WordPress: gestire gli allegati con e senza AJAX

Gli allegati di WordPress sono dei particolari tipi di post che vengono associati ai post tradizionali.

Di solito sono immagini, ma possono anche essere file di altro tipo (come i PDF). Quando si seleziona un allegato dalla Media Library e lo si inserisce nel post, l'allegato viene associato al post. Vediamo come gestire i Loop degli allegati.

Supponiamo di voler visualizzare tutte le immagini associate ad un post ad eccezione dell'immagine in evidenza:

```
// In single.php

global $post;

$attachments = get_posts( array(
                        'post_type' => 'attachment',
                        'posts_per_page' => -1,
                        'post_parent' => $post->ID,
                        'post_mime_type' => array(
'image/jpeg', 'image/png' ),
                        'exclude'     =>
get_post_thumbnail_id()
) );

if ( $attachments ) {
          foreach ( $attachments as $attachment ) {
                $thumb_img =
```

```
wp_get_attachment_image_src( $attachment->ID, 'full'
);
                echo '<img src="' . $thumb_img[0] .
'" alt="" />';
        }

}
```

In questo modo tutte le immagini associate ad un post verranno visualizzate a parte. Questa è una tecnica molto efficace per creare gallerie di immagini a partire da un dato post contenente un numero imprecisato di allegati.

Abbiamo utilizzato due funzioni:

1. get_post_thumbnail_id()
2. wp_get_attachment_image_src()

La prima ci serve ad ottenere l'ID dell'immagine in evidenza e ad escluderla dal Loop, la seconda l'URL assoluto dell'immagine.

Ora che abbiamo capito il meccanismo, possiamo creare qualcosa di più complesso. Immaginiamo di voler creare uno shortcode per visualizzare una lista di file PDF associati ad un post. Ecco come possiamo fare:

```
// In functions.php

function pdf_list() {
        global $post;
        $html = '';

        $args = array(
                'post_type' => 'attachment',
                'posts_per_page' => -1,
```

```
                'post_parent' => $post->ID,
                'post_mime_type' => 'application/pdf'
        );

        $pdfs = get_posts( $args );
        if( $pdfs ) {
                $html .= '<ul id="pdf-list">' . "\n";
                foreach( $pdfs as $pdf ) {
                        $pdf_url =
wp_get_attachment_url( $pdf->ID );
                        $pdf_name_arr = explode( '/',
$pdf_url );
                        $pdf_name = array_reverse(
$pdf_name_arr );
                        $html .= '<li><a href="' .
$pdf_url . '">' . $pdf_name[0] . '</a></li>' . "\n";
                }
                $html .= '</ul>' . "\n";
        }

        return $html;
}

add_shortcode( 'pdf-list', 'pdf_list' );
```

La funzione wp_get_attachment_url() serve a reperire l'URL assoluto dell'allegato. Avrete notato come è sufficiente modificare il parametro `post_mime_type` della funzione `get_posts()` per reperire ogni tipo di allegato.

Altri tipi MIME utili sono:

- Documenti Word: `application/msword`
- Documenti PowerPoint: `application/vnd.ms-powerpoint`

- Documenti Excel: `application/vnd.ms-excel`
- File ZIP: `application/zip`

Tornando al nostro esempio iniziale, notiamo che c'è un problema di performance con le immagini: se infatti carichiamo tutte le immagini nel post singolo potremo notare dei rallentamenti, specie se le immagini sono molte.

Possiamo rendere il caricamento asincrono con AJAX. Come prima cosa modifichiamo il file `single.php` in modo da poter reperire l'ID del post ed avere una call to action per AJAX:

```
// single.php

<?php
while( have_posts() ):
        the_post();
?>
        <article class="post" id="p-<?php the_ID(); ?
>">
                <?php the_content(); ?>
                <p><a href="#p-<?php the_ID(); ?>"
id="get-gallery"><?php _e( 'Visualizza galleria',
'tuotema' ); ?></a></p>
        </article>
<?php
endwhile;
?>
```

A questo punto registriamo una action AJAX in `functions.php`:

```
add_action( 'wp_ajax_my_gallery', 'my_gallery' ); //
```

```
utenti loggati
add_action( 'wp_ajax_nopriv_my_gallery', 'my_gallery'
); // tutti i visitatori

function my_gallery() {
        $post_id = $_GET['post_id'];
        $id = (int) $post_id;
        $html = '';

        // L'ID del post deve essere un intero e non
può superare le 11 cifre

        if( strlen( $post_id ) <= 11 && filter_var(
$id, FILTER_VALIDATE_INT ) ) {
                $args = array(
                        'post_type' =&gt;
'attachment',
                        'posts_per_page' =&gt; -1,
                        'post_parent' =&gt; $id,
                        'post_mime_type' =&gt; array(
'image/jpeg', 'image/png' ),
                        'exclude'     =&gt;
get_post_thumbnail_id( $id )
                );
                $images = get_posts( $args );

          if( $images ) {

                foreach( $images as $image ) {
                        $img =
wp_get_attachment_image_src( $image->ID, 'full' );
                        $html .= '<img src="' .
$img[0] . '" alt="" />';
                }
```

```
          }
        }

        echo $html;
        exit();
}
```

Lo step finale è utilizzare jQuery:

```
(function( $ ) {
        $(function() {
                var $action = $( "#get-gallery" );
                if( $action.length ) {
                        $action.parent().after( "<div
id='gallery'></div>" );

                        $action.click(function( e ) {
                                e.preventDefault();
                                var href = $( this
).attr( "href" ),
                                        postID =
href.replace( "#p-", "" ),
                                        ajaxURL =
"/wp-admin/admin-ajax.php";

                                $.get( ajaxURL, {
action: "my_gallery", post_id: postID }, function(
resp ) {
                                        $( "#gallery"
).html( resp );
                                });
```

```
                        });
                }

        });

})( jQuery );
```

Conclusione

Ci sono molte possibilità da sfruttare con i Loop custom degli allegati di WordPress. La cosa importante è essere selettivi e scegliere solo quegli allegati che effettivamente ci servono.