] = $tag->name;
                    }
                }
                $camp = implode(' ', $noms_tags);
                $camps['post_tags'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_tags'));
                //preparao el titol
                $camp = apply_filters('the_title', $post->post_title, '', '');
                $camps['post_titol'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_titol'));

                $motor->indexa($post->ID, $camps, $post);
            }
        }
    }

    /**
     * Funció que es crida quan s'esborra un post
     * Actualitza l'index del post
     * */
    function esborra_post($id) {
        include_once dirname(__FILE__) . '/fulltextsearch.php';

        $motor = new FullTextSearch();
        $motor->borra($id);
    }

    /*
     * Funció que valida si hem d'excloure un post
     */

    function exclos($post) {

        $categories_excloses = array_filter(explode(',',
get_option('excloure_categories')));
        $articles_exclosos = array_filter(explode(',', get_option('excloure_pagines')));
        $categories = get_the_category($post->ID);

        if (!in_array($post->ID, $articles_exclosos)) {
            //no es un article exclos
            foreach ((array) $categories as $categoria) {
```

```
                if (in_array($categoria->cat_ID, $categories_excloses))
                    return true; // es una categoria exclosa

            }

            // no permetem els privats
            if ($post->post_status == 'private')
                return true;
            // no permetem els temporals
            if ($post->post_status == 'draft')
                return true;
            //no permetem els que estan protegits per password
            if ($post->post_password != '')
                return true;

            return false;
        }

        return true;
    }

    /**
     * Funcio que es crida quan es fa una cerca
     * */
    function modifica_cerca() {
        if (is_search ()) {
            include_once dirname(__FILE__) . '/fulltextsearch.php';
            $this->motor = new FullTextSearch();
            //control de inici i fi del loop
            add_action('loop_start', array(&$this, 'comensar_loop'));
            add_action('loop_end', array(&$this, 'acabar_loop'));
            //actualitzo el número de cerques
            $num_cerca = get_option('num_cerques');
            $num_cerca++;
            update_option('num_cerques', $num_cerca);

            //marquem el text si la variable esta activa
            if (get_option('marcar_text')) {
                include_once dirname(__FILE__) . '/marcar.php';
                //marco el contingut i el resum
                add_filter('the_content', array(&$this, 'marca_titol'), 15, 1);
                add_filter('the_excerpt', array(&$this, 'marca_titol'), 15, 1);
                //marcar titol
                add_filter('the_title', array(&$this, 'marca_titol'));

                //forcem que surti text en els resultats
                add_action('the_time', array(&$this, 'forsa_contingut'));
            }
        }
    }

    /**
     * Funcio que reindexa un comentari
     * */
    function actualitza_comentari($id) {
        $comentari = get_comment($id);
        include_once dirname(__FILE__) . '/fulltextsearch.php';
        include_once dirname(__FILE__) . '/utils.php';
        $motor = new FullTextSearch();
        $motor->borra($id);

        if (!empty($comentari) && !$this->exclos(get_post($comentari->comment_post_ID)))
{
            //no esta buit ni s'ha d'excloure preparo camps i indexo
            $camp = apply_filters('comment_author', apply_filters('get_comment_author',
$post->comment_author));
            $campsc['comentari_autor'] = array('info' => neteja($camp), 'prioritat' =>
get_option('comentari_autor'));

            $camp = apply_filters('comment_text', apply_filters('get_comment_text',
$post->comment_content));
            $campsc['comentari_contingut'] = array('info' => neteja($camp), 'prioritat'
=> get_option('comentari_contingut'));

            $camp = apply_filters('comment_url', $post->comment_author_url);
            $campsc['comentari_url_autor'] = array('info' => neteja($camp), 'prioritat'
=> get_option('comentari_url_autor'));

            $motor->indexa($comentari->comment_post_ID, $campsc, $comentari);
```

```
        }
    }

    /**
     * Forçar el afegir contingut al resultat de cerca
     * */
    function forsa_contingut($contingut) {
        if (in_the_loop ()) {
            global $post;
            return $contingut . '</small>' . $this->marca_titol($post->post_content) .
'<small>';
        }

        return $contingut;
    }

    /**
     * Funció que es crida quan comença un loop d'un post
     * */
    function comensar_loop() {
        $this->loop_comensat = true;
    }

    /**
     * Funció que es crida quan comença un loop d'un post
     * */
    function acabar_loop() {
        $this->loop_acabat = true;
    }

    /**
     * Funció que marca el titol del post
     */
    function marca_titol($text) {

        if (in_the_loop ()) {
            global $post;

            if ($this->darrer_post != $post->ID) {
                $this->darrer_post = $post->ID;
                $this->num_posts = 0;
            }

            $this->num_posts++;

            $marca = new Marcar($text, $this->motor->recull_paraules());
            $marca->marcar_paraules();

            return $marca->recupera_text();
        }

        return $text;
    }

    /**
     * Main administration page
     *
     * @return void
     * */
    function pagina_principal_admin() {
        //pintem la configuració
        include_once dirname(__FILE__) . '/fulltextsearch.php';

        $motor = new FullTextSearch();


        include dirname(__FILE__) . '/configuracio.php';
    }

    /**
     * Funcio que es crida des de les opcions i permet regenerar l'index
     * */
    function reindexa() {
        if (check_ajax_referer('cercadoradvanced-index')) {
            require dirname(__FILE__) . '/fulltextsearch.php';


            ob_start();
```

```
            $motor = new FullTextSearch();

            $offset = intval($_POST['offset']);

            if ($offset == 0)
                $motor->reset();


            list( $remaining, $total ) = $motor->reindexa($offset,
intval($_POST['limit']));

            ob_end_clean();

            $percent = 100;
            if ($total > 0)
                $percent = ( ( $total - $remaining ) / $total ) * 100;

            if ($remaining > 0)
                echo sprintf('%d %d%% ', $remaining, $percent) . sprintf(__('%d of %d /
%d%%'), $total - $remaining, $total, $percent);
            else {
                echo '0 100% ' . __('Fet!');
            }

            die();
        }
    }

}

/**
 * Començar!
 * */
global $cercador;
$cercador = new CercadorAdvanced();
```

fulltextsearch.php

```
<?php

/**
 * Motor de cerca basat en fulltext search de mysql
 * */
class FullTextSearch {

    var $paraules = array();
    var $camps = null;
    var $limits = null;
    var $ordre = null;
    var $phrases = array();
    var $and = array();
    var $petit = array();
    var $full = array();

    /**
     * Constructor.
     * */
    function FullTextSearch() {
        //grabo els camps quan arriben
        add_filter('posts_fields', array(&$this, 'camps_post'));
        //quan em modifiquen els limits
        add_filter('post_limits', array(&$this, 'assigna_limits'));

        //quan em fan una cerca
        add_filter('posts_request', array(&$this, 'cerca'));
        //quan em fan un canvi d'ordre
        add_filter('posts_orderby', array(&$this, 'canvi_ordre'));
    }

    function extract_phrases($search) {
        $search = preg_replace_callback("/['\"](.*?)['\"]/", array(&$this,
'exact_words'), $search);
        $phrases = explode(' ', $search);

        return array_filter(array_merge($this->phrases, $phrases));
    }
```

```
    /**
     * Funcio que em converteix els and a format fulltext search
     * */
    function la_and($literals) {
        $this->full[] = '+' . $literals[1] . ' +' . $literals[2];
        return '';
    }

    /**
     * Guarda les cerques que estan entrecometes "literals"
     * */
    function entre_cometes($literals) {
        $this->petit[] = $literals[1];
        return '';
    }

    /**
     * aplico el canvi d'ordre quan em ve de wp
     * */
    function canvi_ordre($order) {
        $this->ordre = $order;
        return $order;
    }

    /**
     * Guarda els camps a utilitzar
     * */
    function camps_post($fields) {
        $this->camps = $fields;
        return $fields;
    }

    /**
     * Limits de sql
     * */
    function assigna_limits($limits) {
        $this->limits = $limits;
        return $limits;
    }

    /**
     * Funcio que es crida quan hi ha una cerca
     */
    function cerca($request) {
        global $wpdb;
        global $current_user;
        //recollim la cerca
        $_GET['s'] = get_query_var('s');

        // Agafo els parametres
        $parametre = trim($_GET['s']);
        //tinc un literal entre cometes
        $parametre = preg_replace_callback("/['\"](.*?)['\"]/", array(&$this,
'entre_cometes'), $parametre);
        //tinc una and
        $parametre = preg_replace_callback('/(\w*)\s*AND\s*(\w*)/', array(&$this,
'la_and'), $parametre);
        //tinc una or
        $parametre = preg_replace('/(\w*)\s*OR\s*(\w*)/', '$1 $2', $parametre);

        // separa les paraules petites i grans
        $paraules = array_filter(preg_split('/[\s,]+/', trim($parametre)));

        foreach ($paraules as $paraula) {
            //si la paraula es mes petita que 4 la passo per la cerca normal ja que el
fulltextsearch no li faria cas
            if (strlen($paraula) >= 4)
                $this->full[] = $paraula;
            else
                $this->petit[] = $paraula;
        }



        $and = array();
        $have_comments = false;
        //comentari
        if (isset($_GET['comentari_autor'])) {
```

```
            $and[].= "{$wpdb->prefix}cerca_comentari.comentari_autor LIKE '%" . $wpdb-
>escape($_GET['comentari_autor']) . "%'";
            $have_comments = true;
        }
        if (isset($_GET['comentari_contingut'])) {
            $and[].= "{$wpdb->prefix}cerca_comentari.comentari_contingut LIKE '%" .
$wpdb->escape($_GET['comentari_contingut']) . "%'";
            $have_comments = true;
        }
        if (isset($_GET['comentari_url_autor'])) {
            $and[].= "{$wpdb->prefix}cerca_comentari.comentari_url_autor LIKE '%" .
$wpdb->escape($_GET['comentari_url_autor']) . "%'";
            $have_comments = true;
        }
        //post
        if (isset($_GET['post_autor']))
            $and[].= "{$wpdb->prefix}cerca_article.post_autor LIKE '%" . $wpdb-
>escape($_GET['post_autor']) . "%'";
        if (isset($_GET['post_categoria']))
            $and[].= "{$wpdb->prefix}cerca_article.post_categoria LIKE '%" . $wpdb-
>escape($_GET['post_categoria']) . "%'";
        if (isset($_GET['post_contingut']))
            $and[].= "{$wpdb->prefix}cerca_article.post_contingut LIKE '%" . $wpdb-
>escape($_GET['post_contingut']) . "%'";
        if (isset($_GET['post_resum']))
            $and[].= "{$wpdb->prefix}cerca_article.post_resum LIKE '%" . $wpdb-
>escape($_GET['post_resum']) . "%'";
        if (isset($_GET['post_metadata']))
            $and[].= "{$wpdb->prefix}cerca_article.post_metadata LIKE '%" . $wpdb-
>escape($_GET['post_metadata']) . "%'";
        if (isset($_GET['post_slug']))
            $and[].= "{$wpdb->prefix}cerca_article.post_slug LIKE '%" . $wpdb-
>escape($_GET['post_slug']) . "%'";
        if (isset($_GET['post_tags']))
            $and[].= "{$wpdb->prefix}cerca_article.post_tags LIKE '%" . $wpdb-
>escape($_GET['post_tags']) . "%'";
        if (isset($_GET['post_titol']))
            $and[].= "{$wpdb->prefix}cerca_article.post_titol LIKE '%" . $wpdb-
>escape($_GET['post_titol']) . "%'";



        if ($have_comments)
            $this->camps .= ",{$wpdb->prefix}cerca_comentari.comentari_id";

        // anem per els fulltext
        $rellevancies = array();
        $camps = array();
        $altres = array();

        //si tinc elements al array de fulltext
        if (count($this->full) > 0) {
            $paraula = implode(' ', $this->full);
            $paraula = trim($paraula);
            $this->paraules = $this->full;

            ;
            //comentaris
            $camp = $wpdb->prefix . 'cerca_comentari.comentari_autor';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('comentari_autor'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_comentari.comentari_contingut';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('comentari_contingut'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_comentari.comentari_url_autor';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('comentari_url_autor'), $camp, $wpdb->escape($paraula));

            //post
            $camp = $wpdb->prefix . 'cerca_article.post_autor';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_autor'), $camp, $wpdb->escape($paraula));
```

```
            $camp = $wpdb->prefix . 'cerca_article.post_categoria';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_categoria'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_contingut';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_contingut'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_resum';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_resum'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_metadata';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_metadata'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_slug';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_slug'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_tags';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_tags'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_titol';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_titol'), $camp, $wpdb->escape($paraula));

            $camp = $wpdb->prefix . 'cerca_article.post_autor';
            $camps[] = $camp;
            $rellevancies[] = sprintf("(%2.2f * (MATCH(%s) AGAINST ('%s' IN BOOLEAN
MODE)))", get_option('post_autor'), $camp, $wpdb->escape($paraula));



            $this->camps .= ',MAX(' . implode(' + ', $rellevancies) . ') AS
rellevancia';
            $this->ordre = 'rellevancia DESC,' . $this->ordre;
        }

        // la mega consulta de cerca
        $sql = "SELECT DISTINCT SQL_CALC_FOUND_ROWS " . $this->camps . " FROM {$wpdb-
>posts} LEFT JOIN {$wpdb->prefix}cerca_article ON {$wpdb->posts}.ID={$wpdb-
>prefix}cerca_article.article_id ";

        $sql .= " LEFT JOIN {$wpdb->prefix}cerca_comentari ON {$wpdb->posts}.ID={$wpdb-
>prefix}cerca_comentari.article_id ";

        $sql .= ' WHERE 1=1 ';


        if (count($and) > 0)
            $sql .= ' AND ' . implode(' AND ', $and);

        // paraules massa curtes per el fullltextsearch
        foreach ((array) $this->petit as $petit) {
            $this->paraules[] = $petit;

            $altres[] = "{$wpdb->prefix}cerca_comentari.comentari_autor LIKE '%" .
$wpdb->escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_comentari.comentari_contingut LIKE '%" .
$wpdb->escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_comentari.comentari_url_autor LIKE '%" .
$wpdb->escape($petit) . "%'";

            //post
            $altres[] = "{$wpdb->prefix}cerca_article.post_autor LIKE '%" . $wpdb-
>escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_article.post_categoria LIKE '%" . $wpdb-
>escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_article.post_contingut LIKE '%" . $wpdb-
>escape($petit) . "%'";
```

```
            $altres[] = "{$wpdb->prefix}cerca_article.post_resum LIKE '%" . $wpdb-
>escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_article.post_metadata LIKE '%" . $wpdb-
>escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_article.post_slug LIKE '%" . $wpdb-
>escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_article.post_tags LIKE '%" . $wpdb-
>escape($petit) . "%'";
            $altres[] = "{$wpdb->prefix}cerca_article.post_titol LIKE '%" . $wpdb-
>escape($petit) . "%'";
        }

        if (count($this->full) > 0)
            $altres[] = sprintf("(MATCH(" . implode(',', $camps) . ") AGAINST ('%s' IN
BOOLEAN MODE))", $wpdb->escape($paraula));

        $sql .= ' AND (' . implode(' OR ', $altres) . ') ';

        if (count($this->full) > 0)
            $sql .= " GROUP BY {$wpdb->posts}.ID HAVING rellevancia ";

        $sql .= "ORDER BY " . $this->ordre . ' ';
        $sql .= $this->limits;

        error_log('------------------->' . $sql . '<-------------------');

        return $sql;
    }

    /**
     * Retorna les paraules com un array de expressions regulars
     * */
    function recull_paraules() {
        $paraules = array();

        // Convert words into proper patterns
        foreach ((array) $this->paraules as $paraula) {
            if (substr($paraula, 0, 1) != '-') {
                $word = preg_quote($paraula, '/');
                $word = str_replace('\\*', '\w*', $paraula);

                if (trim($paraula) != '')
                    $paraules[] = '\b' . $word . '\b';
            }
        }

        return $paraules;
    }

    /**
     * Afegeix la informació d'un post al index
     * */
    function indexa($id_article, $post_preparat, $post_original) {
        global $wpdb;

        $camps = array();
        $valors = array();
        foreach ($post_preparat as $camp => $valor) {
            $camps[] = $camp;
            $valors[] = "'" . $wpdb->escape($valor['info']) . "'";
        }

        if (isset($post_original->comment_post_ID)) {
            //hi ha comentaris
            $wpdb->query("INSERT INTO {$wpdb->prefix}cerca_comentari
(article_id,comentari_id," . implode(',', $camps) . ") VALUES('$id_article','" .
$post_original->comment_ID . "'," . implode(',', $valors) . ")");
        } else {
            $wpdb->query("INSERT INTO {$wpdb->prefix}cerca_article (article_id," .
implode(',', $camps) . ") VALUES('$id_article'," . implode(',', $valors) . ")");
        }
    }

    /**
     * Funció que inicialitza el index
     * */
    function reset() {
        global $wpdb;
```

```
        $wpdb->query("TRUNCATE {$wpdb->prefix}cerca_article");
        $wpdb->query("TRUNCATE {$wpdb->prefix}cerca_comentari");
    }

    /**
     * Funció que elimina un element del index
     * */
    function borra($id) {
        global $wpdb;

        $wpdb->query("DELETE FROM {$wpdb->prefix}cerca_article WHERE article_id=$id");
        $wpdb->query("DELETE FROM {$wpdb->prefix}cerca_comentari WHERE article_id=$id");
    }

    /**
     * Funció que retorna el numero de elements que tenim indexats
     * */
    function numero_indexats() {
        global $wpdb;

        $numero_articles_indexats = $wpdb->get_var("SELECT COUNT(article_id) FROM
{$wpdb->prefix}cerca_article");
        $numero_comentaris_indexats = $wpdb->get_var("SELECT COUNT(comentari_id) FROM
{$wpdb->prefix}cerca_comentari");

        return $numero_articles_indexats + $numero_comentaris_indexats;
    }

    /**
     * Funció que crea les taules que mantindran el contingut indexat
     * */
    function genera_motor() {
        global $wpdb;

        $this->esborra_motor();


        $article = "CREATE TABLE `{$wpdb->prefix}cerca_article` (
                            `article_id` int(11) unsigned NOT NULL,
                            `post_contingut` text,
                            `post_resum` text,
                            `post_autor` text,
                            `post_categoria` text,
                            `post_titol` text,
                            `post_metadata` text,
                            `post_slug` text,
                            `post_tags` text,
                            PRIMARY KEY (`article_id`),
                            FULLTEXT KEY `fulltext`
(`post_contingut`,`post_resum`,`post_autor`,`post_categoria`,`post_titol`,`post_metadata
`,`post_slug`,`post_tags`)
                          ) ENGINE=MyISAM DEFAULT CHARSET=utf8;";



        $comentari = "CREATE TABLE `{$wpdb->prefix}cerca_comentari` (
                            `article_id` int(11) unsigned NOT NULL,
                            `comentari_id` int(11) unsigned NOT NULL,
                            `comentari_autor` text,
                            `comentari_contingut` text,
                            `comentari_url_autor` text,
                            PRIMARY KEY (`comentari_id`,`article_id`),
                            FULLTEXT KEY `fulltext`
(`comentari_autor`,`comentari_contingut`,`comentari_url_autor`)
                          ) ENGINE=MyISAM DEFAULT CHARSET=utf8;";


        $wpdb->query($article);
        $wpdb->query($comentari);
    }

    /**
     * Funcio que agafa la informació actual del wp i la posa en el nostre index
     * */
    function reindexa($offset, $count) {

        require dirname(__FILE__) . '/utils.php';

        global $wpdb;
```

```
        $total = 0;
        $total_articles = 0;
        $total_comentaris = 0;
        $article_sql = '';
        $comentari_sql = '';

        $categories_excloses = array_filter(explode(',',
get_option('excloure_categories')));
        $pagines_excloses = array_filter(explode(',', get_option('excloure_pagines')));
        $pagines_excloses[] = 0;

        $tipus = array();
        $sql = array();

        $tipus[] = "{$wpdb->posts}.post_type='page'";
        $tipus[] = "{$wpdb->posts}.post_type='post'";

        if (count($tipus) > 0)
            $sql[] = implode(' OR ', $tipus);

        $article_sql = ' WHERE ';
        $comentari_sql = ' WHERE ';
        if (count($categories_excloses) > 0) {
            $article_sql = " LEFT JOIN {$wpdb->term_relationships} AS rel ON ({$wpdb-
>posts}.ID=rel.object_id) LEFT JOIN {$wpdb->term_taxonomy} AS tax ON
(tax.taxonomy='category' AND rel.term_taxonomy_id=tax.term_taxonomy_id) LEFT JOIN
{$wpdb->terms} AS term ON (tax.term_id=term.term_id) WHERE ";
            $article_sql .= 'tax.term_id NOT IN (' . implode(',', $categories_excloses)
. ") AND ";

            $comentari_sql = str_replace("{$wpdb->posts}.ID", "{$wpdb-
>comments}.comment_post_ID", $article_sql);
        }

        $article_sql .= ' (' . implode(') AND (', $sql) . ") AND {$wpdb->posts}.ID NOT
IN (" . implode(', ', $pagines_excloses) . ") AND {$wpdb->posts}.post_type IN ( 'post',
'page' )";
        $comentari_sql .= "{$wpdb->comments}.comment_type='' AND {$wpdb-
>comments}.comment_approved='1' AND {$wpdb->comments}.comment_post_ID NOT IN(" .
implode(', ', $pagines_excloses) . ")";


        $article_sql .= " AND {$wpdb->posts}.post_status!='private'";
        $article_sql .= " AND {$wpdb->posts}.post_status!='draft'";
        $article_sql .= " AND {$wpdb->posts}.post_password=''";




        $total_articles = $wpdb->get_var("SELECT COUNT(DISTINCT {$wpdb->posts}.ID) FROM
{$wpdb->posts} " . $article_sql);
        $total_comentaris = $wpdb->get_var("SELECT COUNT(DISTINCT {$wpdb-
>comments}.comment_ID) FROM {$wpdb->comments} " . $comentari_sql);

        $total = $total_articles + $total_comentaris;


        if ($total_articles > 0 && $offset < $total_articles) {

            $relative = $offset;
            $rows = $wpdb->get_results("SELECT DISTINCT {$wpdb->posts}.*,{$wpdb-
>users}.user_login,{$wpdb->users}.user_nicename,{$wpdb->users}.display_name FROM {$wpdb-
>posts} LEFT JOIN {$wpdb->users} ON {$wpdb->posts}.post_author={$wpdb->users}.ID " .
$article_sql . " ORDER BY {$wpdb->posts}.ID LIMIT $relative,$count");

            foreach ((array) $rows AS $post) {

                //preparem l'autor
                $camp = $post->display_name;
                $camps['post_autor'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_autor'));
                //preparem les categories
                $camp = get_the_category_list(',', '', $post->ID);
                $camps['post_categoria'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_categoria'));
                //preparem el contingut del post
                $content = apply_filters('the_content', $post->post_content);
                $content = apply_filters('the_real_content', $content, $post);
```

```
                $camp = str_replace(']]>', ']]&gt;', $content);
                $camps['post_contingut'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_contingut'));
                //preparem el resum
                $camp = apply_filters('the_excerpt', $post->post_excerpt);
                $camps['post_resum'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_resum'));

                //agafem els camps meta i els posem separats per espai
                $idem = get_post_custom($post->ID);

                if (count($idem) > 0) {
                    $camp = '';
                    foreach ($idem as $camp_meta => $valor_meta) {
                        $camp .= ' ' . implode(' ', $valor_meta);
                    }
                }

                $camps['post_metadata'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_metadata'));
                //agafem el nom de la pagina
                $camp = str_replace(get_bloginfo('home'), '', get_permalink($post->ID));
                $camps['post_slug'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_slug'));
                //agafem els tags
                $noms_tags = array();
                $tags = get_the_tags($post->ID);
                if ($tags) {
                    foreach ((array) $tags AS $tag) {
                        $noms_tags[] = $tag->name;
                    }
                }
                $camp = implode(' ', $noms_tags);
                $camps['post_tags'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_tags'));
                //preparao el titol
                $camp = apply_filters('the_title', $post->post_title, '', '');
                $camps['post_titol'] = array('info' => neteja($camp), 'prioritat' =>
get_option('post_titol'));

                $this->indexa($post->ID, $camps, $post);
            }

            return array($total - count($rows) - $offset, $total);
        } elseif ($total_comentaris > 0 && $offset < $total) {
            $relative = $offset - $total_articles;

            $rows = $wpdb->get_results("SELECT DISTINCT
comment_ID,comment_post_ID,comment_author,comment_author_url,comment_content FROM
{$wpdb->comments} " . $comentari_sql . " ORDER BY comment_ID LIMIT $relative,$count");

            foreach ((array) $rows AS $post) {

                $camp = apply_filters('comment_author',
apply_filters('get_comment_author', $post->comment_author));
                $campsc['comentari_autor'] = array('info' => neteja($camp), 'prioritat'
=> get_option('comentari_autor'));

                $camp = apply_filters('comment_text', apply_filters('get_comment_text',
$post->comment_content));
                $campsc['comentari_contingut'] = array('info' => neteja($camp),
'prioritat' => get_option('comentari_contingut'));

                $camp = apply_filters('comment_url', $post->comment_author_url);
                $campsc['comentari_url_autor'] = array('info' => neteja($camp),
'prioritat' => get_option('comentari_url_autor'));

                $this->indexa($post->comment_post_ID, $campsc, $post);
            }

            return array($total - count($rows) - $offset, $total);
        }


        return array(0, $total);
    }

    /**
     * Funció que elimina les taules de indexament
```

```
     * */
    function esborra_motor() {
        global $wpdb;

        $wpdb->query("DROP TABLE IF EXISTS {$wpdb->prefix}cerca_article");
        $wpdb->query("DROP TABLE IF EXISTS {$wpdb->prefix}cerca_comentari");
    }

}
```

configuracio.php

```
<?php if (!defined('ABSPATH'))
    die('No es permet accés directe'); ?>
<link rel="stylesheet" href="<?php echo path_join(WP_PLUGIN_URL,
basename(dirname(__FILE__))) . '/estils.css'; ?>" type="text/css" media="screen"
title="no title" charset="utf-8"/>

<div class="wrap">
<?php screen_icon(); ?>
    <h2><?php _e('Opcions de configuració del cercador'); ?></h2>

    <form action="options.php" method="post" accept-charset="utf-8"
id="cercador_advanced_opcions">
<?php settings_fields('cercador_advanced'); ?>
        <ul>
            <li>
                <h3><?php _e('Opcions de rellevància'); ?></h3>

                <p style="clear: both"><?php _e('Edita la rellevància dels elements de
cerca.'); ?></p>
                <p><?php _e('Modificar la rellevància dels elements farà que s\'hagi de
regenerar el index'); ?></p>
                <ul>
                    <li>
                        <label><?php _e('Rellevància autor comentari') ?></label>
                        <input type="text" name="comentari_autor" value="<?php echo
get_option("comentari_autor"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància contingut comentari') ?></label>
                        <input type="text" name="comentari_contingut" value="<?php echo
get_option("comentari_contingut"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància url autor comentari') ?></label>
                        <input type="text" name="comentari_url_autor" value="<?php echo
get_option("comentari_url_autor"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància autor article') ?></label>
                        <input type="text" name="post_autor" value="<?php echo
get_option("post_autor"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància categoria article') ?></label>
                        <input type="text" name="post_categoria" value="<?php echo
get_option("post_categoria"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància contingut article') ?></label>
                        <input type="text" name="post_contingut" value="<?php echo
get_option("post_contingut"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància resum article') ?></label>
                        <input type="text" name="post_resum" value="<?php echo
get_option("post_resum"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància metadata/camps afegits article')
?></label>
                        <input type="text" name="post_metadata" value="<?php echo
get_option("post_metadata"); ?>" />
                    </li>
```

```
                    <li>
                        <label><?php _e('Rellevància nom url article') ?></label>
                        <input type="text" name="post_slug" value="<?php echo
get_option("post_slug"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància tags article') ?></label>
                        <input type="text" name="post_tags" value="<?php echo
get_option("post_tags"); ?>" />
                    </li>
                    <li>
                        <label><?php _e('Rellevància titol article') ?></label>
                        <input type="text" name="post_titol" value="<?php echo
get_option("post_titol"); ?>" />
                    </li>
                </ul>
            </li>
            <li>
                <p> </p>
                <h3><?php _e('Opcions generals cercador'); ?></h3>
                <p><?php _e('Si es modica qualsevol opció de les següents s\'ha de
regenerar l\'index'); ?></p>
                <ul>
                    <li>
                        <label><?php _e('Pàgines a excloure') ?></label>
                        <input type="text" name="excloure_pagines" value="<?php echo
get_option("excloure_pagines"); ?>" />
                        <em><?php _e('( Identificadors de articles separats per coma
)'); ?></em>
                    </li>
                    <li>
                        <label><?php _e('Cateories a excloure') ?></label>
                        <input type="text" name="excloure_categories" value="<?php echo
get_option("excloure_categories"); ?>" />
                        <em><?php _e('( Identificadors de categories separats per coma
)'); ?></em>
                    </li>
                    <li>
                        <label><?php _e('Marcar el text trobat als resultats');
?></label>
                        <input type="checkbox" name="marcar_text" id="marcar_text"<?php
if (get_option('marcar_text'))
    echo ' checked="checked"'; ?>/>
                    </li>

                </ul>
            </li>

            <li>
                <br/><br/>
                <input class="button-primary" type="submit" name="save" value="<?php
_e('Guardar'); ?>"/>
            </li>

        </ul>
    </form>


    <p> </p>

    <h3><?php _e('Indexació'); ?></h3>
    <?php if ($motor->numero_indexats() == 0) : ?>
        <p style="clear: both"><?php _e('No hi ha contingut indexat. S\'ha de
reindexar'); ?></p>
<?php else : ?>
            <p style="clear: both"><?php _e('Hi ha '); ?><strong><?= $motor-
>numero_indexats() ?></strong><?php _e(' elements indexats.'); ?>
            </p>
<?php endif; ?>

            <p><?php _e('cerques fetes amb aquest index: '); ?> <?=
get_option('num_cerques') ?></p>
            <p> </p>
            <p>
                <input class="button-primary" type="submit" name="generar_index"
value="<?php _e('Regenerar index!'); ?>"/>
                <img id="loading" style="vertical-align: middle; display: none"
src="<?php echo path_join(WP_PLUGIN_URL, basename(dirname(__FILE__))) . '/loading.gif';
?>" width="16" height="16" alt="Small"/>
```

```
            </p>

            <div id="wrapper" style="display: none">

            </div>
        </div>


        <script type="text/javascript" charset="utf-8">
            jQuery(function() {
                jQuery('#wrapper').inicia( {
                    start:    jQuery('input[name=generar_index]'),
                    cancel:   '<?php _e('Cancelar'); ?>',
                    url:      '<?php echo admin_url('admin-ajax.php') ?>',
                    nonce:    '<?php echo wp_create_nonce('cercadoradvanced-index') ?>',
                    finished: '<?php _e('Fet!'); ?>'
               });
               });
       </script>
```

utils.php

```
<?php

/*
 * grup de funcions utils per diferents llocs
 */

/*
 * Funcio elimina html
 */

function neteja_html($text) {

    $text = @html_entity_decode($text, ENT_NOQUOTES, get_option('blog_charset'));    //
Remove all HTML
    return wp_kses(stripslashes($text), array());    // Remove all HTML
}

/*
 * Funció que neteja el text per a poder-ho indexar
 */

function neteja($text) {

    //guarda el contingut alternatiu
    preg_match_all('/ href=["\'](.*?)["\']/iu', $text, $href);
    preg_match_all('/ alt=["\'](.*?)["\']/iu', $text, $alt);
    preg_match_all('/ title=["\'](.*?)["\']/iu', $text, $title);

    // Elimina javascript i html  xungu
    $text = preg_replace('/<script(.*?)<\/script>/s', '', $text);
    $text = preg_replace('/<!--(.*?)-->/s', '', $text);
    // Afegeix un espai abans dels tags per a que separi be les paraules
    $text = str_replace('<', ' <', $text);
    $text = preg_replace('/&#\d*;/', '', $text);
    $text = addslashes(wp_kses(stripslashes(neteja_html($text)), array()));
    $text = preg_replace('/&\w*;/', ' ', $text);                    // Removes entities
    $text = str_replace("'", '', $text);
    $text = str_replace('&shy;', '', $text);
    $text = preg_replace('/[\'!;#$%&\,_\+=\?\(\)\[\]\{\}\"<>`]/', ' ', $text);

    if (count($href) > 0)
        $text .= ' ' . implode(' ', $href[1]);

    if (count($alt) > 0)
        $text .= ' ' . implode(' ', $alt[1]);

    if (count($title) > 0)
        $text .= ' ' . implode(' ', $title[1]);

    while (preg_match('/\s{2}/', $text, $matches) > 0)
        $text = preg_replace('/\s{2}/', ' ', $text);

    $text = str_replace('"', '', $text);
```

```
    $text = str_replace(get_option('home'), '', $text);
    return stripslashes(trim($text));
}

?>
```

cercador.js

```
CercadorAdvanced = {
    error_message : function() {
        alert('Error al fer petició');
    }
};


(function($){
    $.fn.inicia = function(args){
        args = args || {};

        return this.each(function() {

            function timer() {
                if (running) {
                    $.ajax({
                        url: opts.url,
                        cache: false,
                        data: ({
                            action: 'reindexa',
                            offset: offset,
                            limit: opts.limit,
                            _ajax_nonce: opts.nonce
                        }),
                        type: 'POST',
                        error: CercadorAdvanced.error_message,
                        success: function(data) {
                            var parts   = data.split(' ');
                            var left    = parseInt(parts.shift());
                            var percent = parts.shift();
                            var message = parts.join(' ')

                            $(wrapper).find('p').html(message);
                            $('#' + opts.inside).css('width', percent);

                            if (left > 0) {

                                offset += opts.limit;
                                setTimeout(timer, 0);
                            }
                            else {

                                $(opts.start).val(original);
                                $(opts.loading).hide();
                                running = false;
                            }
                        }
                    });
                }
            }

            var opts = $.extend({
                cancel: 'Cancel',
                inside: 'inner',
                limit: 20,
                nonce: '',
                loading: '#loading'
            }, args);

            var offset  = 0;
            var running = false;
            var wrapper = this;
            var original = $(opts.start).val();

            $(opts.start).click(function() {
                var button = this;

                if (running) {
```

```
                    running = false;
                    $(button).val(original);
                    $(opts.loading).hide();
                }
                else {
                    offset = 0;
                    running = true;


                    $(button).val(opts.cancel);
                    $(opts.loading).show();


                    $(wrapper).empty();
                    $(wrapper).append('<div id="' + opts.inside + '"></div><p></p>');
                    $(wrapper).fadeIn();
                    $('#' + opts.inside).css('width', '0px');

                    setTimeout(timer, 0);
                }

                return false;
            });
        });
    };
})(jQuery);
```

Resultats / proves

Per assegurar el correcte funcionament del nostre plugin s'han realitzat diferents proves que podem dividir en dos grups.

Proves funcionals

Cerques simples en contigut

Després de fer diferents cerques hem pogut apreciar que les cerques simples amb el nostre cercador, ja cerquen a tot el contingut del wp i no només a dins dels articles. Alhora les cerques de paraules separades amb espai, per exemple Arte Santa Monica, cerca tots els resultats en que aparegui Arte, Santa, o Monica, i els elements en els que existeixen més ocurrències apareixen primer. Aquest estil de cerca s'adequa a l'estil de google, i per tant es millora la usabilitat del sistema. Els resultats són els esperats, i no com amb el cercador per defecte.

En la següent imatge podem veure els resultats de Arte Santa Monica

Operands lògics

En el nou cercador, s'ha afegit la opció de cerques amb operadors lògics. Aquesta és una funcionalitat que permet definir amb molta precisió la cerca i per tant ajuda a trobar els resultats que cerquem.

A continuació podem veure les proves realitzades per a cada un dels operands que hem afegit. AND,OR, i cerques entre cometes per a buscar literals.

AND

Cerca Pere AND Barnola, em troba l'únic article en el que apareix el terme Pere i el terme Barnola.

OR

Cerca Pere OR Barnola, em troba els articles en els que apareix el terme Pere o el terme Barnola.

Literals

Al igual que en la majoria de cercadors estandar, el nostre cercador permet fer cerques de literals. Es a dir permet cercar un conjunt de termes de forma exacta, posant els termes entre cometes.

Com podem veure en la següent exemple cercant el literal "falta de un observatorio", retorna el resultat adequat.

Exclusions de contingut

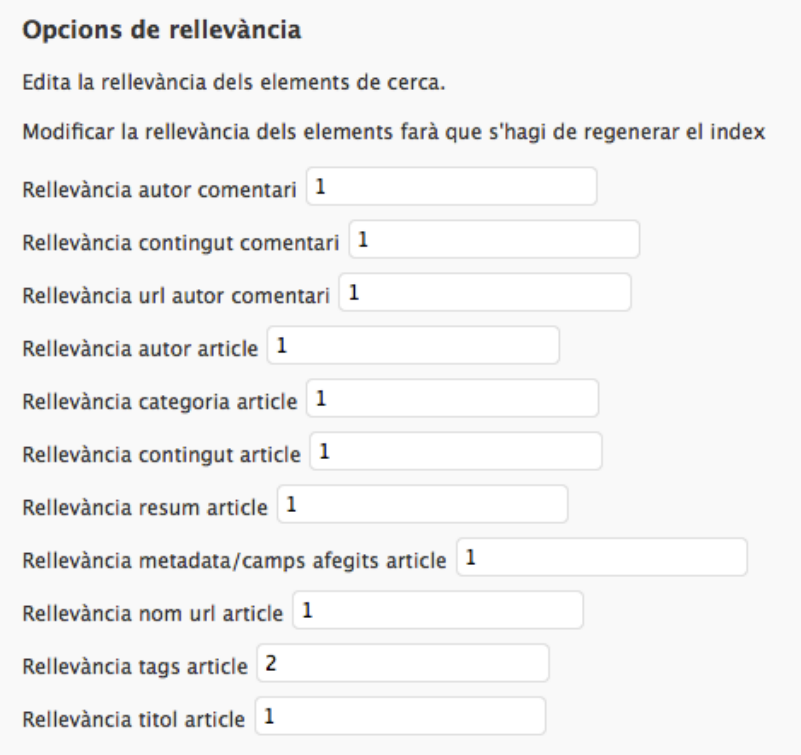
Des de la pàgina d'opcions del cercador podem especificar els identificadors de les pàgines i categories que volem excloure del index. Un cop modificades aquestes opcions i regenerat el index podem veure que les exclusions indicades es realitzen de forma correcta.

Després de excloure la pàgina "2326" que correspon a l'únic resultat de la cerca Pere AND Barnola, en la següent pantalla podem veure que amb la mateixa cerca ja no es troba el resultat, i per tant podem concloure que es sistema d'exclusió de contingut funciona correctament.

Rellevància

S'ha revisat el funcionament del càlcul de rellevància modificant els pesos per a cada un dels camps de cerca i comprobant que dins de l'ordre de rellevància sempre surten primers els dels camps prioritzats.

En el següent exemple podem veure com havent creat 2 articles, un amb el tag pere, i l'altre amb el terme pere en el títol. Si apliquem els següents valors de rellevància.



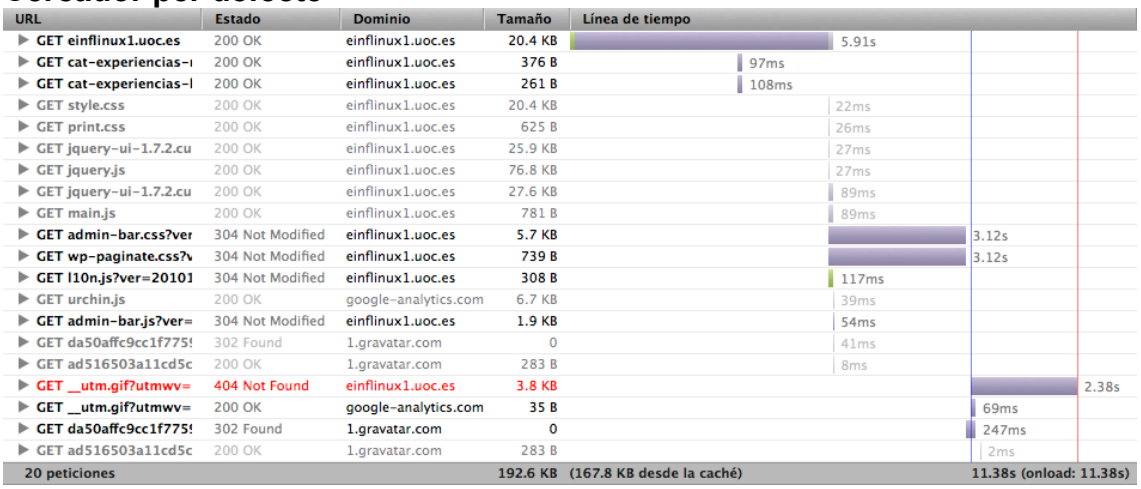
Surt primer el que te el terme en el tag que és el que havíem prioritzat.

Proves de rendiment

En quan a rendiment simplement s'ha fet la proba de temps per a la mateixa cerca. I els resultats de les proves obtingudes evidencien que el nou cercador, a més de ser molt més precís i donar millors resultats de cerca, és el doble de ràpid que el cercador per defecte de wp.

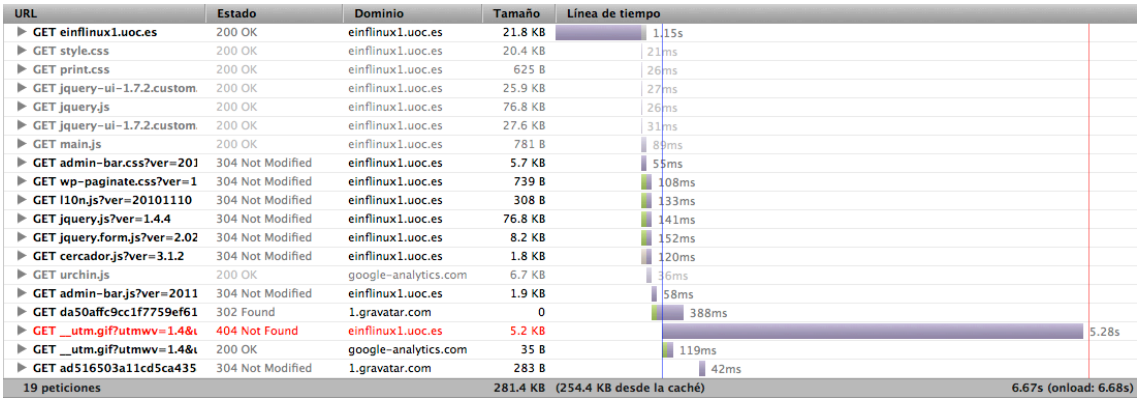
A continuació podem veure els temps de càrrega per a la cerca de la paraula "pere", amb el cercador per defecte de worpress i amb el nou cercador.

Cercador per defecte



URL	Estado	Dominio	Tamaño	Línea de tiempo
GET einflinux1.uoc.es	200 OK	einflinux1.uoc.es	20.4 KB	5.91s
GET cat-experiencias-i	200 OK	einflinux1.uoc.es	376 B	97ms
GET cat-experiencias-l	200 OK	einflinux1.uoc.es	261 B	108ms
GET style.css	200 OK	einflinux1.uoc.es	20.4 KB	22ms
GET print.css	200 OK	einflinux1.uoc.es	625 B	26ms
GET jquery-ui-1.7.2.cu	200 OK	einflinux1.uoc.es	25.9 KB	27ms
GET jquery.js	200 OK	einflinux1.uoc.es	76.8 KB	27ms
GET jquery-ui-1.7.2.cu	200 OK	einflinux1.uoc.es	27.6 KB	89ms
GET main.js	200 OK	einflinux1.uoc.es	781 B	89ms
GET admin-bar.css?ver	304 Not Modified	einflinux1.uoc.es	5.7 KB	3.12s
GET wp-paginate.css?v	304 Not Modified	einflinux1.uoc.es	739 B	3.12s
GET l10n.js?ver=20101	304 Not Modified	einflinux1.uoc.es	308 B	117ms
GET urchin.js	200 OK	google-analytics.com	6.7 KB	39ms
GET admin-bar.js?ver=	304 Not Modified	einflinux1.uoc.es	1.9 KB	54ms
GET da50affc9cc1f775!	302 Found	1.gravatar.com	0	41ms
GET ad516503a11cd5c	200 OK	1.gravatar.com	283 B	8ms
GET __utm.gif?utmwv=	404 Not Found	einflinux1.uoc.es	3.8 KB	2.38s
GET __utm.gif?utmwv=	200 OK	google-analytics.com	35 B	69ms
GET da50affc9cc1f775!	302 Found	1.gravatar.com	0	247ms
GET ad516503a11cd5c	200 OK	1.gravatar.com	283 B	2ms
20 peticiones			192.6 KB	(167.8 KB desde la caché) 11.38s (onload: 11.38s)

Nou cercador



URL	Estado	Dominio	Tamaño	Línea de tiempo
GET einflinux1.uoc.es	200 OK	einflinux1.uoc.es	21.8 KB	1.15s
GET style.css	200 OK	einflinux1.uoc.es	20.4 KB	21ms
GET print.css	200 OK	einflinux1.uoc.es	625 B	26ms
GET jquery-ui-1.7.2.custom	200 OK	einflinux1.uoc.es	25.9 KB	27ms
GET jquery.js	200 OK	einflinux1.uoc.es	76.8 KB	26ms
GET jquery-ui-1.7.2.custom	200 OK	einflinux1.uoc.es	27.6 KB	31ms
GET main.js	200 OK	einflinux1.uoc.es	781 B	89ms
GET admin-bar.css?ver=201	304 Not Modified	einflinux1.uoc.es	5.7 KB	55ms
GET wp-paginate.css?ver=1	304 Not Modified	einflinux1.uoc.es	739 B	108ms
GET l10n.js?ver=20101110	304 Not Modified	einflinux1.uoc.es	308 B	133ms
GET jquery.js?ver=1.4.4	304 Not Modified	einflinux1.uoc.es	76.8 KB	141ms
GET jquery.form.js?ver=2.02	304 Not Modified	einflinux1.uoc.es	8.2 KB	152ms
GET cercador.js?ver=3.1.2	304 Not Modified	einflinux1.uoc.es	1.8 KB	120ms
GET urchin.js	200 OK	google-analytics.com	6.7 KB	36ms
GET admin-bar.js?ver=2011	304 Not Modified	einflinux1.uoc.es	1.9 KB	58ms
GET da50affc9cc1f7759ef61	302 Found	1.gravatar.com	0	388ms
GET __utm.gif?utmwv=1.4&u	404 Not Found	einflinux1.uoc.es	5.2 KB	5.28s
GET __utm.gif?utmwv=1.4&u	200 OK	google-analytics.com	35 B	119ms
GET ad516503a11cd5ca435	304 Not Modified	1.gravatar.com	283 B	42ms
19 peticiones			281.4 KB	(254.4 KB desde la caché) 6.67s (onload: 6.68s)

Conclusions

Les conclusions són molt positives ja que s'ha aconseguit complir amb tots els objectius plantejats. S'ha aconseguit un plugin de cerca per a wordpress, ràpid, funcional, i configurable. En definitiva un cercador que millora a nivell de rendiment i funcionalitat el cercador per defecte de wordpress.

Per altra banda, queden moltes lines de futur obertes per a seguir-hi treballant. És va començar a crear un sistema de marcat de cerca, però que no va ser possible finalitzar en els terminis d'entrega. També es podrien afegir noves opcions al cercador que permetessin crear un vista de cerca avançada i alhora que permetes ordenar el resultats per altres conceptes a part de la rellevància, com podria ser la popularitat.

En resum s'ha fet una bona feina de base però alhora es pot seguir ampliant per aconseguir més opcions i funcionalitats.

Annexes

Annex 1: Recursos utilitzats

Recursos online

- http://codex.wordpress.org/

  Referència general oficial de wordpress, on es poden trobar diferents tutorials i la documentació de totes les clases

- http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html

  Referència de l'ús de índexs full text search

- http://www.php.net/

  Referència oficial de php

- http://netbeans.org/

  Pàgina referència del editor de text utilitzat per al desenvolupament

Llibres

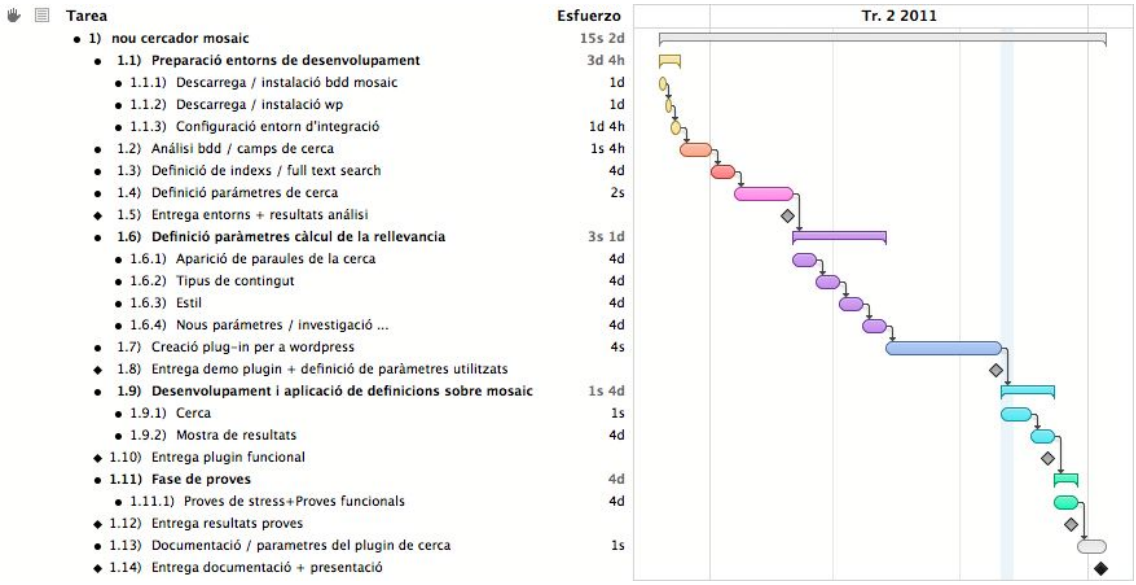
- WordPress Bible

  Aaron Brazell
  2010 Wiley Publishing,Inc

- Professional WordPress Plugin Development

  Brad Williams, Richard Justin Tadlock
  2011 Wiley Publishing,Inc

Annex 2: Planificació del projecte



Annex 3: Com crear un plugin per a wp

Introducció

No podem oblidar que el nostre objectiu principal és el de crear un plugin per a wordrpress, i per tant el primer pas és conèixer a fons l'entorn i quins són els passos per a poder-ho fer.

Utilitzant la API (Application Program Interface) que ens subministra la plataforma, podrem crear noves funcionalitats tant per a la part privada com a la part pública dels nostres llocs webs.

També revisarem el que són els "hooks" que es poden tractar d'accions (actions) o filtres (filters) i són una eina bàsica que utilitzarem per incrustar els nostres trossos de codi dins del worpress. Aquests trossos de codi, podran interactuar amb les dades de formularis via POST, accedir a la base de dades de wordpress, afegir noves pàgines en l'administració i en general estendre funcionalitats en moltes parts del wordpress.

Mentre anem avançant en el procés de crear el nostre plugin personalitzat per a wordpress intentarem centrar-nos en els conceptes bàsics generals per tal de que aquesta part de la documentació pugui servir per a un plugin general i no només en el que pretenem realitzar nosaltres.

Preparació de l'entorn de desenvolupament

El WP esta desenvolupat amb la plataforma de desenvolupament open source PHP, i per tant haurem de preparar un servidor web amb mysql i que permeti executar php. Una bona opció per treballar en local és el MAMP,XAMP.

Un cop preparat el servidor, s'ha d'instal·lar el wp. Instal·lar el wp es una tasca molt senzilla i directa. A continuació explicarem el procés, tot i que es pot trobar una guia molt complerta a http://codex.wordpress.org/installing_wordpress on es pot trobar solucions a possibles problemes durant la instal·lació o instal·lacions en entorns diferents al estable.

Assumint que ja es te el servidor web amb php y mysql, els passos seran els següents:

1.- crear una bdd
2.- descarregar el wp, descomprimir-lo i copiar-ho a l'arrel al servidor
3.- executar l'arxiu install.php a traves del navegador on es demanaran les dades per a configurar i accedir a la bdd i les dades bàsiques del nou wp

Un cop executats aquests passos podrem accedir a l'administració del wp a traves de (http://elnostreservidor/wp-admin)

Com a eina per editar codi i els fitxers que haurem de modificar, es recomana el Netbeans o l'Eclipse, tot i que es pot fer servir qualsevol editor de text.

Conceptes bàsics abans de començar

Els plugins

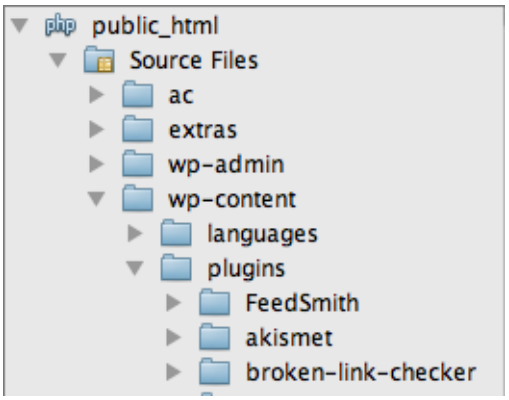
Una de les raons de que el wp sigui una eina tan potent i utilitzada és per la seva capacitat d'afegir-li noves funcionalitats, i la raó d'això és la seva API. Així doncs revisarem que són els plugins, i perquè son tan útils per als administradors, desenvolupadors i usuaris de wp.

El directori de plugins el podem trobar a http://www.wordpress.org/extend/plugins/, tots els plugins que hi existeixen estan distribuïts per wp tot i que la majoria estan desenvolupats per programadors o empreses independents, i tots són open-source, el que vol dir que es poden descarregar i utilitzar lliurement per als nostres projectes.
El fet de que siguin open-source, vol dir que podem modificar el codi, i per tant per als programadors ens pot ser molt útil partir de un plugin ja desenvolupat i modificar-lo per a que s'adapti a les nostres necessitats.

Actualment hi ha més de 12000 plugins disponibles que s'instal·len de forma molt senzilla amb no més de 3 clics.

El procés d'instal·lació d'un plugin és un procés força directe, quan s'instal·la un plugin el que es fa internament, és guardar una carpeta o un fitxer en un directori específic del wp. A vegades podem descarregar plugins des de pàgines de tercers que podem instal·lar des de l'opció "upload" dins del procés

“add new” de plugins. El que fa aquest procés es descomprimir el zip del plugin i descomprimir-ho dins de la carpeta plugins.



Com es pot veure a la imatge anterior en la carpeta wp-content>plugins es troben els plugins que tenim instal·lats.

Els “hooks”

Existeix un element clau a l’hora de desenvolupar plugins per wp, i aquest és el hook. Els hooks són coneguts com accions o filtres, i bàsicament són punts dins del cercle d’execució del wp als que es pot especificar que executin altres troços de codi. Per tant serà la manera bàsica d’afegir funcionalitat al wp sense tocar res dels seus processos bàsics. Per exemple podem especificar que s’executi codi en el moment que un usuari demana una pàgina, quan un administrador la crea, quan un autor envia un “post”, i en molts altres punts d’execució del wp.

El concepte doncs és registrar una funció que haguem desenvolupat a qualsevol d’aquests hooks, i per tant aquesta funció s’executarà quan el hook en concret passi.

Per exemple quan un usuari crea un post, existeix un hook, si nosaltres volem que cada cop que algú crei un post s’enviï un email, registraríem una funció d’enviament d’email a aquest hook.
Podriem dir que el procés de creació d’un plugin consisteix en cercar el hook adequat on volem que s’executi el nostre codi, crear una funció amb el codi, i registrar la funció al corresponent hook.

A http://codex.wordpress.org/Plugin_API podem trobar un explicació detallada de que són les “actions” i els “filters” i diferents exemples per aclarir els conceptes.

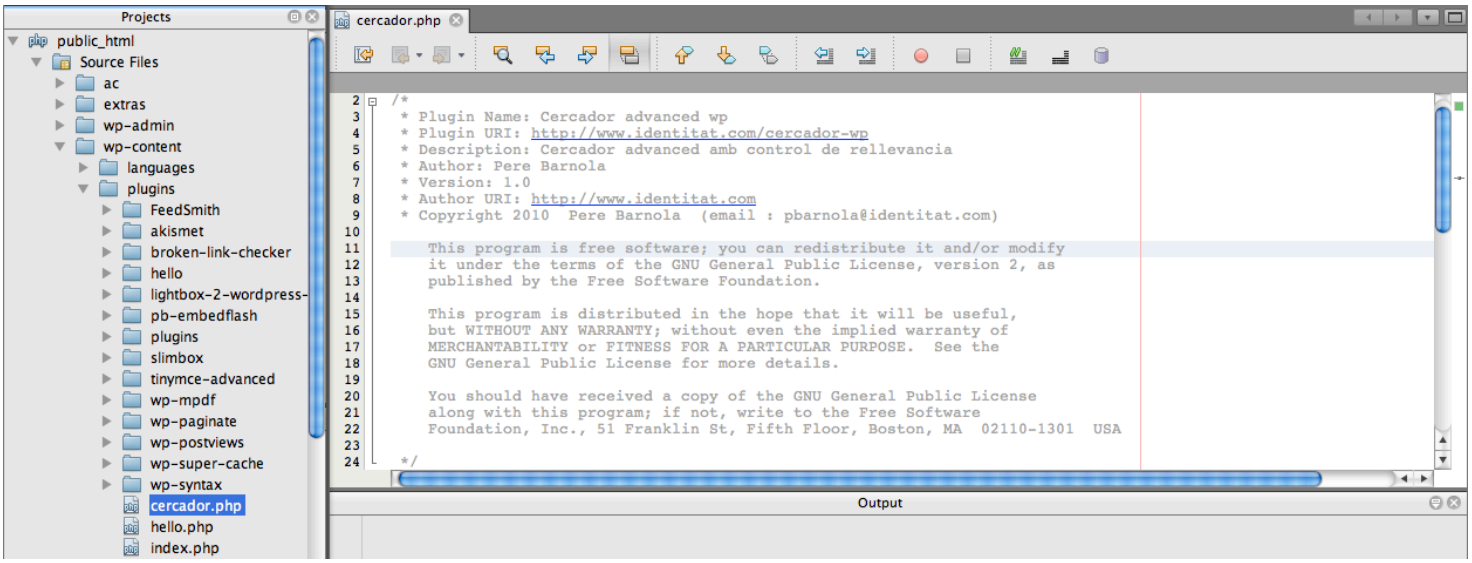
El fitxer base

Per començar a crear el nostre primer plugin per a wordpress començarem per el fitxer php principal. Aquest fitxer te una secció predefinida amb unes capçaleres que serveixen per a que el wp pugui identificar e instal·lar el plugin.

Mans a la feina doncs.

Crearem un fitxer cercador.php dins del directori pluigns i afegirem la secció de comentari que tal i com podem veure a la següent figura conté les següents seccions:

- Plugin Name: El nom del nostre plugin
- Plugin URI: La web on tindrem informació sobre el nostre plugin
- Description: La descripció del nostre plugin
- Author: L'autor
- Version: La versió del codi
- Author URI: la url de l'autor



Si ara anem a l'administració i comprovem la secció de plugins veurem que ja tenim el plugin registrat amb la informació especificada.

Com veiem no esta activat, i encara que ho estes no faria res per que no te cap codi.

Una altra cosa a tenir en compte, és la llicència. Com podeu veure en la imatge amb el codi hi ha un tros de codi amb la llicència estàndard "GNU general public liscence version 2". La podeu trobar a http://codex.worpress.org/Writing_a_Plugin#License

No es obligatori que s'inclogui, però és molt recomanable fer-ho sobretot si publicareu el vostre plugin a la pàgina de wp.

Aquesta és la primera part a l'hora de crear un plugin, afegir la part de capçalera amb tota la informació del plugin.

Els "hooks", accions i filtres

Com ja hem vist els hooks són la base del plugins, existeixen dos tipus de hooks, les accions i els filtres. Cadascun d'ells serveix per a una cosa diferent. Les accions s'executen en qualsevol tipus de tasca en els processos de wp, en canvi els filtres són específicament per als continguts, abans d'afegir-los a la base de dades o enviar los a pantalla.

Si mirem la referencia, veurem que existeixen funcions especifiques per a filtres. Les funcions claus són "has_filter" que ens dirà si un filtre existeix, "add_filter" que s'utilitza per afegir un nou filtre, i "remove_filter" per a esborrar un dels filtres existents.

Hi han diferents categories de filtres. Els pots trobar a la referència de filtres a http://codex.wordpress.org/Plugin_API/Filter_Reference . Podem trobar filtres per "lectura y escriptura a base de dades", "comentaris", "trackbacks", "temps i data", "autor i usuari", i per tots els altres llocs del wp on s'introdueix informació. Per tant, cada cop que es vulgui modificar text abans d'introduir o enviar informació a la base de dades o al navegador aquest és el lloc on fer-ho.

Per altra banda les accions són on s'executen tasques especifiques del wp, quan alguna cosa passa, quan un comentari s'envia, quan es crea un menú,... tot això serien tasques. Les accions funcionen lleugerament diferent als filtres, ja que no tenen que veure només en els processos relacionats amb introducció de text, normalment tenen a veure amb alguna funció que esta passant. Però la

manera com funcionen és pràcticament igual a la del filtres. “has_action” ens dirà si l’acció existeix, “add_action” amb la que podrem relacionar la nostra funció amb la funció específica, i “remove_action” que ens permetrà eliminar una acció concreta. Al igual que amb els filtres la referència, que podem trobar a http://codex.wordpress.org/Plugin_API/Action_Reference, esta dividida per tipologies. Per exemple, accions que ocorren en una petició típica (“typical request”), el get_sidebar, que s’executa quan es demana el lateral de la pàgina, i molts d’altres.

Aquest són els dos tipus de hooks que existeixen, i la part important a recordar és que els filtres s’utilitzen quan volem executar una funció en el moment en que s’interactuï o es registri text, i les accions per a la resta de funcionalitats que ocorren en el wp.

Instal·lació i activació

El primer pas per a un administrador per a utilitzar un plugin és instal·lar-lo, però per a que el plugin pugui funcionar s’ha d’activar.

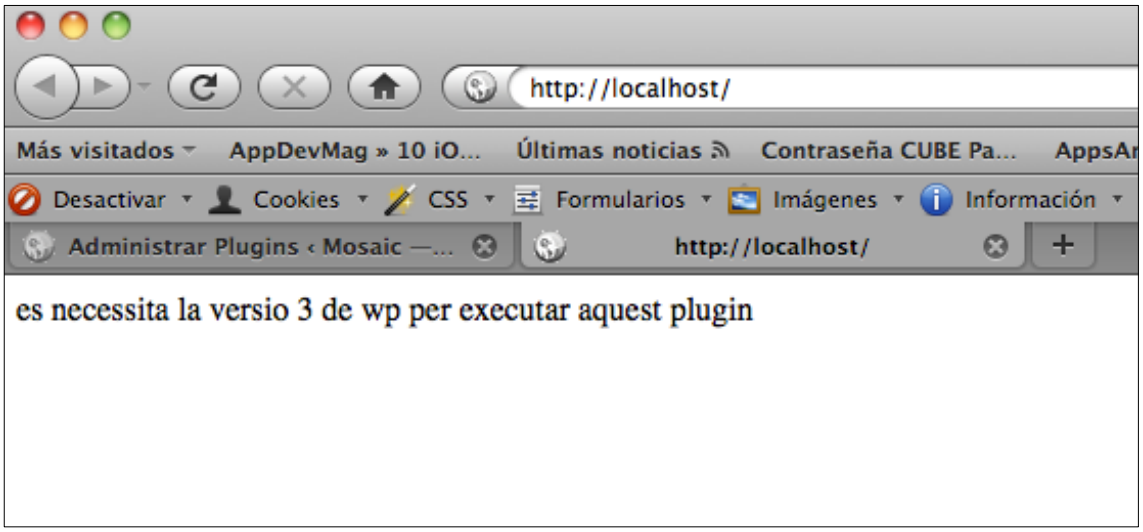
El moment de l’activació és un moment important, ja que normalment necessitarem fer algunes accions per preparar els elements necessaris per al nostre plugin, com per exemple donar d’alta una taula en la base de dades. El que fa l’activació, és permetre que s’executi el codi que em creat en el nostre fitxer, es a dir registrar els hooks necessaris i executar les nostres funcions. El fet de poder activar i desactivar els plugins, permet als administradors activar o desactivar les extensions de manera molt àgil.

Com hem vist abans, el procés d’instal·lar un plugin, és simplement el procés de copiar un fitxer o carpeta a un directori concret del wp. A aquests fitxers no se’ls hi fa cap cas fins que el plugin no s’activa. El fet d’activar el plugin vol dir que els fitxers del plugin s’inclouran en cada execució del wp.

Anem al nostre plugin que hem creat abans. Per provar que el plugin s’executa posarem el següent codi que comprova la versió del wp que tenim i executa una funció en el cas de que la versió sigui inferior a la 3.2. Posteriorment activarem el plugin des de l’administrador i veurem que s’executa.

```php
<?php
/*
 * Plugin Name: Cercador advanced wp
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: Cercador advanced amb control de rellevancia
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License, version 2, as
    published by the Free Software Foundation.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

 */

global $wp_version;

if(!version_compare($wp_version,"3.2",">="))
{
    die("es necessita la versio 3 de wp per executar aquest plugin");
}

?>
```

un cop activat, si executem el wp hauríem de veure el següent missatge



d'aquesta manera comprovem que el plugin s'executa un cop activat.

Quan s'activa un plugin s'inclou el codi del plugin a l'execució del wp. El wp ens permet l'opció d'executar codi addicional en el moment de l'activació o desactivació del plugin.

Per exemple, si el nostre plugin necessita noves taules a la base de dades o bé si volem modificar algunes opcions de configuració o demanar certes versions de php o wp per poder executar el nostre plugin.

El moment ideal per a executar i preparar tot el que necessitarem per l'execució del plugin és el moment de l'activació. De la mateixa manera també podem executar codi addicional en el moment de la desactivació del plugin.

La manera de fer-ho és mitjançant les funcions "register_activation_hook" i "register_deactivation_hook". Les dues ens permeten especificar la funció a executar en el moment de l'activació o desactivació per a un fitxer concret. Com a primer paràmetre reben el fitxer, que normalment serà el fitxer on executarem el codi, per el que s'utilitza la variable __FILE__ de php. Com a segon paràmetre reben la funció que volem que s'executi en el moment de l'activació o en el moment de la desactivació.

Si retornem al nostre plugin i creem dos funcions una amb tot el que volem que passi en l'activació i una altra amb tot el que volem que passi en la desactivació. Per fer la proba registrarem en cada una de les funcions un missatge en el fitxer de logs. Un cop fetes les registrarem amb les funcions "register_activation_hook" i "register_deactivation_hook".

Així doncs el nostre codi hauria de quedar quelcom com això:

```php
<?php
/*
 * Plugin Name: Cercador advanced wp
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: Cercador advanced amb control de rellevancia
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License, version 2, as
    published by the Free Software Foundation.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

 */


function activacio_plugin() {
    // creo el que faci falta a l'inici
    error_log("s'ha activat el plugin");
}

function desactivacio_plugin() {
    error_log("desactivat");
}

register_activation_hook(__FILE__, "activacio_plugin");
register_deactivation_hook(__FILE__, "desactivacio_plugin");

```

Si ara anem a l'administració i desactivem el nostre plugin. En el nostre fitxer de logs hauria d'aparèixer quelcom com el següent:

[25-Apr-2011 11:24:36] desactivat

Si ara anem a l'administració i l'activem. Hauríem de veure en el fitxer de logs el següent:

[25-Apr-2011 11:26:07] s'ha activat el plugin

Així doncs ja sabem com podem executar codi addicional en el moment de l'activació o en el moment de la desactivació d'un plugin.

Les accions

Com ja hem vist, quan volem executar un codi en el moment que passi alguna cosa en el wp utilitzarem una acció. Per a fer-ho utilitzarem la funció add_action que rep com a primer paràmetre el nom de l'acció i com a segon paràmetre la funció que volem que s'executi cada cop que ocorri l'acció especificada.

Per exemple si volem que cada cop que algú editi un post s'enviï un email amb informació, farem el següent:

Obrim el fitxer del nostre plugin cercador.php, i crearem una funció amb el codi que volem que s'executi, en el nostre cas, el que farem és enviar un correu electrònic. Després afegirem l'acció amb el add_action. Per tant el nostre codi quedaria quelcom així.

```
41
42  function enviar_en_edicio() {
43      global $_REQUEST;
44
45      $to = "pbarnola@identitat.com";
46      $subject = "Edició del post " . $_REQUEST['post_title'];
47      $message = "Edició del post " . $_REQUEST['post_title'];
48
49      mail($to, $subject, $message);
50  }
51
    add_action('edit_post', 'enviar_en_edicio');
```

Si ho guardem, i activo el plugin, a partir d'aquest moment qualsevol edició sobre un post hauria d'enviar un email amb el contingut que hem especificat.

Els filtres

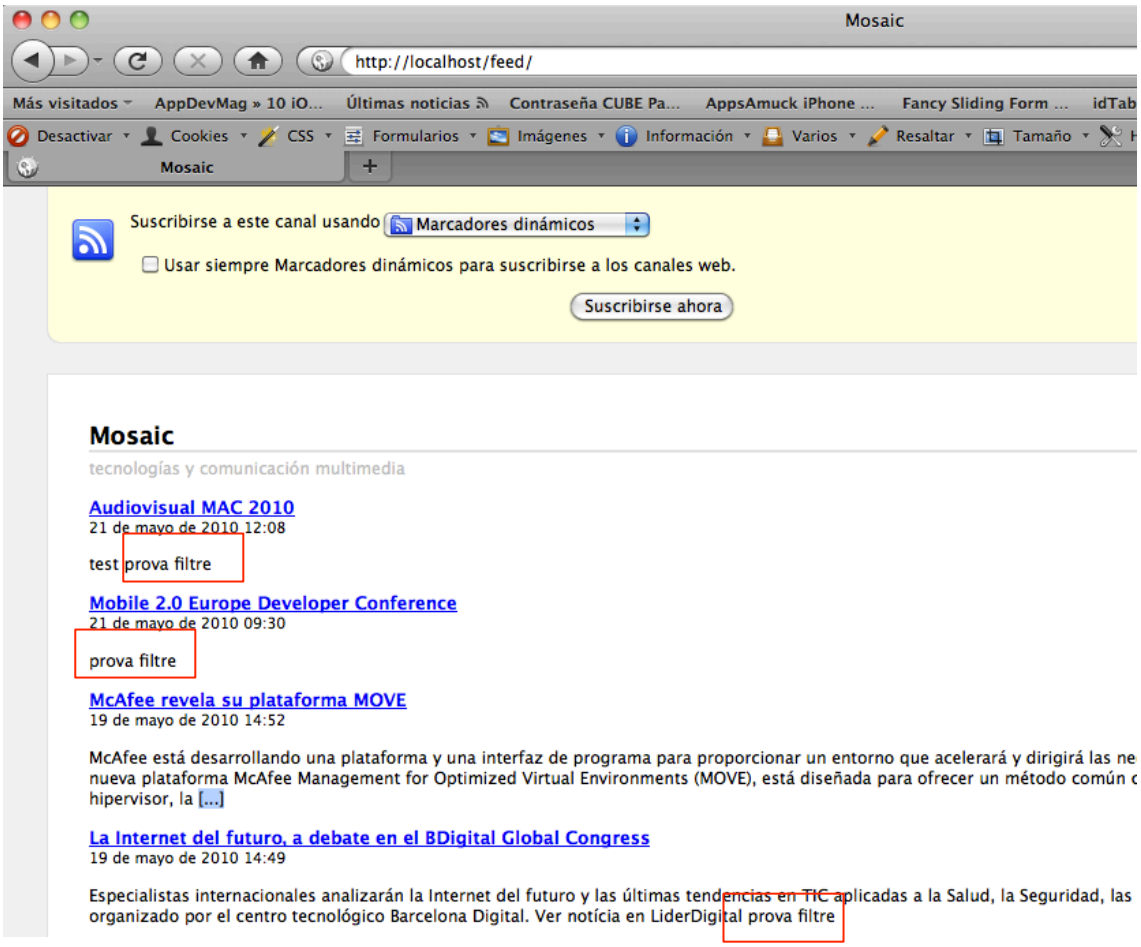
Com hem vist, els filtres són "hooks" que podem fer servir per modificar contingut de texte. Per a crear un filtre, ho farem amb la funció add_filter, però al igual que hem fet quan hem creat l'acció, primer de tot crearem una funció amb el codi que volem que s'executi quan ocorri el filtre. En el cas dels filtres, les funcions són lleugerament diferents ja que reben un paràmetre, que és el contingut.

El que farem és crear un filtre que ens afegeixi al final del contingut una cua si el contingut en si és un "feed" de rss.

El codi seria alguna cosa semblant al següent:

```
function afegir_cua_feeds( $contingut ){
    if(is_feed ())
    {
        $contingut = $contingut. "prova filtre";
    }
    return $contingut;
}

add_filter("the_content", "afegir_cua_feeds");
```

Si afegim el codi al nostre plugin, el guardem i ara visualitzem el contingut del nostre blog en format rss. Hauríem de veure que s'afegeix al final del contingut el text "prova filtre" tal i com ho hem especificat.



Les funcions Pluggable

Les funcions “pluggable”, són un conjunt de funcions permeten en un plugin sobreescriure algunes funcions del “core” de wp i per tant afegir noves funcionalitats a aquestes funcions. Es pot trobar el llistat d’aquestes funcions a la següent url http://codex.wordpress.org/Pluggable_Functions

Una bona manera de sobreescriure una funció, es agafar l’original i modificar-la per afegir la funcionalitat que necessitem.

Totes les funcions “pluggable”, les podem trobar dins del fitxer wp-includes>pluggable.php

Per exemple, abans quan hem creat l’acció que enviava un email, hem utilitzat la funció mail de php. Hi ha una funció “pluggable” que és wp_mail que és la que utilitza el wp per enviar emails internament (http://codex.wordpress.org/Function_Reference/wp_mail)

Imaginem que volguéssim afegir sempre un tag en el subject quan utilitzem la funció dins del nostre plugin. Agafaríem el codi de la funció original, el posaríem dins del nostre plugin i afegiríem en la línia del subject el tag que volguéssim.

Per tant ara quan cridéssim la funció wp_mail utilitzaríem la que ha estat sobreescrita per el plugin amb la funcionalitat específica.

```php
function enviar_en_edicio() {
    global $_REQUEST;

    $to = "pbarnola@identitat.com";
    $subject = "Edició del post " . $_REQUEST['post_title'];
    $message = "Edició del post " . $_REQUEST['post_title'];

    wp_mail($to, $subject, $message);
}

add_action('edit_post', 'enviar_en_edicio');
```

Els template tags

Els "template tags" són petites funcions que s'utilitzen en les plantilles (templates) de wp per poder realitzar un seguit de tasques. El que faran bàsicament és recollir i treure informació per pantalla. Per als desenvolupadors de plugins, és una altra manera d'afegir noves funcionalitats al wp.

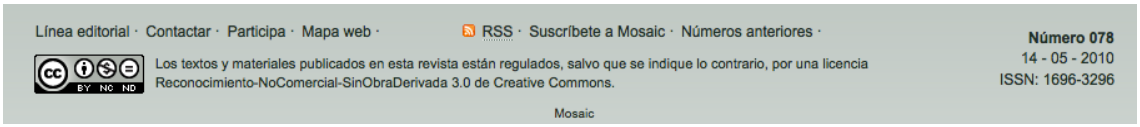
A http://codex.wordpress.org/Template_Tags podem trobar més informació sobre els template tags, i els que ja existeixen en el wp. Per exemple la funció <?php bloginfo("show"); ?> que serveix per recollir diferent informació del nostre blog.

Aquestes funcions es criden des dels nostres "templates" (plantilles). Des de la secció "Aparencia>Editor" del menú de l'àrea d'administració podrem accedir i editar el codi de la nostra plantilla activa. Si ho fem i editem el fitxer principal (index.php) i anem al final de tot, veurem que allà es criden ja alguns "template tags" com el get_sidebar() o el get_footer() que recuperen la informació de la barra lateral i del peu de pàgina respectivament.


```
                <div class="box4-foot">
                        <?php if(function_exists('wp_paginate')) { wp_paginate(); } ?>
                </div>
                <?php else : ?>
                <div class="box4-wrap">
                        <h2>No se han encontrado entradas</h2>
                        <?php get_search_form(); ?>
                </div>
                <div class="box4-foot">
                        <?php if(function_exists('wp_paginate')) { wp_paginate(); } ?>
                </div>
                <?php endif; ?>
        </div>
</div>

<?php get_footer(); ?>
```

Anem a afegir un altra. Afegirem "bloginfo('title')" que ens retornarà el títol del nostre blog. Tan senzill com això, si ara guardem i visualitzem el nostre blog,

veurem aquesta informació al final de la pàgina tal i com podem veure en la següent pantalla.



Com ja hem vist amb anterioritat, tot el contingut dels fitxers d'un plugin s'inclouen en l'execució del wp quan el plugin s'activa. I per tant si afegim una funció dins del nostre plugin, la podrem cridar des de qualsevol punt del wp i automàticament es convertirà en un "template tag". Així doncs anem a crear la nostra funció "template tag". Obrim el fitxer del nostre plugin (cercador.php) i afegirem una funció nou_text() que traurà per pantalla la data actual.

```
66  /*
67   * template tag
68   */
69  function nou_text(){
70      print("la data actual és ".date('Y'));
71  }
72
73  ?>
```

Si ara la cridem dins del fitxer de peu del nostre template (Apariencia->Editor->footer.php)

```
<?php nou_text(); ?>
```

Salvem el fitxer i recarreguem el nostre blog, hauríem de veure quelcom com el següent

Així doncs, ja sabem com podem crear funcions que poden ser cridades per els administradors del site des del template. El que és evident, és que per a que els administradors que utilitzin el nostre plugin sàpiguen de quines funcions disposen i com les poden cridar, haurem de crear una documentació prou complerta.

Els “shortcodes”

Els shortcodes són una manera de permetre als nostres usuaris, que no són desenvolupadors, el poder cridar certes funcionalitats. La manera de funcionar és molt semblant als tag d’html.

Per exemple, un de típic és el caption que ens incrusta un peu a la imatge dins d’un post.

[caption id=”22” align=”alignleft” width=”200” caption=”peu de pàgina”]
<img src=”la imatge”></img>
[/caption]

Básicament un shortcode, es basa en el shortcode, que vindria a ser com el tag d’html, els atributs que són equivalents als atributs dels tags html, i que en el cas de l’exemple correspondrien al “align”, “width”, ..., i el contingut que és el text o codi que va entremig del shortcode d’obertura [caption] i el shortcode de tancament [/caption]

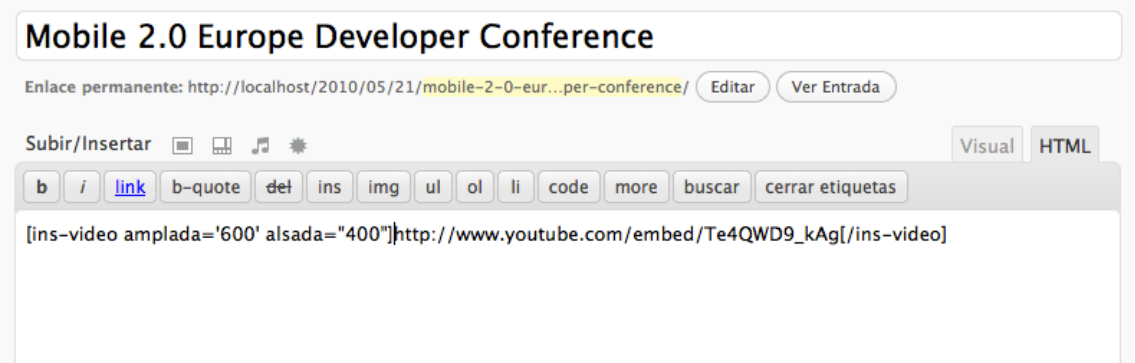
En els nostres plugins podem crear els nostres propis shortcodes per facilitar als administradors el fet d’afegir certes funcionalitats a dins d’un post.

Anem doncs a crear-ne un dins del nostre plugin. Obrim el fitxer “cercador.php” i primer crearem la funció amb la funcionalitat que vulguem donar-li al nostre shortcode. En aquest cas farem un shortcode per insertar un vídeo en els nostres posts. Per a crear un shortcode utilitzarem la funció add_shortcode, que rep dos paràmetres, el primer el tag a utilitzar per el shortcode, i el segon la funció que es cridarà quan es trobi aquest tag.

Les funcions que es criden a traves d’un shortcode, reben també dos paràmetres, el primer són els atributs que s’especifiquen al shortcode, i el segon, el contingut que hi ha dins del shortcode. En el següent codi podem veure com es fan les crides.

```
74  /*
75   * short code
76   */
77
78  function insertar_video($atts,$content=null){
79      shortcode_atts(array('amplada'=>'425','alsada'=>'349'), $atts);
80
81      return "<iframe width='".$atts['amplada']."' height='".$atts['alsada']."'
82          src='".do_shortcode($content)."' frameborder='0' allowfullscreen></iframe>";
83  }
84
85  add_shortcode('ins-video', 'insertar_video');
```

Per tant si ara cridem al nostre shortcode dins d’un post com en la següent imatge.



Hauríem de veure quelcom com el següent en la publicació de l'entrada.

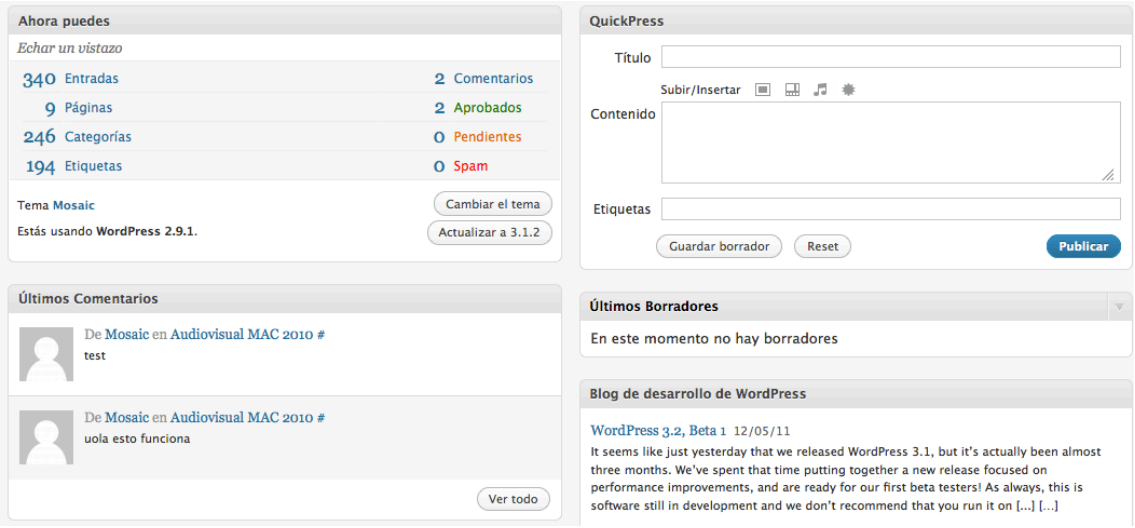
Modificant els atributs i els paràmetres podem configurar els paràmetres del vídeo tal i com ho hem programat.

Els shortcodes per tant, poden ser molt útils per als desenvolupadors de plugins.

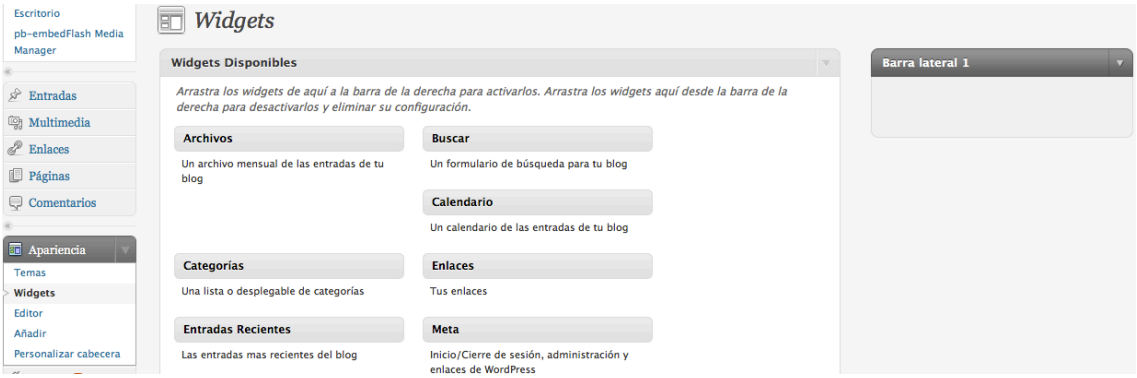
Utilització i creació de widgets

Els widgets son petites “finestres de funcionalitat” que es visualitzen en les diferents pàgines del wordpress, i que permeten als usuaris visualitzar o interactuar amb certa informació. Per exemple podem tenir un widget que ens visualitzi certes estadístiques dins de l’administració, o bé un widget en la barra lateral que ens mostri els posts més visitats.

Els widgets existeixen des de la versió 2.2 del wp. Si accedim a la pàgina principal de l’administració (escritori o dashboard) podrem veure diferents widgets.

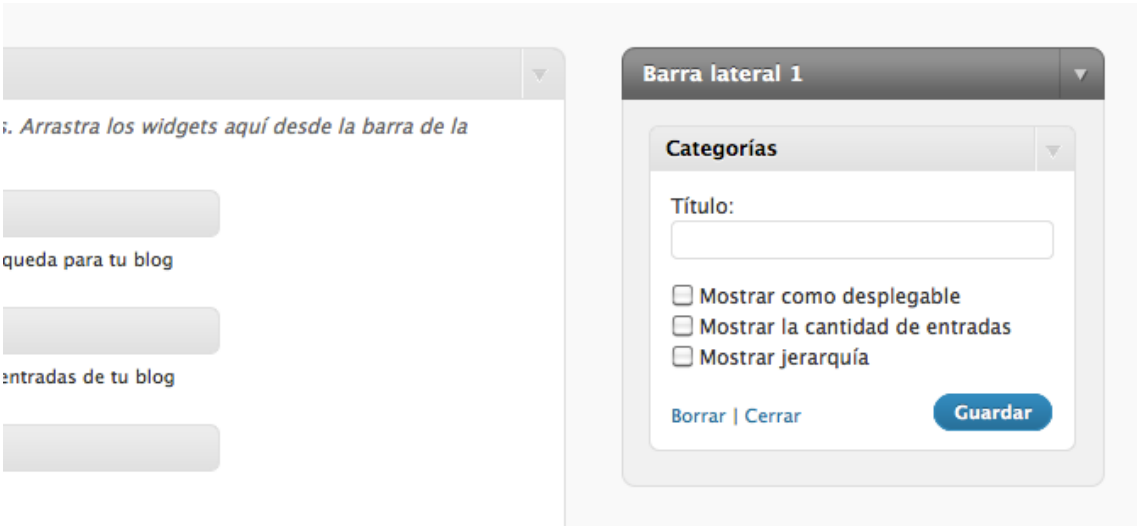


Si el que volem es gestionar o veure els widgets de la par pública accedirem al submenú “widgets” dins de la secció “apariencia” de l’àrea d’administració. A la part esquerra podrem veure tots els widgets disponibles i a la part dreta les seccions del nostre tema on podem afegir-los. Aquestes seccions variaran en funció del tema que utilitzem, fins hi tot pot haver un tema que no accepti aquesta funcionalitat.



Per afegir un widget a alguna de les àrees preparades, només cal que s’arrossegui el widget desitjat a l’àrea on es vol insertar. Des d’aquí podrem configurar les opcions del widget en concret, si es que aquest té opcions

configurables. Per eliminar un widget, simplement l'arrossegarem a la part de "widgets disponibles" de manera que s'esborrarà el widget i la seva configuració per complert, o bé el podem arrossegar a l'àrea de widgets inactius. D'aquesta manera mantindrem la configuració del widget per si necessitem tornar a utilitzar-lo.



Quines són les diferències entre els widgets i els plugins llavors? Doncs podríem dir que un widget sempre serà un plugin, però un plugin no te per que ser un widget. Un plugin és una part de codi que interactua amb el wp, i no ha de ser necessàriament visible. En canvi els widgets sempre són visibles i tècnicament són plugins.

El wp porta per defecte widgets que es poden instal·lar de forma senzilla sense coneixements de programació.

El procés per crear un nou widget és bàsicament el mateix procés que per crear un plugin.

De la mateixa manera que hem fet amb el plugin anem a crear doncs el nostre primer widget.
Crearem dins de la carpeta de plugins un nou fitxer testWidget.php.
La manera senzilla de crear un widget és heretant de la classe WP_Widget. Per tant en el nostre fitxer crearem una classe que heretarà de la classe WP_Widget. Per a que un widget funcioni necessitem 3 funcions bàsiques. widget() que visualitzarà la interfície d'usuari, update() que s'encarregarà d'actualitzar la funcionalitat que es necessiti, i form() que s'utilitzarà si necessitem alguna opció de configuració.

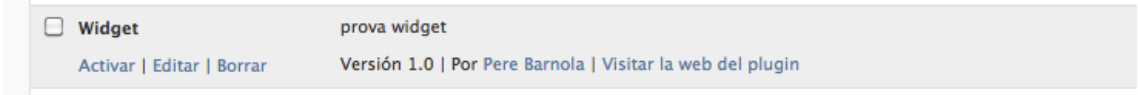
Podrem trobar la referencia de la classe en el següent enllaç
http://phpdoc.wordpress.org/trunk/WordPress/Widgets/WP_Widget.html

Per a que un widget es pugui utilitzar en el wp primer l'haurem de registrar. Per fer-ho cridarem a una funció on cridarem la funció register_widget("TestWidget") a la que li passarem el nom de la classe que hem creat.

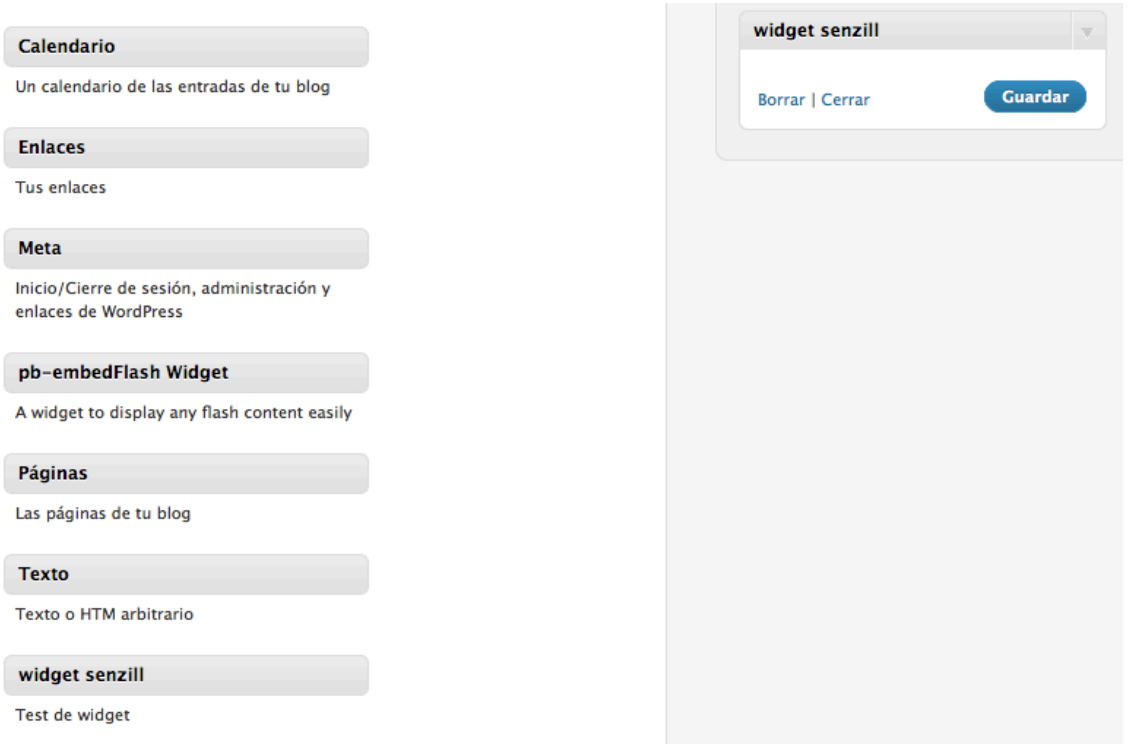
Finalment com fèiem amb els plugins crearem una acció per a la inicialització del widget que cridarà la funció que hem creat.

```
11
12  class TestWidget extends WP_Widget
13  {
14      /*
15       * constructor
16       */
17      function TestWidget(){
18          $opcions = array(
19              'classname'=>'TestWidget',
20              'description'=>'Test de widget',
21              );
22
23          //crida al constructor de la classe mare
24          parent::WP_Widget('TestWidget', 'widget senzill', $opcions);
25
26      }
27
29      function widget($args,$instance){
30          extract($args, EXTR_SKIP);
31          $titol = ($instace['titol']) ? $instance['titol'] : "plugin sencill";
32          $descripcio = ($instace['descripcio']) ? $instance['descripcio'] : "Descripció test";
33          print($abans_widget.$pre_titol.$titol.$post_titol."<p>".$descripcio."</p>");
34
35      }
37      function update(){
38
39      }
41      function form(){
42
43      }
44
45  }
46
47
48  function test_widget_init(){
49      register_widget("TestWidget");
50  }
51  add_action('widgets_init','test_widget_init');
```

Per tant si ara anem a l'àrea de plugins hauríem de veure el nostre widget sense activar.



Si l'activem i després anem al panell de widgets hauríem de veure quelcom com el següent.



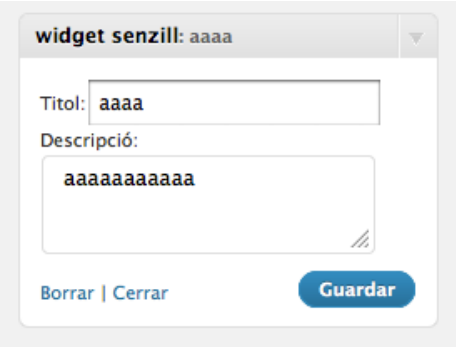
En alguns casos serà necessari poder oferir opcions de configuració per tal de que l'usuari pugui controlar certs paràmetres o funcionalitats dels widgets. Per a poder permetre afegir aquest tipus de configuració dels widgets utilitzarem les funcions form() i widget() de les que hem parlat abans.

Anem doncs a veure com es fa això.

En la funció form() afegirem el formulari per a poder donar d'alta els nostres paràmetres.
En la funció Update ens encarregarem de guardar aquests paràmetres. Podem cridar a la funció Update de la classe pare. Per tant si no necessitem fer cap validació especial podem eliminar la funció Update i es cridarà la funció Update de la classe pare. El codi resultant seria el següent.

```php
class TestWidget extends WP_Widget
{
    /*
     * constructor
     */
    function TestWidget(){
        $opcions = array(
            'classname'=>'TestWidget',
            'description'=>'Test de widget',
            );

        //crida al constructor de la classe mare
        parent::WP_Widget('TestWidget', 'widget senzill', $opcions);

    }

    function widget($args,$instance){
        extract($args, EXTR_SKIP);
        $titol = ($instace['title']) ? $instance['title'] : "plugin sencill";
        $descripcio = ($instace['body']) ? $instance['body'] : "Descripció test";
        print($before_widget.$before_title.$titol.$after_title."<p>".$descripcio."</p>");

    }

    function form($instance){
       print("<label for='".$this->get_field_id('title')."'>");
       print("Titol:");
       print("<input id='".$this->get_field_id('title')."' name='".$this->get_field_name('title')."' value='".esc_attr($instance['title'])."'/>");
       print("</label>");

       print("<label for='".$this->get_field_id('body')."'>");
       print("Descripció:");
       print("<textarea id='".$this->get_field_id('body')."' name='".$this->get_field_name('body')."'> ");
       print(esc_attr($instance['body'])."</textarea>");
       print("</label>");
    }
}

function test_widget_init(){
    register_widget("TestWidget");
}
add_action('widgets_init','test_widget_init');
```

Si ara visualitzem el widget hauríem de veure el següent.



Així doncs aquesta serà la manera de crear els nostres propis widgets per a wp.

Administració i opcions dels plugins

Moltes vegades els nostres plugins requeriran opcions de configuració, o bé que l'administrador hagi d'entrar certa informació per a que funcionin adequadament. Per exemple si ens hem de connectar a una base de dades externa, s'haurà d'introduir el usuari, el password i el servidor extern. Si es així probablement necessitarem crear una interfície que gestioni aquest tipus d'informació.

Hi ha tres passos bàsics per fer això.

1.- Crear una funció que visualitzi la interfície d'usuari
2.- Crear una funció que ens afegeixi una opció en el menú d'administració
3.- Utilitzar un hook per a fer que aquestes s'executin

En la primera creem el formulari necessari, en la segona afegim l'enllaç al menú i especifiquem els permisos. Podem veure les diferents opcions que

tenim de permisos al següent enllaç http://codex.wordpress.org/Roles_and_Capabilities . En el nostre cas utilitzarem el ‘manage_options’

El codi quedaria similar al següent

```
/*
 * interficie administrador
 */

function cercador_opcions(){
    print("<div class='wrap'>");
    screen_icon();
    print("<h2>Opcions cercador</h2>");
    print("<p>En aquesta secció es gestionen els paràmetres del plugin de cerca</p>");
    print("</div>");
}


/*
 * afegim la opció al menú indicant la funció amb la interficie a cridar
 */

function cercador_plugin_menu(){
    add_options_page('Opcions cercador', 'Opcions cercador', 'manage_options','cercador_plugin','cercador_opcions');
}

/*
 * registrem l'acció per a que es cridi
 */
add_action('admin_menu', 'cercador_plugin_menu');
```

Si ara accedim al administrador, veurem que ens apareix l’opció “opcions cercador” sota el menú “opciones” tal i com hem especificat. I si fem clic se’ns carrega la interfície que hem afegit.

Quan creem un plugin que requereix opcions de configuració, haurem de guardar aquestes opcions a la base de dades. Hi ha dues maneres de fer-ho.

1.- Si la informació que es vol guardar és molta, probablement es voldra crear una taula pròpia. Parlarem de com interactuar amb la base de dades de wp més endavant.

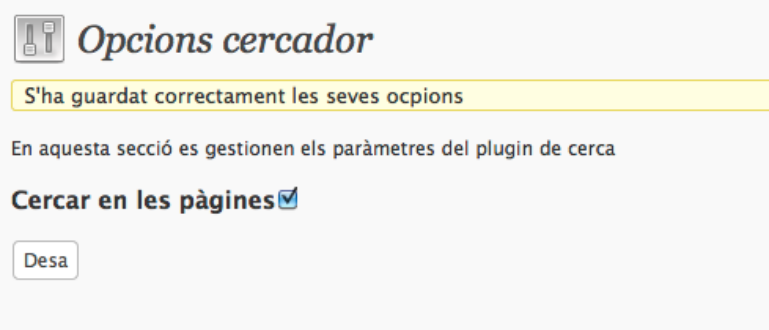
2.- L'altra manera és creant els nostres propis camps en la taula d'opcions del wp. Anem a veure com faríem això.

Per fer-ho crearem un formulari en la nostra interfície i afegirem els camps a la taula de options del wp amb la funció update_option. Per tal de que el formulari sigui segur, el wp ve amb una funcionalitat de sèrie per tal d'assegurar que no s'estigui enviant la informació del formulari des d'un altre lloc. Aquesta funcionalitat s'anomena "nonce", i el que fa es bàsicament incrustar un codi únic en el nostre formulari per tal de tenir un formulari segur.

Si veiem el codi, seria així:

```
function cercador_opcions(){

    if(check_admin_referer('cercador_opcions_update')){
        update_option('cercador_opcions_pages', $_POST['cerca_pagines']);
        print("<div id='message' class='updated'>S'ha guardat correctament les seves ocpions</div>");
    }

    print("<div class='wrap'>");
    screen_icon();
    print("<h2>Opcions cercador</h2>");
    print("<p>En aquesta secció es gestionen els paràmetres del plugin de cerca</p>");

    print("<form action='' method='post' id='cercador_opcions_form'>");
    print("<h3><label for='cercador_opcions_pages'>Cercar en les pàgines</label>");
    $checked ="";
    if(get_option('cercador_opcions_pages')=='true')
        $checked = "checked";

    print("<input type='checkbox' name='cerca_pagines' id='cerca_pagines' value='true' ".$checked."/></h3>");

    print("<p><input type='submit' name='submit' value='Desa'/></p>");
    wp_nonce_field('cercador_opcions_update');
    print("</form>");
    print("</div>");
}


/*
 * afegim la opció al menú indicant la funció amb la interficie a cridar
 */

function cercador_plugin_menu(){
    add_options_page('Opcions cercador', 'Opcions cercador', 'manage_options','cercador_plugin','cercador_opcions');
}

/*
 * registrem l'acció per a que es cridi
 */
add_action('admin_menu', 'cercador_plugin_menu');
```

I el resultat hauria de ser el següent:



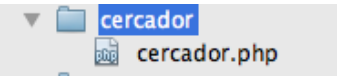
A partir del wp 2.7 hi ha una opció més senzilla de fer el mateix, i és utilitzant les funcions register_setting i settings_fields, que ja ens gestionen automàticament els “nonces” i els missatges de guardat. El primer mètode que s’ha explicat és vàlid per a les versions anteriors, tot i que ja és un mètode “deprecated” de fer-ho.

El codi amb aquesta segona opció seria el següent.

```php
function cercador_inicial(){
    register_setting('opcions_cercador','cerca_pagines');
}

add_action('admin_init','cercador_inicial');

function cercador_opcions(){

    print("<div class='wrap'>");
    screen_icon();
    print("<h2>Opcions cercador</h2>");
    print("<p>En aquesta secció es gestionen els paràmetres del plugin de cerca</p>");

    print("<form action='options.php' method='post' id='cercador_opcions_form'>");
    settings_fields('opcions_cercador');
    print("<h3><label for='cerca_pagines'>Cercar en les pàgines</label>");
    $checked ="";
    if(get_option('cerca_pagines')=='true')
        $checked = "checked";

    print("<input type='checkbox' name='cerca_pagines' id='cerca_pagines' value='true' ".$checked."/></h3>");
    print("<p><input type='submit' name='submit' value='Desa'/></p>");

    print("</form>");
    print("</div>");
}
```

Acabem de veure com crear una opció dins del menú d'opcions del wp, però potser que necessitem afegir una opció dins d'un altre menú, o fins hi tot crear el nostre propi menú per al nostre plugin. Anem a veure com fer-ho.

Per començar crearem una carpeta "cercador" dins del directori de plugins per tal de tenir més ben organitzat el desenvolupament. Un cop fet, cal assegurar-se que el plugin s'ha tornat a activar, o el wp seguirà buscant l'anterior i no el trobarà.



Si anem al nostre codi, recordem que per afegir l'opció en el menú d'opcions, utilitzàvem la funció add_options_page, doncs bé, igual que aquesta funció n'existeixen moltes d'altres. Existeix el add_post_page, add_media_page, add_themes_page, add_dashboard_page ..., i per tant podem moure l'opció al menú que vulguem.

Si volem crear el nostre propi menú, utilitzarem la funció add_menu_page, que a part dels mateixos paràmetres que teníem abans permet especificar una icona especifica i l'ordre dins del menú. Per afegir un submenú podem crear a la funció add_submenu_page, tal i com podem veure a la imatge següent.

```php
function cercador_plugin_menu(){
    add_menu_page('Opcions cercador', 'Opcions cercador', 'manage_options',
                  'cercador_plugin','cercador_opcions','/wp-content/plugins/cercador/icona.png');
    add_submenu_page('cercador_plugin', 'Configuració', 'Configuració cercador', 'manage_options', 'cercador_plugin');
}

/*
 * registrem l'acció per a que es cridi
 */
add_action('admin_menu', 'cercador_plugin_menu');
```

Des de la versió 2.7 el wp ens dona l'opció de personalitzar el nostre escriptori (dashboard), el que vol dir que podem afegir els nostres propis widgets en l'administració del wp. Aquest tipus de widget són pràcticament iguals que els que acabem de veure, però preparats per al dashboard de l'administració. Per a crear-ne un, només hem de crear la funció que ens generi el contingut del widget, la funció que ens afegeix el widget al dashboard i el hook per tal de que aquesta s'executi al instal·lar el dashboard. El codi seria similar al següent.

```php
function prova_dashboard_wg(){
    print("<h2>Aquesta es una proba de widget per al escritori</h2>");
    print("<p>Podem afegir el contingut que volguem.</p>");
    print("<p>Amb enllaços <a href='http://www.uoc.edu'>inclosos</a></p>");
}

function registra_dashboard_wg(){
    wp_add_dashboard_widget('prova_dashboard_wg', 'Prova de widget dashboard', 'prova_dashboard_wg');
}

add_action('wp_dashboard_setup','registra_dashboard_wg');
```

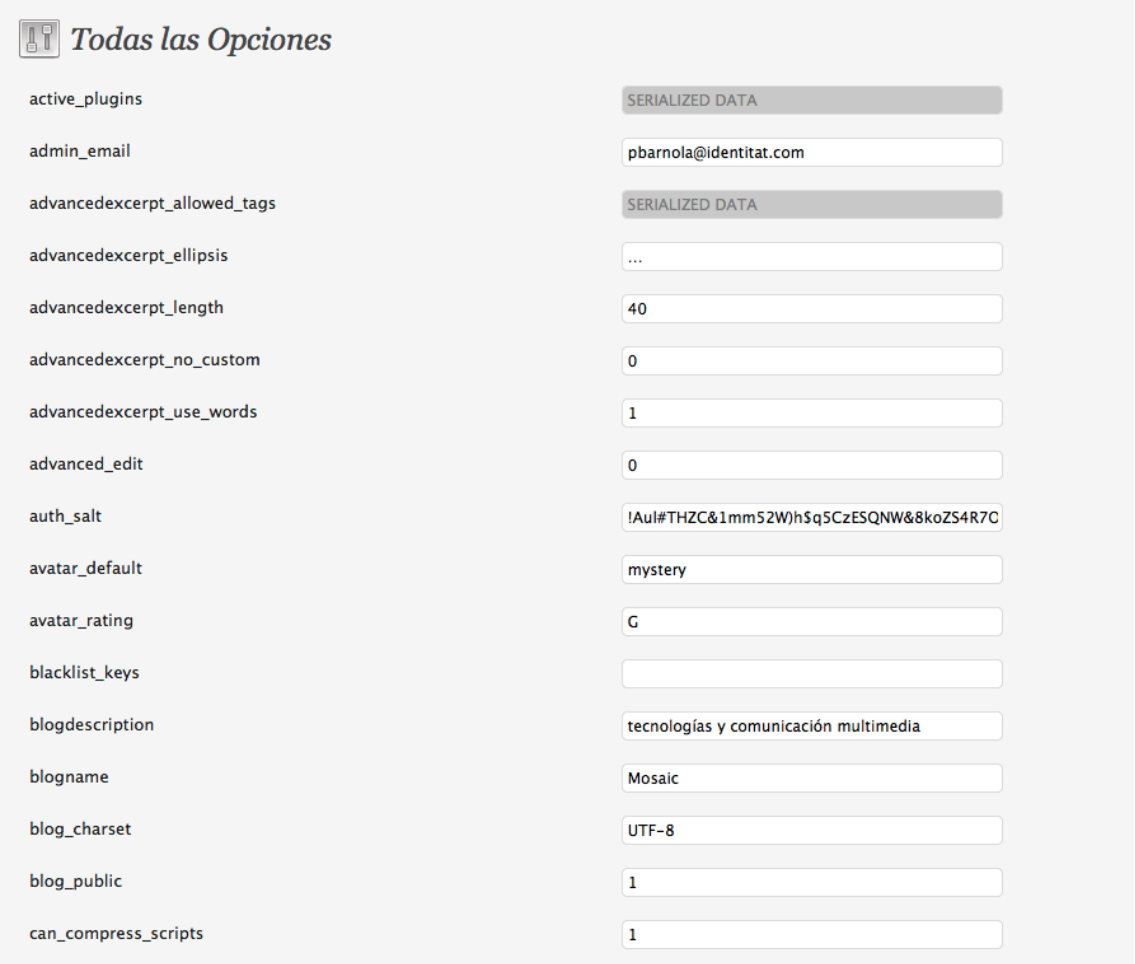
i el resultat aquest

La base de dades d'opcions del wp té un nombre d'opcions registrades.

Algunes d'elles venen per defecte i poden ser útils en els nostres desenvolupaments, i a part també hi voldrem guardar les nostres pròpies opcions tal i com em vist abans.

A continuació veurem com ens integrem amb aquestes opcions.

Per visualitzar totes les opcions disponibles hem d'accedir a la següent url (http://localhost/wp-admin/options.php) de forma directa, ja que no hi ha cap enllaç en l'administració que ens hi porti.

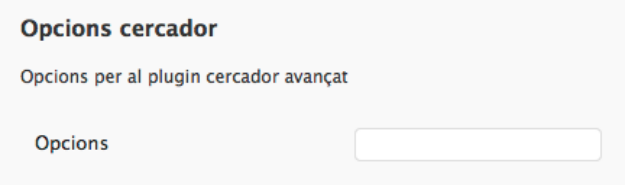


Per accedir a elles ho podem fer amb el get_option, al igual que ho hem fet en el nostre fitxer d'opcions.

Si volem afegir les nostres pròpies opcions ho faríem amb el següent.

```
function cercador_opcions_camp(){
    print("<input type='text' name='cercador_parametre' id='cercador_parametre' value='".get_option('cercador_parametre')."'/>");
}

function cercador_opcions_seccio(){
    print("<p>Opcions per al plugin cercador avançat</p>");
}

function cercador_plugin_menu(){
    add_settings_section('cercador', 'Opcions cercador', 'cercador_opcions_seccio', 'general');
    add_settings_field('cercador_parametre','Opcions','cercador_opcions_camp','general','cercador');
}

add_action('admin_menu', 'cercador_plugin_menu');
```

I obtindríem el resultat similar a aquest en la secció d'opcions.



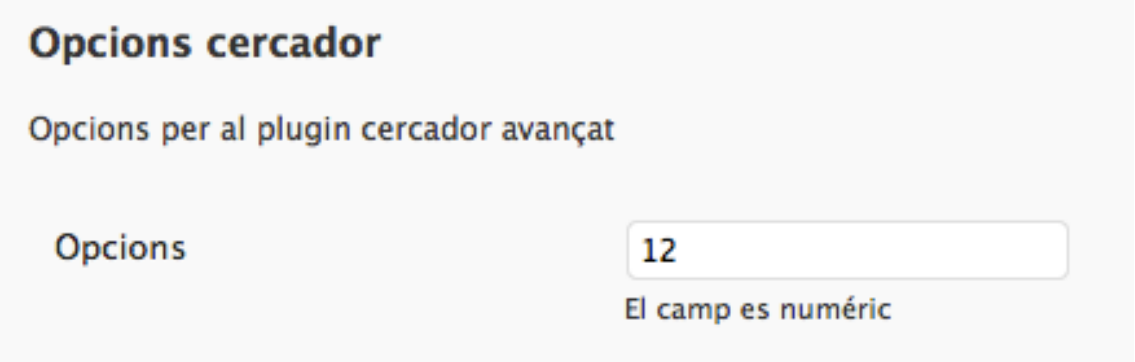
AJAX i jquery

El wp ens permet fer crides directes a funcions AJAX i porta incorporades per defecte diferents llibreries com jquery per millorar la interfície d'usuari.

Les crides AJAX permeten executar processos de servidor sense tenir que recarregar la pàgina, millorant considerablement l'experiència d'usuari.

Per a que veure el funcionament farem un petit exemple que simularà la comprovació numèrica del camp opcions del cercador que hem afegit a les opcions de configuració en la secció anterior. El que volem aconseguir es que un cop introduïm informació en el nostre camp, ens digui si el valor introduït és o no numèric.

El resultat que volem aconseguir seria el següent:



Per a fer-ho, primer crearem la part de servidor, que l'únic que haurà de fer és recollir el camp i comprovar que el seu valor sigui numèric.

Crearem la funció cercador_validacio() , que serà la que donat un valor ens traurà per pantalla si aquest és o no numèric. La registrarem com a crida ajax amb el add_action, tal i com podem veure en l'exemple.

```php
function cercador_opcions_camp(){
    print("<input type='text' name='cercador_parametre' id='cercador_parametre' value='".get_option('cercador_parametre')."'/>");
    print("<div id='cercaInfo' align='left'></div>");

}

function cercador_opcions_seccio(){
    print("<p>Opcions per al plugin cercador avançat</p>");
}

function cercador_plugin_menu(){
    add_settings_section('cercador', 'Opcions cercador', 'cercador_opcions_seccio', 'general');
    add_settings_field('cercador_parametre','Opcions','cercador_opcions_camp','general','cercador');
}

add_action('admin_menu', 'cercador_plugin_menu');

function cercador_validacio(){
    $cerca = isset($_POST['cercador_parametre']) ? $_POST['cercador_parametre'] : null;
    $msg = "no_numero";
    if($cerca){
        if(is_numeric($cerca)) $msg="numero";
    }
    echo $msg;
    die();

}

add_action('wp_ajax_cercador_validacio','cercador_validacio');

add_action('admin_print_scripts-options-general.php', 'cercador_validacio_script');


function cercador_validacio_script(){
    wp_enqueue_script('cercador', path_join(WP_PLUGIN_URL, basename(dirname(__FILE__))).'/cercador.js', array('jquery'));
}

```

El següent pas serà crear un fitxer javascript que serà el que un cop sortim del camp del nostre formulari faci la crida a la funció corresponent i escrigui en la capa “cercaInfo”, que hem creat expressament, la informació del resultat.

Finalment, només ens caldrà registrar el fitxer javascript i la llibreria jquery.

cercador.php | cercador.js

```
jQuery(document).ready(function($){
    $("input[name='cercador_parametre']").blur(function(){
        $.ajax({
            type: "POST",
            data: "cercador_parametre="+$(this).attr("value") + "&action=cercador_validacio",
            url: ajaxurl,
            beforeSend: function(){
                $("#cercaInfo").html("comprovant numéric...");
            },
            success: function(data){
                if(data=="numero"){
                    $("#cercaInfo").html("El camp es numéric");
                }else{
                    $("#cercaInfo").html("El camp no es numéric");
                }
            }
        })
    })
})
```

Per a més informació sobre l’us de ajax i llibreries en javascript integrades en el wp, podeu consultar la referència a http://codex.wordpress.org

Ús de la base de dades de wordpress

El wp funciona amb la base de dades mysql. La manera directa per accedir a la base de dades, és utilitzant la classe wp_db que es basa en sql, i per tant requereix coneixement bàsic sobre quèries en sql. De tota manera, tot i que es pot accedir directament a la base de dades, normalment no és el millor mètode de fer-ho. Doncs existeixen varies funcions ja preparades que ens permeten recuperar diferent informació de la base de dades de wp.

Anem doncs a fer un petit exemple d’accés directe a la bd del wp. Aprofitarem l’arxiu que hem creat anteriorment per a fer el nostre widget d’escriptori.

Extraurem contingut de la base de dades per utilitzar-lo en el nostre contingut de manera que podrem veure les diferents maneres de recollir informació de ña base de dades de manera directa. Per fer-ho utilitzarem la variable global $wpdb que és la que conté la informació sobre la base de dades del wp per a fer diferents quèries per extreure informació.

El nostre codi font seria el següent:

```php
function prova_dashboard_wg(){
    global $wpdb;
    global $current_user;
    ?>
    <h2>Informació base de dades</h2>
    <p><strong>Ultima Query:</strong> <?php echo $wpdb->last_query; ?></p>
    <p><strong>Ultim Error:</strong> <?php echo $wpdb->last_error; ?></p>
    <p><strong>Número usuaris:</strong> <?php echo $wpdb->query('SELECT ID FROM wp_users'); ?></p>
    <p><strong>Ultim Post:</strong> <?php echo $wpdb->get_var('SELECT post_title FROM '.$wpdb->posts.
                                        ' WHERE post_author = '.$current_user->ID); ?></p>
    <p><strong>Emails: </strong>
        <pre>
            <?php $user_emails =  $wpdb->get_col('SELECT user_email FROM ' . $wpdb->users);
                echo var_dump($user_emails); ?>
        </pre></p>
        <p><strong>Informació usuaris: </strong>
        <pre>
            <?php $my_user_data = $wpdb->get_row('SELECT * FROM ' . $wpdb->users . ' WHERE ID = ' . $current_user->ID, ARRAY_N);
                echo var_dump($my_user_data); ?>
        </pre></p>
        <p><strong>Tots els posts: </strong>
        <pre>
            <?php $terms = $wpdb->get_results('SELECT * FROM ' . $wpdb->terms, ARRAY_N);
                echo var_dump($terms); ?>
        </pre></p>

    <?php
    //per veure el contingut de l'objecte wpdb podem fer el següent
    print('<pre>');
    print_r($wpdb);
    print('</pre>');
}

function registra_dashboard_wg(){
    wp_add_dashboard_widget('prova_dashboard_wg', 'Prova de widget dashboard', 'prova_dashboard_wg');
}

add_action('wp_dashboard_setup','registra_dashboard_wg');
```

i el resultat el podrem veure en el nostre dashboard

Prova de widget dashboard

Informació base de dades

Ultima Query: SELECT meta_value FROM wp_usermeta WHERE user_id = 2 AND meta_key = 'default_password_nag'

Ultim Error:

Número usuaris: 2

Ultim Post: ¡Hola mundo!

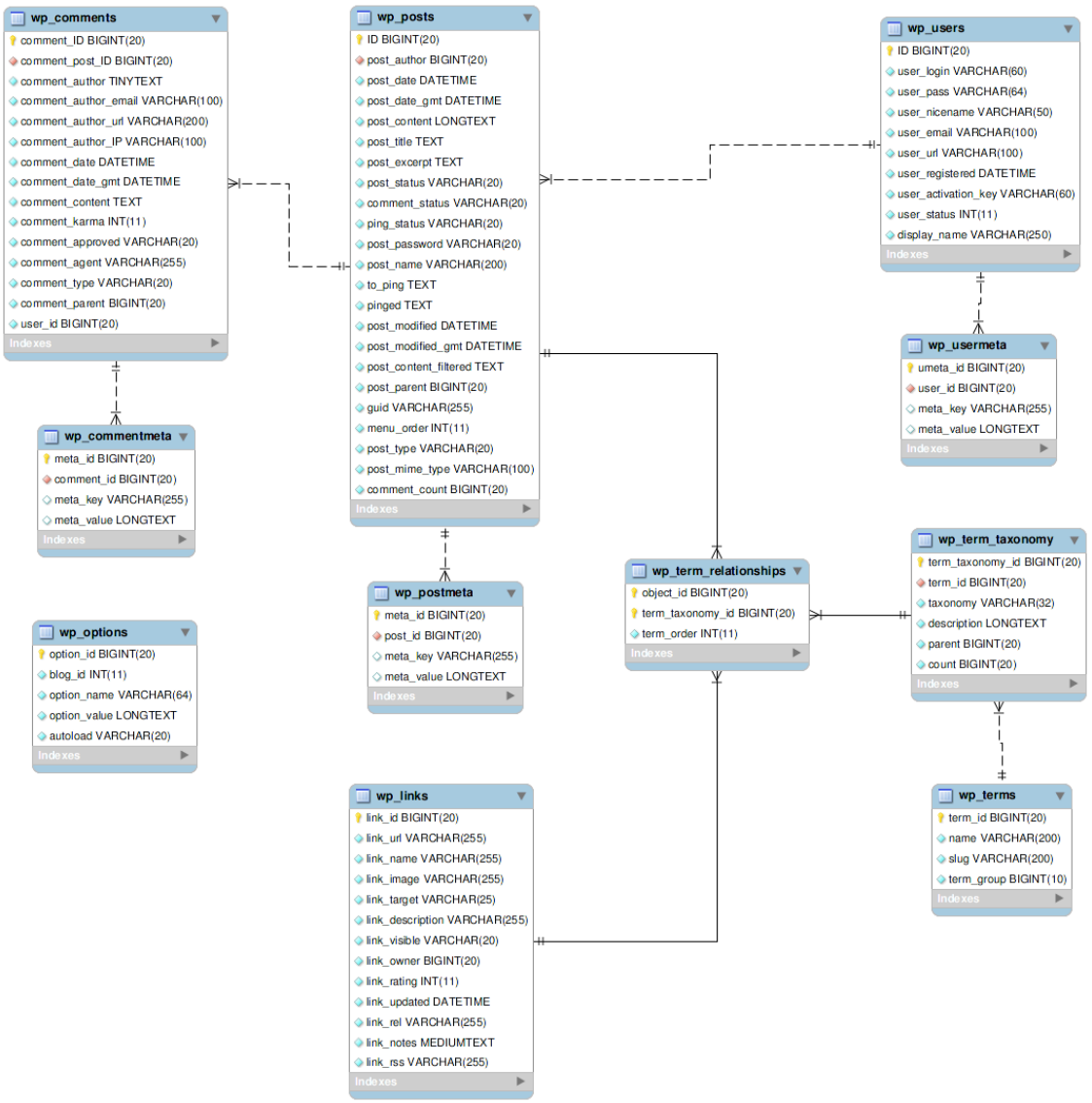
```
wpdb Object
(
    [show_errors] =>
    [suppress_errors] =>
    [last_error] =>
    [num_queries] => 20
    [last_query] => SELECT post_title FROM wp_posts WHERE post_au
    [col_info] => Array
        (
            [0] => stdClass Object
                (
                    [name] => post_title
                    [table] => wp_posts
                    [def] =>
                    [max_length] => 106
                    [not_null] => 1
                    [primary_key] => 0
                    [multiple_key] => 0
                    [unique_key] => 0
                    [numeric] => 0
                    [blob] => 1
                    [type] => blob
                    [unsigned] => 0
                    [zerofill] => 0
                )

        )
```

L'esquema de bdd

Com ja hem dit, el worpress esta basat amb mysql, i en la base de dades es guarda tota la informació del wp: les pàgines, els posts, les categories, els tags ... , qualsevol element del wp.

És molt important conèixer la estructura per tal de poder desenvolupar un plugin per a wp amb accés directe a la bd. Cal tenir en compte que la estructura de la base de dades del wp evoluciona constantment i varia en funció de les versions, per tant sempre que puguem es millor utilitzar funcions d'accés a la bdd enlloc d'accedir-hi directament.

Tota la informació i estructura de la base de dades, la podem trobar explicada a http://codex.wordpress.org/Database_Description



Crear noves taules a la bd

La raó perquè estem revisant l'estructura i la manera d'accedir a la base de dades de wp, es perquè moltes vegades en els nostres plugins necessitarem tenir les nostres pròpies estructures de dades. Per tal de poder fer això el wp ens dona una sèrie d'eines per a que puguem crear la nostra pròpia estructura.

Com sempre, haurem de fer una funció que ens creí les nostres taules mitjançant sql.

A continuació podem veure la funció que crearà una taula quan s'activi el nostre plugin, utilitzant la funció 'register_activation_hook' que ja vam veure a l'inici.

```
162    function crearTaules(){
163        global $wpdb;
164
165        $nom_taula = $wpdb->prefix."taula_cerca";
166
167    //provem si la taula existeix, si existeix no la crearem
168        if($wpdb->get_var('SHOW TABLES LIKE '.$nom_taula)!=$nom_taula){
169            $sql = 'CREATE TABLE '.$nom_taula. '(
170                id INTEGER(10) UNSIGNED AUTO_INCREMENT,
171                data TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
172                cerca VARCHAR(255),
173                PRIMARY KEY (id))';
174
175            require_once(ABSPATH.'wp-admin/includes/upgrade.php');
176
177            dbDelta($sql);
178
179            add_option('crear_taules_version','1.0');
180        }
181
182    }
183
184    register_activation_hook(__FILE__, 'crearTaules');
185
```

El resultat hauria de ser el següent.

	Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
☐	id	int(10)		UNSIGNED	No	*None*	auto_increment
☐	data	timestamp			No	CURRENT_TIMESTAMP	
☐	cerca	varchar(255)	utf8_general_ci		Sí	*NULL*	

Per a realitzar una inserció, utilitzarem la funció $wpdb->insert, que rep 3 paràmetres: el primer, el nom de la taula, el segon, un array associatiu amb el contingut de la inserció per a cada camp, i el tercer el format del valor que s'està insertant per evitar injeccions sql.

```php
function insertarInfo()
{
    global $wpdb;
    $nom_taula = $wpdb->prefix."taula_cerca";

    $wpdb->insert($nom_taula,array('cerca'=>"nova cerca"),array('%s'));
    $wpdb->insert_id;
}

add_action('wp_footer', 'insertarInfo');
```

Per a recuperar l'id de la inserció podem accedir a la propietat "insert_id" que ens facilita el wp. Per a realitzar un Update es realitzaria de la mateixa manera, però rebent un paràmetre addicional per la clàusula de where.

			id	data	cerca
☐	✎	✕	1	2011-05-29 13:34:20	nova cerca
☐	✎	✕	2	2011-05-29 13:34:29	nova cerca

Podeu trobar més informació en la web de referencia de wp.

En el capítol de seguretat parlarem més a fons dels mètodes per evitar problemes de seguretat en l'accés a la base de dades.

Posts, pàgines (pages) i el bucle (Loop)

Quan es treballa amb el wp, en algun o altre moment haurem de treballar amb "posts". No hem d'oblidar que el wp, com a base, és un gestor de contingut per a "blogs".

El wp te un objecte especial per treballar amb els "posts" i les pàgines (pages) . Aquesta estructura clau per a l'arquitectura del wp és el "bulce" (Loop) .

El "loop" conté la informació relacionada amb posts i pàgines de cada secció. Per exemple, en la pàgina principal conté els 10 últims posts, o en un post o pàgina concreta, conté tota la informació del post/pàgina.

Anem doncs a crear un petit widget que ens mostri la informació del loop en cada secció. Per a recórrer el "loop" farem servir la funció "have_posts()" que en la primera crida retornarà el contingut del primer post o si no n'hi ha, no retornarà res.

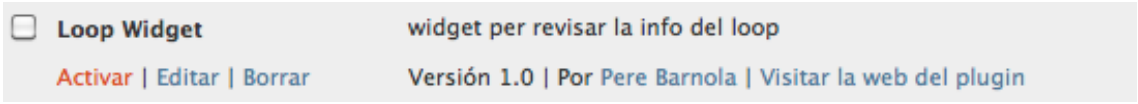
Iterarem sobre el loop i accedirem a la informació del post mitjançant els template tags "the_post()" que ens retorna el post actual dins del loop, "the_permalink()" amb l'enllaç al post, "the_title()" amb el títol del post i "comments_number()" que ens retorna el número de comentaris.

El nostre codi seria el següent:

loop_widget.php

```php
<?php
/*
 * Plugin Name: Loop Widget
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: widget per revisar la info del loop
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)
 */

function loop_widget()
{
        ?>
        <h3>Posts de la pàgina:</h3>
        <?php if ( have_posts()) :
                        while ( have_posts()) :
                                the_post();
        ?>
        <div>
        <a href="<?php echo the_permalink(); ?>"
                title="<?php echo the_title(); ?>"><?php echo the_title(); ?></a>
                (<?php echo comments_number(); ?>)
        </div>
        <?php   endwhile;
                  endif;
        ?>
        <?php
}

function loop_widget_init() {
        register_sidebar_widget('Els posts', 'loop_widget');
}
add_action('widgets_init','loop_widget_init');
```

Un cop guardat, l'activem



i arrosseguem el widget a la barra lateral

El resultat hauria de ser quelcom com el següent, recuperant la informació del loop en cada secció.

El objecte/estructura "loop" ens dona als desenvolupadors l'opció d'accedir a la informació de pàgines i posts de cada secció.

Utilitzant el WP_Query

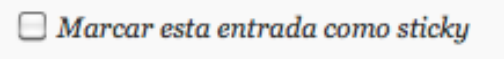
Com acabem de veure cada pàgina de wp, té el seu propi "loop" que conté la informació rellevant per a cada pàgina.

Pot ser que en algun moment necessitem accedir al nostre propi "loop", per accedir al mateix tipus d'informació però recollir-la del nostre propi objecte i no del loop per defecte. Per a fer-ho utilitzarem el mètode WP_Query()

Modificarem el exemple anterior i crearem el nostre loop amb la funció WP_Query(), de manera que ara, quan carreguem una pàgina que no tingui el seu propi loop, veurem la informació que tinguem carregada al nostre objecte.

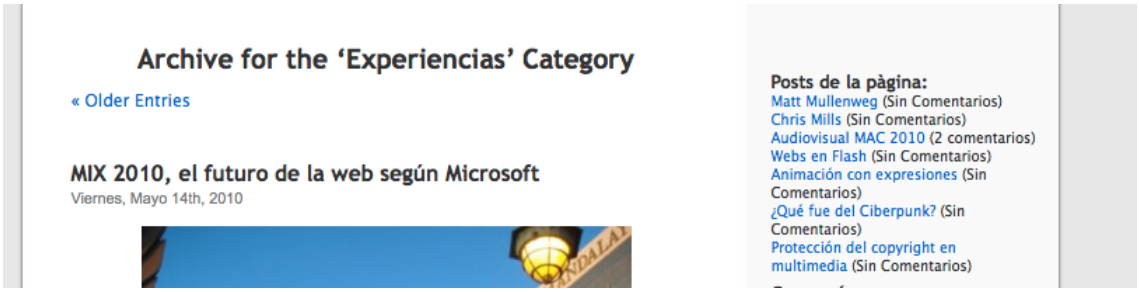
```php
<?php
/*
 * Plugin Name: Loop Widget
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: widget per revisar la info del loop
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)
 */

function loop_widget()
{
        $el_nostre = new WP_Query();
        $el_nostre->get_posts();
        ?>
        <h3>Posts de la pàgina:</h3>
        <?php if ( $el_nostre->have_posts()) :
                        while ( $el_nostre->have_posts()) :
                                $el_nostre->the_post();
        ?>
        <div>
        <a href="<?php echo the_permalink(); ?>"
                title="<?php echo the_title(); ?>"><?php echo the_title(); ?></a>
                (<?php echo comments_number(); ?>)
        </div>
        <?php   endwhile;
                  endif;
        ?>
        <?php
}

function loop_widget_init() {
        register_sidebar_widget('Els posts', 'loop_widget');
}
add_action('widgets_init','loop_widget_init');
```

La funció WP_Query, retorna bàsicament la informació que tenim a la nostra pàgina principal, però existeixen maneres de filtrar aquesta informació en funció del títol, de l'autor ... Això és fa a través de la funció query_posts() en la que podem controlar els posts a mostrar en el loop. A la web http://codex.wordpress.org/Function_Refernce/query_posts podem veure els diferents paràmetres que pot rebre. Si recuperem el nostre exemple d'abans, ara li afegirem filtres recuperant la informació amb el mètode query. Limitarem el nombre de posts a 5 posts, ordenarem de més gran a més petit, en funció del número de comentaris que tingui aquell post, i per últim només treurem els posts marcats com a "sticky".



El codi seria el següent, i en el resultat veiem com realment només apareixen els posts que compleixen les condicions.

```php
<?php
/*
 * Plugin Name: Loop Widget
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: widget per revisar la info del loop
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)
 */

function loop_widget()
{
        $el_nostre = new WP_Query();
        $el_nostre->query(array('posts_per_page'=>5,
                                'orderby'=>'comment_count',
                                'order'=>'DESC',
                                'post_in'=>  get_option('sticky_posts')));
        ?>
        <h3>Posts de la pàgina:</h3>
        <?php if ( $el_nostre->have_posts()) :
                        while ( $el_nostre->have_posts()) :
                                $el_nostre->the_post();
        ?>
        <div>
        <a href="<?php echo the_permalink(); ?>"
                title="<?php echo the_title(); ?>"><?php echo the_title(); ?></a>
                (<?php echo comments_number(); ?>)
        </div>
        <?php   endwhile;
                  endif;
        ?>
        <?php
}

function loop_widget_init() {
        register_sidebar_widget('Els posts', 'loop_widget');
}
add_action('widgets_init','loop_widget_init');
```



Anem a veure un exemple de com s'utilitzaria a aquesta informació mitjantçant ajax sense tenir que recarregar el document. El que farem es visualitzar una previsualització del contingut dels posts que mostrem en el nostre widget anterior.

Per a fer-ho primer crearem la funció de servidor que quan la cridem ens retornarà informació necessària i ens retornarà la informació del post especificat. La funció que utilitzem en l'exemple és "loop_posts_comentaris", un cop feta la registrarem per a que pugui ser cridada via AJAX tal i com ja vam veure en seccions anteriors.

```php
<?php
/*
 * Plugin Name: Loop Widget
 * Plugin URI: http://www.identitat.com/cercador-wp
 * Description: widget per revisar la info del loop
 * Author: Pere Barnola
 * Version: 1.0
 * Author URI: http://www.identitat.com
 * Copyright 2010  Pere Barnola  (email : pbarnola@identitat.com)
 */

function loop_widget()
{
        $el_nostre = new WP_Query();
        $el_nostre->query(array('posts_per_page'=>5,
                                'orderby'=>'comment_count',
                                'order'=>'DESC',
                                'post_in'=>  get_option('sticky_posts')));
        ?>
        <h3>Posts de la pàgina:</h3>
        <?php if ( $el_nostre->have_posts()) :
                        while ( $el_nostre->have_posts()) :
                                $el_nostre->the_post();
        ?>
        <div class="loop_posts">
        <a href="<?php echo the_permalink(); ?>"
                id="<?php echo the_id(); ?>" title="<?php echo the_title(); ?>" class="comentari_link"><?php echo the_title(); ?></a>
                (<?php echo comments_number(); ?>)
        </div>
        <?php   endwhile;
                  endif;
        ?>
        <?php
}

function loop_posts_comentaris(){
    $post_id = isset($_POST['post_id']) ? $_POST['post_id'] : 0;

    if($post_id>0){
        $post = get_post($post_id);
        print("<div id='post'>".$post->post_content."</div>");
    }
    die();
}

add_action('wp_ajax_nopriv_loop_posts','loop_posts_comentaris');


/* afegim el js */

function loop_posts_get_scripts(){
    wp_enqueue_script('loop_widget',path_join(WP_PLUGIN_URL, basename(dirname(__FILE__))."/loop_widget.js"), array("jquery"));
}
add_action("wp_print_scripts","loop_posts_get_scripts");

function loop_widget_init() {
        register_sidebar_widget('Els posts', 'loop_widget');
}
add_action('widgets_init','loop_widget_init');
```

Per altra banda haurem de crear un javascript que ens farà la crida des de client en el moment que fa el mouseover. Per a fer-ho utilitzarem jQuery per fer la crida AJAX i controlar el mouse over sobre els elements. El jQuery necessita tenir identificats els elements del html mitjantçant id o classes per tal de poder accedir-hi de forma àgil, i per tant els afegirem en el codi.

```javascript
jQuery(document).ready(
    function($){
        $("div.loop_posts").mouseover(
            function(){
                var div = $(this);

                $.post('wp-admin/admin-ajax.php',
                {
                action: "loop_posts",
                post_id: $(this).find("a").attr("id")
                },
                function (data){
                    div.append($(data));
                }
                );

                return false;
            }
        );

        $('div.loop_posts').mouseout(
            function(){
                $('#post').remove();
            }
        );
    }
);
```

Com a darrer pas i com ja vam veure en el tema en que parlàvem de ajax, hem d'afegir el nostre js amb la funció wp_enqueue_script.

Si ho visualitzem un cop salvat, veurem el resultat i quan fem mouseover s'hauria de carregar la informació sense tenir que recarregar la pàgina.

Cercle de vida de un plugin

Doncs ja sabem com desenvolupar plugins, però ara que?, imaginem que ja tenim el nostre plugin acabat, és el moment de enviar-lo a http://codex.wordpress.org/Plugin_Submission_and_Promotion i publicar-lo per a que altra gent el pugui utilitzar. Tots els plugins que trobem en les cerques del nostre admin, estan inclosos en la web de wordpress. Per a poder tenir el nostre plugin publicat a la web de wordpress, aquest ha de seguir certes característiques. Podreu trobar tota la informació sobre el procés de publicació de un plugin a la web de wordpress.org en el següent enllaç http://wordpress.org/extend/plugins/about/#readme

Per una banda te que ser compatible amb la llicència GNU, que bàsicament vol dir que altra gent el pot utilitzar, te que ser legal, és a dir que no es pot utilitzar codi d'altres parts, i ha de ser compatible amb svn per poder pujar el codi en el seu repositori.

Per a fer l'alta s'ha de tenir un compte d'usuari i emplenar el formulari d'alta http://wordpress.org/extend/plugins/add

També necessitareu tenir un fitxer readme.txt amb les característiques principals del vostres plugin. Podeu trobar un fitxer readme.txt base en el següent enllaç http://wordpress.org/extend/plugins/about/readme.txt

Un cop publicat a la web de wordpress, també ho podreu publicar a altres webs com http://wp-plugins.net o http://wp-plugins.org o molts d'altres.

Creació de la funció de desinstal·lació

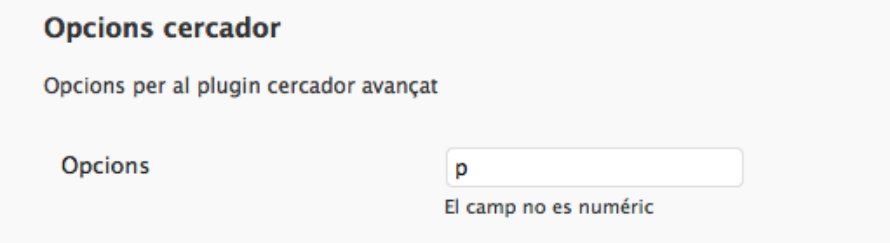
Un dels problemes de treballar en un entorn obert com és el wp, és que si utilitzes molts plugins, aquests toquen opcions i variables en la base de dades, i un cop vulguem deixar de utilitzar-los, al desintal·lar-los, ens podem trobar

que quedin pendents i tocades certes opcions a la nostra estructura i base de dades.

Per tant, una bona practica com a desenvolupadors de plugins, serà la de crear un script de desinstal·lació que elimini totes les opcions o estructures que haguem tocat i deixi la instal·lació igual que abans d'instal·lar el nostre plugin.

Existeixen un parell de maneres de fer això. La primera és registrar un hook de desinstal·lació al que especificarem una funció que es cridarà cada cop que es desinstal·li el nostre plugin, i la segona es utilitzar el fitxer uninstall.php.

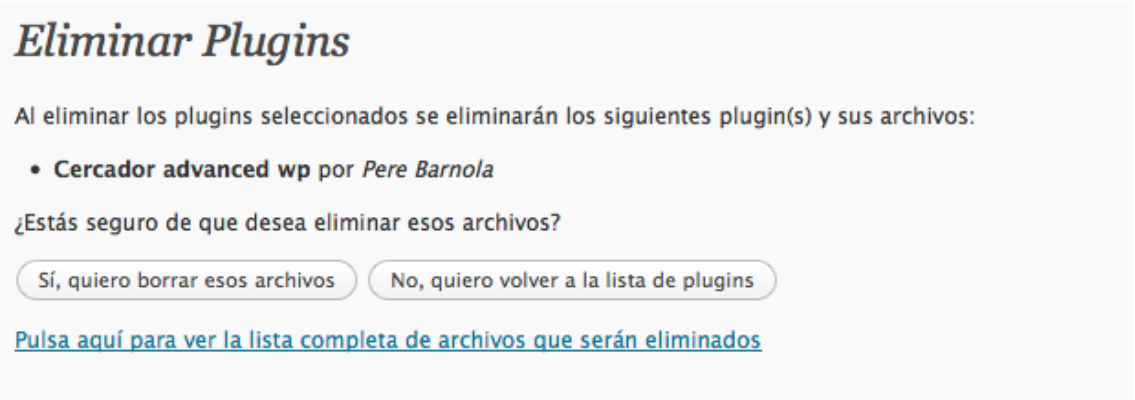
Si recuperem el nostre plugin d'exemple sobre el que hem estat treballant. "cercador.php". Recordem que havíem creat una opció en les opcions de configuració del nostre admin en el que simplement comprovàvem si el camp era numèric o no.



El que volem ara, és que en el moment de la desinstal·lació aquest camp s'elimini i la nostra configuració quedi tal i com estava abans d'haver-lo instal·lat. Per a fer-ho crearem una funció cercador_uninstall() en la que eliminarem aquesta opció. Per a eliminar una opció, simplement s'ha de cridar a la funció "delete_option" i passar-li el nom de la opció que volem eliminar. En el nostre cas "cercador_parametre". Després, com sempre, registrarem aquesta funció amb un hook, en concret, el register_uninstall_hook.

```
197
198     function cercador_uninstall(){
199         delete_option('cercador_parametre');
200     }
201
202     register_uninstall_hook(__FILE__,'cercador_uninstall');
203
```

Abans de continuar, feu una copia del plugin, guardant la carpeta a un altre lloc del disc, i desactivarem i eliminarem el nostre plugin.



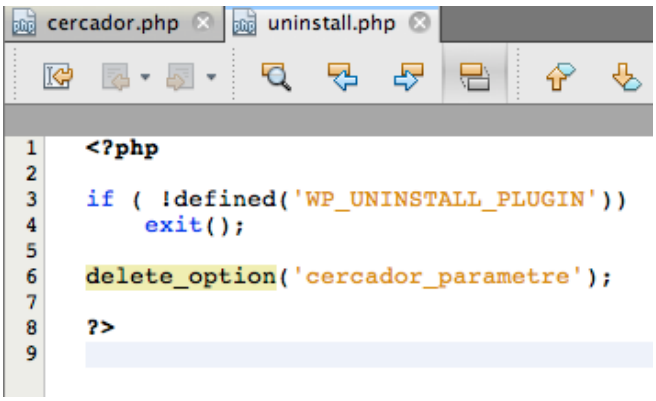
Un cop fet, si tornem a la pàgina d'opcions, veurem que la nostra opció haurà desaparegut tal i com volíem, i la nostra carpeta del plugin també.

Anem a fer una proba amb l'altra opció. Recuperem el nostre plugin del backup previ i el tornem a activar.

Anem a veure com faríem el mateix amb utilitzant el fitxer uninstall.php. La diferència bàsica, és que en el primer mètode, s'executa tot el fitxer per poder registrar el hook, i es probable que no vulguem que tot s'executi.

Per a fer-ho, primer esborrarem la part de desinstal·lació del nostre fitxer, i crearem un fitxer "uninstall.php" dins de la carpeta del nostre plugin, on posarem el codi d'esborrat d'opció "delete_option('cercador_parametre');".

Per evitar que aquest fitxer no s'executi de forma involuntària i esborri el que ha d'esborrar sense que l'usuari ho vulgui, només s'executarà el codi si s'ha cridat a l'opció de desinstal·lació. Això s'aconsegueix mitjançant la variable WP_UNISTALL_PLUGIN. Per tant el nostre codi quedaria de la següent manera.



Versions

El wordpress s'està constantment renovant, i això significa que sovint tenim noves millores i versions. Com a desenvolupadors, hem de tenir en compte la possibilitat de que el nostre plugin pugui córrer en diferents versions. Per a que

això sigui possible hem de seguir algunes pràctiques que veurem a continuació.

A la web http://codex.wordpress.org/WordPress_Versions , podem trobar informació sobre totes les versions que han sortit, quan han sortit, i quines diferencies han aportat sobre la versió anterior.

El primer punt a tenir en compte, és la no utilització de funcions antigues o "deprecated". A http://codex.wordpress.org/Category_Deprecated_Functions podem trobar tota la informació sobre aquestes funcions que ja no existeixen o no funcionen en les darreres versions. La manera d'assegurar que una funció existeixi en la versió que estem corrent és utilitzant la funció function_exists, de manera que si la funció que volem cridar no existeix, no la cridarem i evitarem l'error.

```
if(function_exists('register_uninstall_hook')){
   register_uninstall_hook(__FILE__,'cercador_uninstall');
}
```

Seguretat

El WP com qualsevol altra aplicació online esta exposat a atacs. És públic i obert a tothom. Per tant hem de tenir especial cura amb la seguretat i no desenvolupar plugins que puguin fer vulnerables a la gent que els instal·li.

Una de les coses bones de wp és que conté ja de per si moltes mesures de seguretat incorporades i per tant no ens n'hem de preocupar en excés, però hi ha algunes coses que cal tenir en compte.

Un dels moments més crítics es quan es recull informació provinent de l'usuari, això es pot fer a traves dels mètodes POST o GET, i es possible que algú pugui intentar utilitzar-los per afegir codi maliciós en el contingut. Una de les tècniques més utilitzades, és el sql injection, que vol dir que intenten afegir codi sql en les dades per tal d'extreure informació privada de la base de dades.

Amb anterioritat ja hem vist un exemple on inseríem informació a la nostra bdd mitjançant el mètode $wpdb->insert, en el que especificàvem el tipus de contingut a inserir com a string i per tant evitava que algú pogués afegir sql en aquella inserció, i si ho fes simplement aquell sql seria inserta a la bdd com a string.

```
$wpdb->insert($nom_taula,array('cerca'=>"nova cerca"),array('%s'));
```

Anem a veure com ho faríem sense utilitzar la funció insert per a que tinguem una idea clara de fer-ho.

El primer que necessitarem fer és cridar al mètode prepare() de la classe wpdb que bàsicament prepararà un query segura per a després executar-la amb el mètode query. Aquesta manera és una manera segura d'executar quèries amb

informació dinàmica provinent de la part pública i és com ho hauríem de fer per evitar-nos problemes de seguretat.

```
$query_segura = $wpdb->prepare('INSERT INTO '.$nom_taula.' SET user_agent = %1s',$_SERVER['HTTP_USER_AGENT']);
$wpdb->query($query_segura);
```

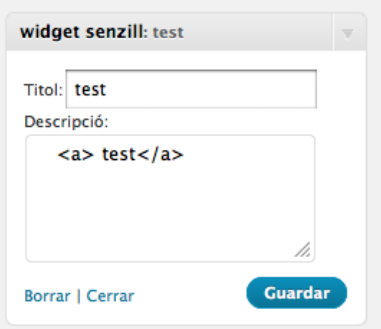
Una altra bona practica a tenir en compte, és la d'analitzar la informació que ens arriba. Hi ha un altre tipus d'atac comú que s'anomena "cross site attack" i la manera d'evitar-lo és tractant/netejant la informació que ens arriba.

Amb anterioritat hem fet un widget que recollia informació i la inseria a la base de dades. Al ser informació externa, per anar bé l'hauríem d'haver "netejat" abans de inserir-la a la nostra bdd. Vam utilitzar la classe WP_Widget per a fer-lo, i cridàvem als mètodes widget, update i form. Doncs si recuperem aquell exemple, i implementem el mètode Update, abans d'actualitzar la informació la "netejarem" per no tenir problemes amb la base de dades i els atacs.

El primer que farem és utilitzar per al títol la funció strip_tags que ens elimina qualsevol rastre de html, i per al cos utilitzarem la funció integrada dins del wp, que elimina els tags i a la que podem especificar quins tags estan permesos, passant-los mitjançant un array.

```
function update($new_instance,$old_instance)
{
  $instance = $old_instance;
  $instace['title']= strip_tags($new_instance['title']);
  $allowed_html = array('a'=>
                        array('href'=> array(),
                              'title'=>array()),
                       'br' => array(),
                       'strong' => array(),
                       'em' => array()
                    );
  $instance['body'] = wp_kses($new_instance['body'],$allowed_html);
  return $instance;

 }
```

per tant si ara provem de posar codi en html a dins dels camps veurem que s'elimina automàticament a excepció dels tags que em especificat que estaven permesos.



També podem utilitzar altres funcions com validate_file, esc_js, esc_url ,... per assegurar que tota la informació que ens arriba a la base de dades és informació correcta i permesa.

Internacionalitzar el plugin

El wp s'utilitza arreu del món. Hi ha gent que té blocs en francès, en suec, en anglès ..., i per tant a l'hora de fer el nostre plugin si volem que sigui fàcil d'utilitzar i entendible per gent amb altres idiomes, haurem de tenir cura dels idiomes i la traducció del nostre plugin. El WP te un sistema per a gestionar-ho tot plegat, i podem trobar més informació de com fer-ho a http://codex.wordpress.org/Translating_WordPress on ens explica informació general, o a http://codex.wordpress.org/I18n_for_WordPress_Developers on ens explica els detalls més tècnics de com fer-ho.

El sistema es basa en la funció php Gettext. L'únic que em de fer els desenvolupadors, és marca els punts en el nostre codi, on existeixin etiquetes que puguin tenir que ser traduïdes. Per fer-ho utilitzarem la funció __("text traduïble"). Això no farà res a no ser que creem els dominis de text o els fitxer POT ("Portable Object Translation") necessaris. Aquests fitxers, són els que contindran la informació del que s'ha de traduir i on s'ha de traduir quan tinguem diferents idiomes. La idea és que algú que vingui de Xina ho vegi amb Xinés i algú que vingui d'estats units amb angles.

La funció __() només la utilitzarem quan aquella traducció la vulguem assignar a una variable, però per altra banda utilitzarem la funció _e() si volem treure aquella informació per pantalla. Es a dir simplement marquem el text que s'hauria de traduir si algú amb un altre idioma visualitzes el nostre plugin.

El següent pas, seria crear un fitxer .pot amb les traduccions necessàries. Hi ha diferents softwares que s'encarreguen d'això. No entrarem en com fer-ho, en tot cas ja sabem com preparar el nostre plugin per si necessitem que aquest es tradueixi en els idiomes necessaris.