Plug-ins WordPress, primers passos

César Córcoles

PID_00201051

UOC
Universitat Oberta de Catalunya

Índex

1. Introducció

A l'inici del web els llocs que es publicaven eren relativament senzills i es creaven "a mà", usant editors de text que es van adaptar a poc a poc al recentment creat llenguatge d'etiquetatge HTML. A poc a poc va augmentar la complexitat de molts d'aquests llocs web, fins que es va fer imprescindible algun procés de gestió automàtica del contingut. Així va néixer una nova categoria de programari: els **sistemes de gestió de continguts** (o CMS per les sigles en anglès, de *content management system*), que permeten que un col·lectiu administri els continguts i la presentació d'un lloc web amb una arquitectura d'informació sofisticada.

D'altra banda, i també des de l'inici del web, alguns usuaris van començar a editar "diaris web", que van anar evolucionant i van donar lloc als *weblogs* (posteriorment blogs), terme encunyat per Jorn Barger a la fi de 1997. No hi ha una definició estricta i consensuada del que és un blog, però el terme s'usa per a definir infinitat de llocs web que presenten una col·lecció d'"entrades" que s'agrupen en ordre cronològic invers en una o més pàgines web.

Amb la popularització del *blogging*, els últims anys del segle XX i primers del XXI, van anar apareixent sistemes de gestió de continguts molt lleugers dedicats exclusivament a la creació i manteniment de blogs. El 2003 un d'aquests sistemes era b2/*cafelog*, creat per Michel Valdrighi. Matt Mullenwegg i Mike Little van prendre el codi de b2 (de codi lliure) i van crear una variant sota el nom de *WordPress*.

Si bé la primera versió del WordPress va aparèixer el maig de 2003, amb número de versió 0.7, és a partir de 2004, amb l'aparició de la versió 1.2, que el programa comença a adquirir popularitat. L'evolució del WordPress, juntament amb un conjunt de fets aliens, ha comportat un augment continu d'aquesta popularitat, que ha vingut lligada a l'aparició d'una forta comunitat de desenvolupament de codi obert entorn de la plataforma.

Aquest procés de creixement ha portat a l'aparició gradual de noves funcionalitats en el programari, que fan que avui dia el WordPress mereixi el nom de *sistema de gestió de continguts*, amb millores en les característiques per a administrar plantilles per a la presentació de continguts, la interfície d'administració, l'editor de continguts, el rendiment del sistema, o la definició de fluxos de treball (que permeten, per exemple, que un editor pugui introduir continguts però que només un administrador pugui donar el vistiplau per a la seva publicació definitiva).

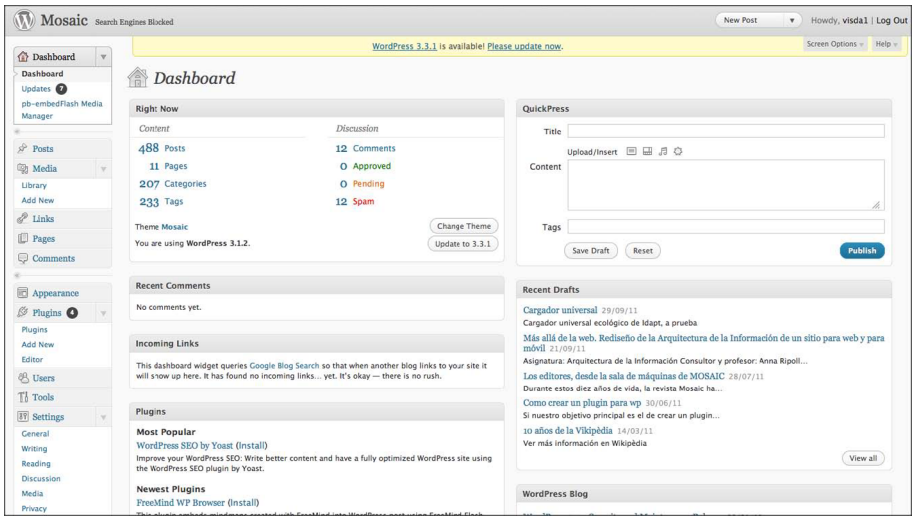
Avui dia hi ha un bon nombre de sistemes de gestió de continguts de propòsit general que, com el WordPress, són de codi lliure i funcionen sobre plataformes AMP: servidor web Apache, base de dades MySQL i llenguatge de programació "de servidor" PHP. Podem destacar Drupal, Joomla!, Mambo o TYPO3. Tots (i molts altres) es poden personalitzar per a crear el lloc web d'una revista com *Mosaic*.

Entre els factors que van influir en l'elecció del WordPress com a CMS per a *Mosaic* hi ha la familiaritat de l'equip amb aquesta eina, la popularitat (tant entre desenvolupadors com entre usuaris), el fet que està ben documentat o la facilitat d'ús de les eines d'administració.

2. Un petit recorregut per la interfície d'administració del WordPress

La pantalla principal d'administració del WordPress (en la versió 3.1.2) és la següent:

Figura 1. Pantalla principal del WordPress (v. 3.1.2.)

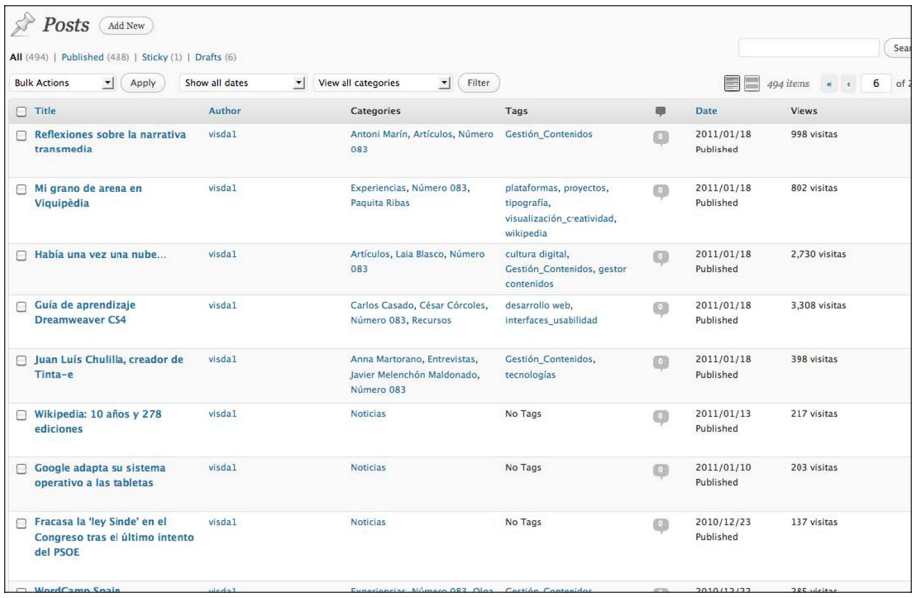


En el tauler *Right Now* podem apreciar l'estructura del WordPress (o, més exactament, l'estructura del WordPress que s'ha usat en aquesta instal·lació particular, que correspon a *Mosaic*):

Figura 2. Tauler *Right Now*

Podem apreciar que hi ha 488 *posts* (o entrades, que són la unitat de contingut que se sol usar en el WordPress), que s'agrupen en 207 categories diferents i als quals s'han assignat 233 *tags* (o etiquetes). El lector acostumat a treballar amb sistemes de gestió de continguts apreciarà que el nombre de categories és inusualment alt. Si accedim a la pantalla d'administració d'entrades en veurem el motiu.

Figura 3. Administració d'entrades



En l'adaptació del WordPress a *Mosaic* es van prendre una sèrie de decisions de disseny sobre l'estructura de les entrades. Vegem com és una entrada del WordPress i quines particularitats es presenten en aquesta instal·lació en concret. Cada entrada es compon del següent:

- **Títol** (en la captura de pantalla, en el primer ítem, "Reflexiones sobre la narrativa transmedia"). És el títol del contingut corresponent. No presenta particularitats.

- **Autor** (visda1). Correspon a l'usuari del WordPress que ha creat l'entrada. En la majoria de blogs aquest usuari es correspon amb l'autor del contingut. En molts llocs més complexos (com és el cas), amb l'usuari que crea l'entrada.

- **Categories** (Antoni Marín, Artículos, Número 083). Habitualment, les categories són una taxonomia de grandària petita o mitjana que agrupen les entrades del lloc web en temàtiques. No obstant això, en el cas de *Mosaic* s'utilitzen per a una sèrie d'objectius addicionals. Com es pot intuir, l'autor del contingut és una categoria, cada tipus de contingut també ho és (els tipus de contingut disponibles són Artículos, Entrevistas, Experiencias, Recursos, Noticias i Agenda) i, finalment, també s'usen les categories per a diferenciar cadascun dels nombres de la revista *Mosaic*. Actualment hi ha maneres millors de fer aquestes categoritzacions (mitjançant la funcionalitat de taxonomies a la mida del WordPress), però en el moment de creació del lloc WordPress aquesta funcionalitat estava encara en fase experimental i es va decidir no usar-la.

- **Etiquetes** (Gestión_Contenidos). Les etiquetes o *tags* s'usen per a anotar el contingut de cada entrada, habitualment intentant afegir informació semàntica. Així, etiquetarem amb "disseny web" totes les entrades que

facin referència a aquesta temàtica, independentment de la categoria a què pertanyi l'entrada. Sovint a mesura que es van afegint continguts al lloc van apareixent etiquetes noves, mentre que el nombre de categories se sol mantenir estable.

- **Comentaris**. Indica la quantitat de comentaris que han fet els lectors a aquesta entrada. La capacitat dels usuaris d'escriure comentaris es pot activar o desactivar individualment per a cada entrada. Com es pot apreciar, actualment a *Mosaic* moltes entrades no permeten deixar comentaris.

- **Data**. Indica la data i hora de l'última edició feta sobre l'entrada, i l'estat de l'entrada (que sol ser "esborrany" o "publicada").

- **Vistes**. Indica el nombre de vistes que ha tingut el contingut.

L'estructura de categories de *Mosaic*

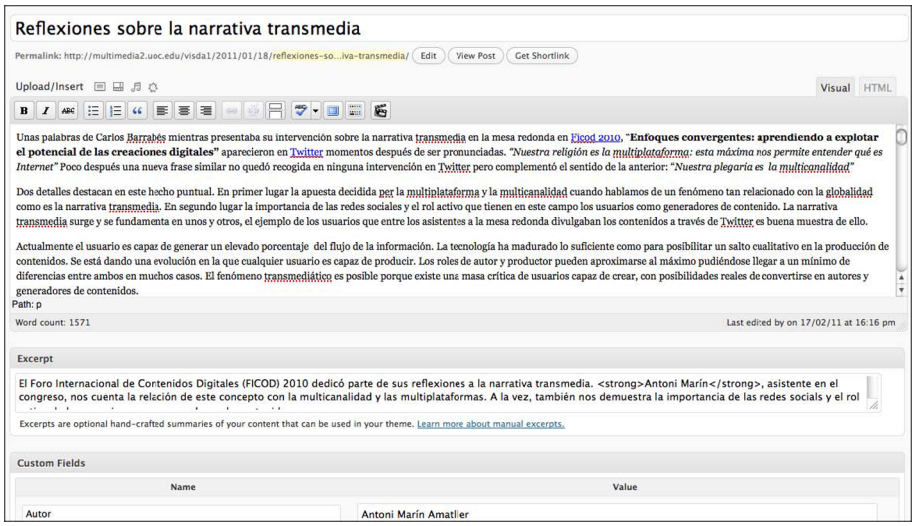
Hem vist que l'estructura de categories de *Mosaic* és bastant particular i que agrupa diferents tipus d'informació fent ús de categories i subcategories.

Aquestes són:

- Agenda
- Artículos
- Boletines (indica el número a què pertany el contingut específic)
  - Número 001
  - Número 002
  - …
- Entrevistas
- Experiencias
- Noticias
- Recursos
- www-Autores (indica el nom de l'autor del contingut, o el nom de la persona entrevistada en el cas de les entrevistes)
  - Adrià Ferrer Castro
  - Albert Alfonso Moreno
  - …

Vegem, finalment, el contingut d'una entrada. Si accedim a l'editor de contingut per a la primera entrada de la captura anterior veurem una cosa com el següent:

Figura 4. Editor de contingut



Tenim diferents ítems:

- El **títol de l'entrada**, que ja hem vist anteriorment.

- El ***permalink*** (o enllaç permanent) sota el qual es publicarà el contingut. Es compon de dues parts:
  - Una estructura.

  - Un ***slug*** (en aquest cas reflexions-sobre-la-narrativa-transmedia). L'*slug* se sol generar de manera automàtica a partir del títol de l'entrada (eliminant o modificant els caràcters que podrien resultar problemàtics). Podem canviar l'*slug*, normalment per a facilitar-ne la memorització o per a introduir termes importants per tal que els cercadors els tinguin en compte.

- El **contingut de l'entrada**. Disposem d'un editor visual i d'un en el qual podem treballar directament en HTML. Sobre la caixa de l'editor disposem d'una botonera des de la qual podem pujar tot tipus d'arxius (principalment, imatges) que volem associar a l'entrada.

- L'**extracte**. Si en algun lloc (la portada, per exemple) volem publicar un text breu i no tot el contingut de l'entrada, podem optar per deixar que el WordPress seleccioni automàticament les primeres paraules de l'entrada (podem configurar quantes paraules seleccionarà) o bé redactar un text alternatiu, com és el cas.

- Els **camps personalitzats** de l'entrada. Aquests se solen utilitzar per a afegir informació a l'entrada que usarem posteriorment. Tot seguit veurem quins són els camps personalitzats que s'usen a *Mosaic*.

Els camps personalitzats de *Mosaic*

En la instal·lació del WordPress per a *Mosaic* s'usen per defecte cinc camps personalitzats:

- **Autor**. Conté el nom de l'autor del contingut. Duplica la informació ja continguda en una subcategoria. Això és així perquè, en funció d'on volem usar aquesta informació, és més fàcil recuperar-la de la subcategoria o del camp personalitzat.

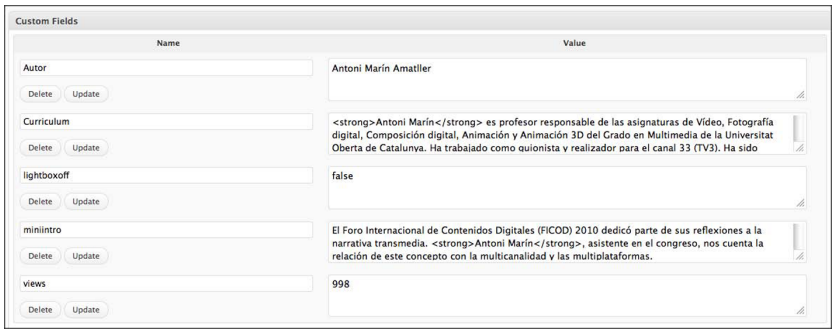
- **Currículum**. Com es pot veure, conté un text breu sobre l'autor.

- ***Lightboxoff***. És un camp generat i usat per un dels *plug-ins* del WordPress utilitzats a *Mosaic*, Lightbox 2, que permet crear galeries d'imatges.

- **Miniintro**. Conté un resum breu del contingut. Sol ser lleugerament diferent de l'extracte que s'ha vist anteriorment.

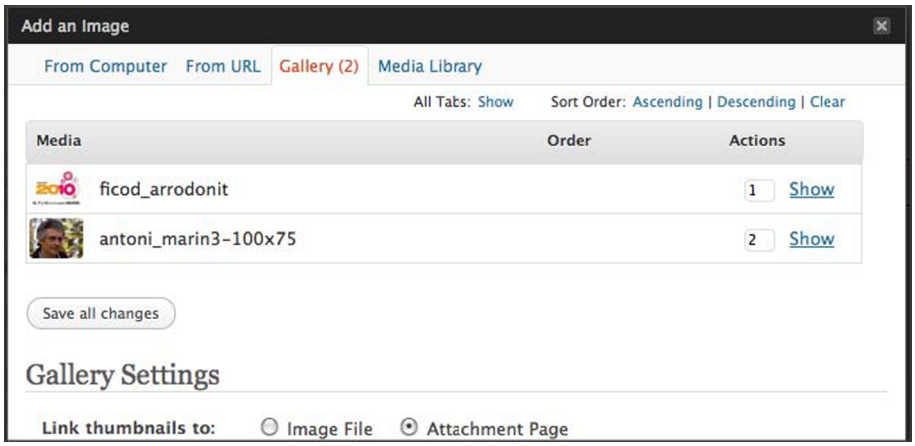
- ***Views***. És un camp generat i usat pel *plug-in* WP-PostViews per a explicar el nombre de visualitzacions que ha tingut cada contingut del lloc.

Figura 5. Camps personalitzats de *Mosaic*



Juntament amb l'entrada es poden utilitzar imatges. Vegem com es poden gestionar les entrades de la imatge (accedim a aquesta funcionalitat des de l'enllaç corresponent Upload/Insert, sobre la caixa d'edició del contingut de l'entrada).

Figura 6. Galeria d'imatges



A mesura que anem pujant imatges es crea, per a cada entrada, una galeria. Tal com s'ha definit, a *Mosaic* la primera imatge s'usa per als destacats en portada, mentre que la segona s'utilitza per al currículum de l'autor. Naturalment, si cal es pot canviar l'ordre de les imatges que s'han pujat en qualsevol moment.

Altres ítems que corresponen a cada entrada són l'estat de publicació (principalment "publicada" o "esborrany", encara que es poden definir altres estats si són necessaris per al flux de treball), la visibilitat (pública, privada o protegida per contrasenya) i la data de publicació (es pot disposar una data en el

futur, de manera que el contingut es publiqui en un moment determinat automàticament), com també els que ja s'han vist corresponents a les categories i etiquetes assignades a l'entrada.

Podem conèixer l'ús de tota la informació que hem vist en la captura de pantalla següent de l'entrada tal com apareix publicada:

Figura 7. L'entrada tal com queda publicada

3. *Plug-ins* WordPress

Hem vist que una de les maneres d'estendre la funcionalitat del WordPress és utilitzant i creant *plug-ins*. Els *plug-ins* permeten l'extensió o modificació del codi del WordPress sense tocar el codi font de l'aplicació, la qual cosa permet, si se segueixen unes bones pràctiques de programació, que una actualització del programari del WordPress no afecti les funcionalitats afegides.

Un *plug-in* WordPress és un programa, o un conjunt d'una o més funcions escrites en el llenguatge de *scripting* PHP, que afegeix un conjunt específic de característiques o serveis al *weblog* WordPress, que es poden integrar de manera transparent amb el *weblog* usant punts d'accés i mètodes oferts per l'API del WordPress.

Nota

Disposeu de la documentació oficial a https://codex.wordpress.org/Writing_a_Plugin

En el nostre cas particular, en què volem oferir una visualització del WordPress (i, en particular, del WordPress sobre el qual funciona *Mosaic*), el que volem fer són tres coses:

- En primer lloc, volem una manera d'**accedir amb facilitat i de manera eficient a la informació** continguda en WordPress i posar-la a la disposició de l'eina de visualització que crearem.

- En segon lloc, volem oferir una eina per a **administrar les diferents opcions** de personalització del *plug-in* (per a adaptar-lo a una instal·lació qualsevol del WordPress en funció de la seva estructura, però també per a modificar-ne l'aspecte gràfic, per exemple).

- Finalment, necessitem una manera simple d'**inserir la visualització** allà on sigui convenient.

3.1. L'esquelet d'un *plug-in*

Quins elements componen un *plug-in*?

- Un nom únic que identifiqui el *plug-in* (cal tenir en compte que hi ha, literalment, milers de *plug-ins* per a WordPress, per la qual cosa és important assegurar-nos que el nom del nostre *plug-in* no coincideix amb cap altre).

- Un arxiu, també amb **nom únic**, més una probable carpeta que contingui més arxius, si és necessari. Normalment aquest **arxiu** o grup d'arxius resideix en una carpeta estàndard de WordPress, `wp-content/plugins`.

De totes maneres, no podem assegurar que aquesta sigui la carpeta en què s'emmagatzemen els *plug-ins* en totes les instal·lacions del WordPress, per la qual cosa no és convenient assumir que és aquesta la carpeta, i és millor utilitzar la funció `plugins_url()` per a obtenir la ruta, si és necessari.

- En cas que vulguem publicar el *plug-in* en el repositori de wordpress.org, també haurem d'incloure un arxiu **readme.txt** que descrigui el *plug-in*.

Nota

Trobareu el format de l'arxiu readme.txt detallat a https://wordpress.org/extend/plugins/about/readme.txt

- No és un requisit, però sí és molt convenient, disposar d'una pàgina web per al *plug-in* en la qual s'informi sobre les funcionalitats i des de la qual es pugui donar suport als usuaris i recollir-ne els comentaris i suggeriments.

L'arxiu.php principal del *plug-in* ha de començar de la manera següent:

```
<?php
/*
Plugin Name: ______
Plugin URI: http://______
Description: _____
Version: _____
Author: _______
Author URI: http://____
License: por ejemplo, GPL2
*/
?>
```

I ha de seguir-li la informació de la llicència triada pel *plug-in*.

Nota

Trobareu les condicions de la llicència a https://wordpress.org/about/gpl/ i informació addicional a https://wordpress.org/about/license/

Una consideració sobre la llicència

El WordPress es distribueix sota la llicència de codi lliure GNU General Public License (en la versió 2 o posterior). Segons Wordpress.org les obres derivades del WordPress inclouen tant els temes com els *plug-ins* que es desenvolupin per al WordPress i hereten les condicions de la GPL. Per tant, s'han de llicenciar, com a mínim, amb una llicència compatible amb GPL v2.

Un recurs interessant per a generar automàticament l'esquelet d'un *plug-in* es pot trobar a http://themergency.com/generators/wordpress-plugin-boilerplate/.

3.2. Com s'activa un *plug-in*

Una vegada que hàgim escrit el codi per al *plug-in*, necessitarem llançar-lo en els moments convenients. Disposem de diverses maneres de fer-ho.

- Si volem que el *plug-in* s'activi sempre en determinats punts de l'execució del WordPress (per exemple, cada vegada que es dibuixi la capçalera, o cada vegada que editem una entrada), farem ús dels *hooks* (ganxos) del WordPress.

- Si el volem inserir "a mà" dins d'una plantilla, podem crear un *template tag* nou. Si, a més, volem que un editor de contingut pugui utilitzar aquesta *template tag* dins d'una entrada, podem crear posteriorment un `[comando]` que activi l'etiqueta.

Nota

Disposeu de més informació sobre l'estructura de plantilles a https://codex.wordpress.org/Stepping_Into_Templates

L'estructura bàsica d'arxius del WordPress

WordPress.org recomana que qualsevol tema per al WordPress respecti uns estàndards mínims en l'estructura que, posteriorment, faciliten el funcionament universal dels *plug-ins* que fan servir *hooks*.

Atès que habitualment els llocs web tenen almenys una capçalera i un peu en cada pàgina, s'assumeix que totes les plantilles contenen un `header.php` i un `footer.php` que s'encarreguen de dibuixar-los. Fins i tot quan els llocs web no tenen una capçalera o un peu, se solen incloure aquests arxius, de manera que podem comptar amb la seva existència.

Les barres laterals no són universals, però també són molt freqüents. En cas d'haver-n'hi, el codi resideix en un arxiu `sidebar.php`. Altres arxius freqüents són `single.php` (per al contingut principal de pàgines que continguin una única entrada), `category.php` (per a les pàgines amb arxius de categories) o `archive.php` (per als arxius temporals). No és la intenció d'aquest text entrar en profunditat en aquesta estructura d'arxius i el seu funcionament.

Template tags

Un *template tag* o etiqueta de plantilla és una etiqueta que diu a WordPress que faci alguna cosa. Hi ha etiquetes per a fer pràcticament qualsevol cosa que se'ns passi pel cap, des d'escriure el títol del blog (`<?php bloginfo('name'); ?>`) a fer la llista de les categories que hem creat (`<?php wp_list_cats(); ?>`). Si necessitem una etiqueta inexistent, podem definir-la creant un *plug-in*. Trobareu el catàleg complet d'etiquetes de plantilla del WordPress, amb la documentació corresponent, a https://codex.wordpress.org/Template_Tags.

Tipus de *hooks*

Disposem de dos tipus de *hooks* en WordPress:

- Les **accions** es llancen en punts específics o com a resposta a determinats esdeveniments.

- Els **filtres** s'interposen entre les consultes o actualitzacions per a la base de dades, modificant valors.

Un exemple d'acció seria `get_header`, que es crida just abans d'accedir a l'arxiu `header.php` i que ens podria servir per a inserir tot allò que volem que passi abans que el tema escrigui la primera línia d'HTML.

Per la seva banda, un exemple de filtre seria `the_content`, que s'aplica cada vegada que recuperem el contingut d'una entrada. Podríem usar aquest filtre per a buscar aparicions de determinades paraules i substituir-les per unes altres, o per a substituir les imatges pels seus textos alternatius.

Accions i filtres en WordPress

Trobareu la llista d'accions disponibles a https://codex.wordpress.org/Plugin_API/Action_Reference.

Trobareu la llista de filtres disponibles a https://codex.wordpress.org/Plugin_API/Filter_Reference.

Com s'afegeix una acció

El codi que hem d'usar per a afegir una acció és

```
add_action ( 'hook_utilitzat', 'funcio_a_cridar', [prioritat], [arguments_acceptats] );
```

- `hook_utilitzat` i `funcio_a_cridar` haurien de ser autoexplicatius.

- `prioritat` és un paràmetre optatiu. Per defecte el seu valor és 10, i valors més baixos indiquen prioritats més altes. Cal pensar que és probable que hi hagi altres *plug-ins* en funcionament que utilitzin el mateix *hook* que nosaltres i que pot ser important (o no) que el nostre *plug-in* s'executi tan aviat com sigui possible. És una mala pràctica sol·licitar una prioritat alta si no és imprescindible.

- `arguments_acceptats`, també opcional, indica el nombre de valors que accepta `funcio_a_cridar`.

3.3. Creació, manteniment i ús d'opcions

És molt probable que en un *plug-in* hi hagi determinats paràmetres que vulguem que l'usuari pugui configurar: el color de fons de l'element interactiu que volem crear, la mida, el nombre de categories que volem usar (o quines), l'ordre dels elements (de més gran a més petit, de més petit a més gran o aleatòriament), etc.

Per a fer-ho el WordPress disposa d'un sofisticat sistema de gestió d'opcions.

El primer pas que hem de fer és **crear l'opció**, amb

```
add_option($nom, $valor, $deprecated, $autoload)
```

L'únic paràmetre obligatori és el nom de l'opció. Si volem, podem especificar un valor per defecte amb $`valor`. `$autoload`, que per defecte val `yes`, i indica si l'opció es recuperarà automàticament quan el WordPress executi `wp_load_alloptions`. `$deprecated` és un paràmetre sense valor semàntic, que es conserva (només si volem especificar `$autoload`) per motius de compatibilitat amb versions antigues del WordPress.

Per a **recuperar el valor de l'opció** en qualsevol moment hem d'usar

```
get_option($opcio);
```

i per a actualitzar-ne el valor

```
update_option($nom_opcio, $valor_nou);
```

Així, per exemple, en el cas concret de la creació d'un *plug-in* per a *Mosaic*, podríem tenir:

```
/* Creació i destrucció de les opcions en activar i desactivar el <emphasis>plug-in</emphasis> */
/* En activar el <emphasis>plug-in</emphasis> es creen les opcions */
function activacion_VisDa() {
   add_option('VisDaWidth', '344','','yes');
   add_option('VisDaHeight', '392','','yes');
}

/* I en desactivar-lo s'esborren */
function desactivacion_VisDa() {
   delete_option('VisDaWidth');
   delete_option('VisDaHeight');
```

Per a fer que aquestes funcions es llancin en el moment de la instal·lació i desinstal·lació del *plug-in*, usarem un parell de *hooks*:

```
register_activation_hook(__FILE__,"activacion_VisDa");

register_deactivation_hook(__FILE__,"desactivacion_VisDa");
```

Una vegada hem creat les opcions que necessitarem, hem de crear un mecanisme per a poder-les gestionar des del tauler de control del WordPress. Per a això disposem de tres alternatives: crear una pàgina nova dins del menú d'administració, crear una subpàgina dins d'una que ja existeix o bé afegir les nostres opcions a una pàgina que ja existeix.

Per **crear una pàgina nova** farem:

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

function menu_del_plugin() {
   add_options_page($titol_pagina, $titol_menu, '
   manage_options', $slug_menu, $funcio);
}
```

- `titol_pagina` és el text que es mostrarà com a títol de pàgina en el navegador en carregar-la.

- `titol_menu` és el text que s'usarà en el text del menú.

- `manage_options` indica al WordPress que aquesta pàgina només s'ha de mostrar als usuaris que tinguin permisos concedits per a administrar les opcions del sistema.

- `slug_menu` és l'*slug* únic amb què posteriorment s'identificarà el menú.

- `funcio`, finalment, és la funció que es cridarà per a generar el contingut de la pàgina.

Administrar permisos en WordPress

El WordPress disposa d'un sistema sofisticat que ens permet tenir, per exemple, usuaris que puguin administrar el lloc complet, mentre que altres només puguin escriure nous continguts que hagi d'aprovar una tercera persona abans de publicar-se.

Si volem **crear una subpàgina** (serà el més convenient, en general) el que farem serà:

```
add_submenu_page( $slug_pare, $titol_pagina, $titol_menu,
'manage_options', $slug_menu, $funcio);
```

L'única opció nova és `slug_pare` que, com podem imaginar, identifica la pàgina del menú de la qual haurà de penjar la nostra i que pot ser, per exemple, `plugins.php`, `tools.php` o `options_general.php` (per a situar la subpàgina en el menú *Settings*).

Finalment, en molts casos serà convenient **crear una secció dins d'una pàgina** o subpàgina existent. Per a això farem servir:

```
add_settings_section($id, $titol, $funcio, $pagina);
```

- id és un identificador únic per a la secció.

- titol és el títol de la secció.

- funcio és la funció encarregada d'escriure els continguts de la secció.

- pagina és la pàgina a la qual s'ha d'afegir la secció. Pot ser `general`, `reading`, *`writing`*, `media`, o l'*slug* d'una pàgina o subpàgina que nosaltres hàgim creat prèviament.

La funció per a generar el contingut serà del tipus:

```
function opciones_plugin() {
```

```
?>
<div>
<h2>El títol</h2>
Text descriptiu.
<form action="options.php" method="post">
<?php settings_fields('plugin_options'); ?>
<?php do_settings_sections('plugin'); ?>
<input name="Submit" type="submit" value="<?php esc_attr_e('Guardar canvis); ?>" />
</form></div>
<?php
}?>
```

Nota

Fixeu-vos com podem sortir i entrar del codi en PHP a voluntat (amb `?>` i `<?php`) per a poder escriure el contingut HTML amb facilitat.

`settings_fields` pren com a argument, en l'exemple, `'plugin_options'`, un nom arbitrari d'un grup d'opcions.

Per la seva banda, `do_settings_sections` s'encarrega d'escriure totes les seccions que hem afegit prèviament a una pàgina, que en l'exemple és `'plugin'`.

També haurem d'afegir una acció per inicialitzar l'administració del nostre *plug-in*:

```
<?php

add_action('admin_init', 'plugin_admin_init');
```

Aquesta acció crida una funció com la següent:

```
function plugin_admin_init(){

register_setting( 'plugin_options', 'plugin_option_name',
'plugin_option_name_validate' );
```

en què `plugin_options` ve del fragment de codi anterior, `'plugin_option_name'` és el nom de cadascuna de les opcions que hem de registrar, i el tercer paràmetre (opcional) és el nom de la funció que emprarem per a netejar i validar el valor de l'opció.

A continuació haurem d'afegir cadascun dels camps que necessitem, amb

```
add_settings_field('plugin_text_string', 'Plugin Text In-
put', 'plugin_setting_string', 'plugin', 'plugin_main');
```

```
}?>
```

Els paràmetres per a `add_settings_field` són, per ordre, els següents:

- Un identificador per a cada camp (en el nostre cas, `'plugin_text_string'`);
- El títol del camp (`'Plugin Text Input'`).
- Una funció que mostra el camp (`'plugin_setting_string'`).
- La pàgina en què ha d'aparèixer el camp (`'plugin'`).
- La secció en què ha d'aparèixer (és un paràmetre opcional, i en l'exemple hi hem donat el valor `'plugin_main'`).
- A continuació afegirem, si són necessaris, els arguments que s'hagin de passar a la funció que completa el camp. Actualment, l'únic argument que es pot passar és un `label_for`, per a assignar al camp una etiqueta diferent del títol.

Les funcions que creen cada camp són del tipus:

```
<?php function plugin_setting_string() {

$options = get_option('plugin_options');

echo "<input id='plugin_text_string'
name='plugin_options[text_string]' size='40' type='text'
value='{$options['text_string']}' />";

} ?>
```

Finalment, les opcions de validació i/o "sanitització" són del tipus:

```
function plugin_options_validate($input) {

$newinput['text_string'] = trim($input['text_string']);

if(!preg_match('/^[a-z0-9]{32}$/i', $newinput['text_string'])) {
   $newinput['text_string'] = '';
}

return $newinput;
}
```

Seguint amb l'exemple d'un *plug-in* per a *Mosaic*, el codi corresponent a la pàgina d'opcions podria ser el següent:

```
/* Definició de la pàgina d'opcions */

function VisDa_opciones() {

/* Comprova si hem entrat a la pàgina i canviat opcions */
   if( isset($_POST[ancho]) ) {
      $opt_ancho = $_POST['ancho'];
      $opt_alto = $_POST['alto'];
      update_option('VisDaWidth', $opt_ancho );
      update_option('VisDaHeight', $opt_alto );
?>
<div class="updated">
<p><strong>Opcions guardades.</strong></p></div>
<?php
   }
/* Escribim el formulari de les opcions */
 print("<div class='wrapper'>");
 screen_icon();
 print("<h2>Opcions del <emphasis>plug-in</emphasis> VisDa</h2>");
 ?>
 <form id="opciones_VisDa" method="post" actions="options.php">
  <p>
   <label for="ancho">Amplada:</label>
   <input name="ancho" type="text" id="ancho"
    value="<?php echo get_option('VisDaWidth','false'); ?>"
    size="10" />
  </p>
  <p>
   <label for="alto">Alçada:</label>
   <input name="alto" type="text" id="alto"
    value="<?php echo get_option('VisDaHeight','false'); ?>"
    size="10" />
  </p>
  <p>
   <input type="submit" name="enviar" id="enviar"
    value="Guardar canvis" class="button-primary" />
  </p>
 </form>
    <?php
}

/* Crea la pàgina d'opcions */
function VisDa_opcions_menu() {
 add_options_page('Opcions VisDa',
```

```
    'VisDa','manage_options','plugin_VisDa','VisDa_opcions');
}
```

3.4. Algunes recomanacions finals

És molt útil i recomanable consultar els estàndards de programació del WordPress, a https://codex.wordpress.org/WordPress_Coding_Standards.

És molt convenient prefixar tots els nostres noms de funció seguint alguna convenció relacionada amb el nom del *plug-in* per a evitar coincidències amb funcions d'altres *plug-ins*.

En el nostre cas, doncs, les funcions tindrien noms del tipus

```
visda_nomfuncio()
```

Quant a la base de dades:

- No usem `wp_` com a prefix per als noms de taules, sinó `$wpdb->prefix` (aquesta segona opció funcionarà sempre, inclosos els casos en què s'hagi canviat el prefix habitual).

- Escriure a la base de dades és una operació molt cara i lenta, per la qual cosa hem de minimitzar les escriptures i, quan ho fem, és molt millor utilitzar els mètodes natius del WordPress que accedir a la base de dades directament, sempre que això sigui possible.

- Llegir de la base de dades és bastant més eficient que escriure, però continua essent una operació costosa: com menys `SELECT`, millor.

Un altre coll d'ampolla potencial és la càrrega d'estils CSS i arxius JavaScript. Per a evitar-los és molt convenient fer servir `wp_enqueue_style()` i `wp_enqueue_script()`.

Finalment, mentre provem, pot ser útil `define('WP_DEBUG', true);` en el `wp-config.php`. Això farà, per exemple, que en la pàgina web apareguin qualssevol missatges d'error que es puguin generar, inclosos errors de MySQL i PHP.