_doing_it_wrong

Improving Your Development Skills Through Examining Bad Practices

Zack Tollman ‹› Senior Web Engineer ‹› 10up LLC
tollmanz.com ‹› 10up.com ‹› @tollmanz

SCRETS-LATED POSTS

It’s all the buzz

SCRETS-LATED POSTS

It’s all the buzz

SCRETS-LATED POSTS ALABLE

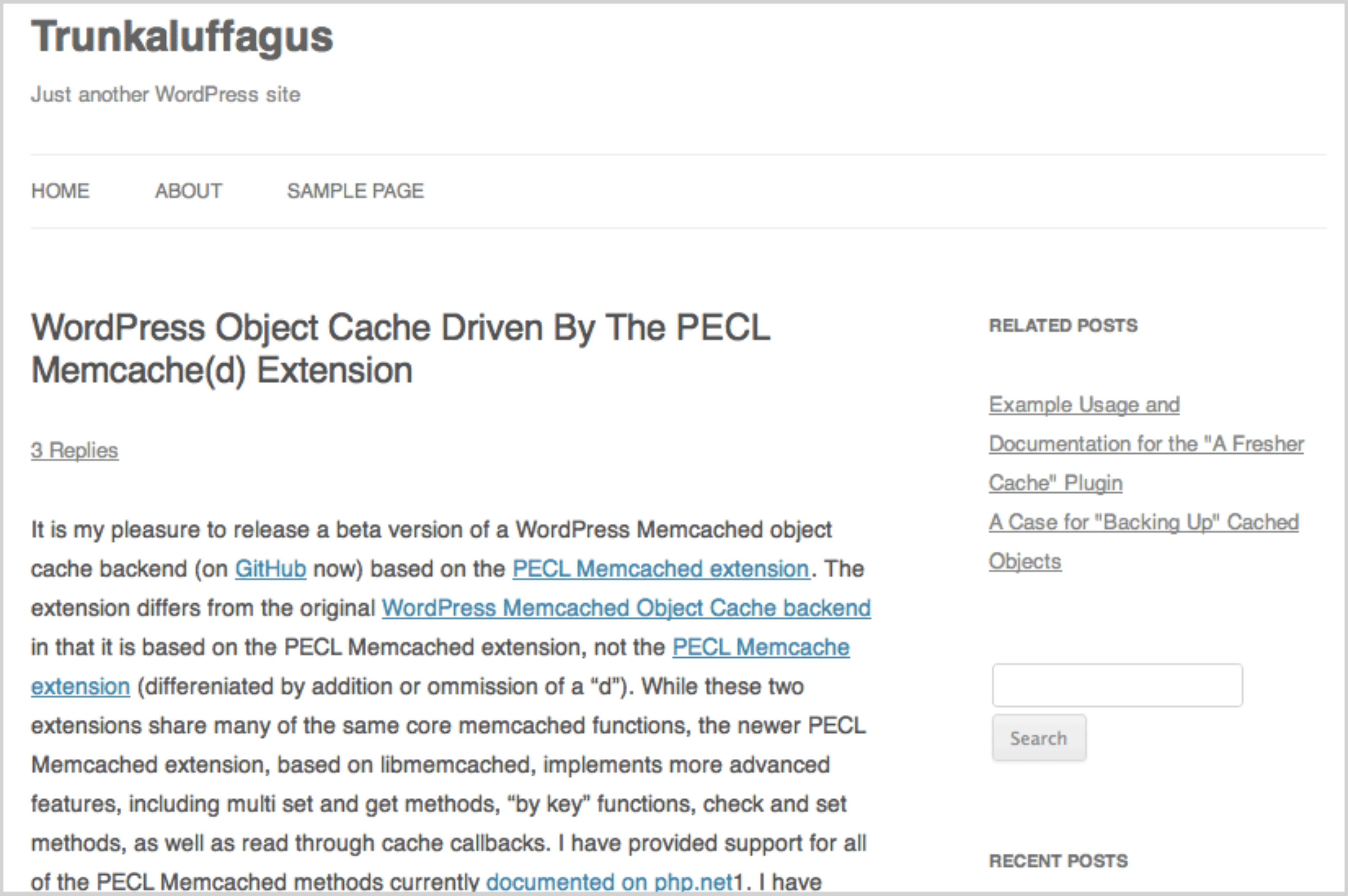
It's all the buzz

SCRETS-LATED POSTS ONSIVE

It’s all the buzz

SCRETS-LATED POSTS INA READY

It's all the buzz



Trunkaluffagus

Just another WordPress site

HOME ABOUT SAMPLE PAGE

WordPress Object Cache Driven By The PECL Memcache(d) Extension

3 Replies

It is my pleasure to release a beta version of a WordPress Memcached object cache backend (on GitHub now) based on the PECL Memcached extension. The extension differs from the original WordPress Memcached Object Cache backend in that it is based on the PECL Memcached extension, not the PECL Memcache extension (differeniated by addition or ommission of a “d”). While these two extensions share many of the same core memcached functions, the newer PECL Memcached extension, based on libmemcached, implements more advanced features, including multi set and get methods, “by key” functions, check and set methods, as well as read through cache callbacks. I have provided support for all of the PECL Memcached methods currently documented on php.net1. I have

RELATED POSTS

Example Usage and Documentation for the "A Fresher Cache" Plugin

A Case for "Backing Up" Cached Objects

RECENT POSTS

Game of Globals

“wp” Rules the Kingdom

- A single instance of WP_Query is issued for a page load
- $wp_query, $post, and $wp_the_query are set
- These are the rightful rulers

"query_posts" is Power Hungry

- "query_posts" completely replaces $wp_query
- Changes the $post global object
- After using “query_posts”, context has completely changed

"WP_Query" is Righteous with But Not Innocent

- "WP_Query" does not replace any globals
- "the_post" replaces the $post global
- "wp_reset_postdata" resets the $post global
- Note that "wp_reset_query" is usually unnecessary

Just Leave Globals Alone

- Don’t interact with globals when a function does it for you
    - e.g., unset($wp_meta_boxes[‘blah’]) / remove_meta_box(‘blah’), $post->ID / get_the_ID()
- If given $post, $post_id, $wp_query, use it
    - e.g., save_post, pre_get_posts

Bad Developer

```php
add_action( 'save_post', 'my_save_post', 10, 2 );

function my_save_post( $post_id, $local_post ) {
    /**
     * Check nonce, autosave, permissions,
     * validate and sanitize data.
     */

    global $post;
    update_post_meta( $post->ID, '_my_meta', $data );
}
```

Better Developer

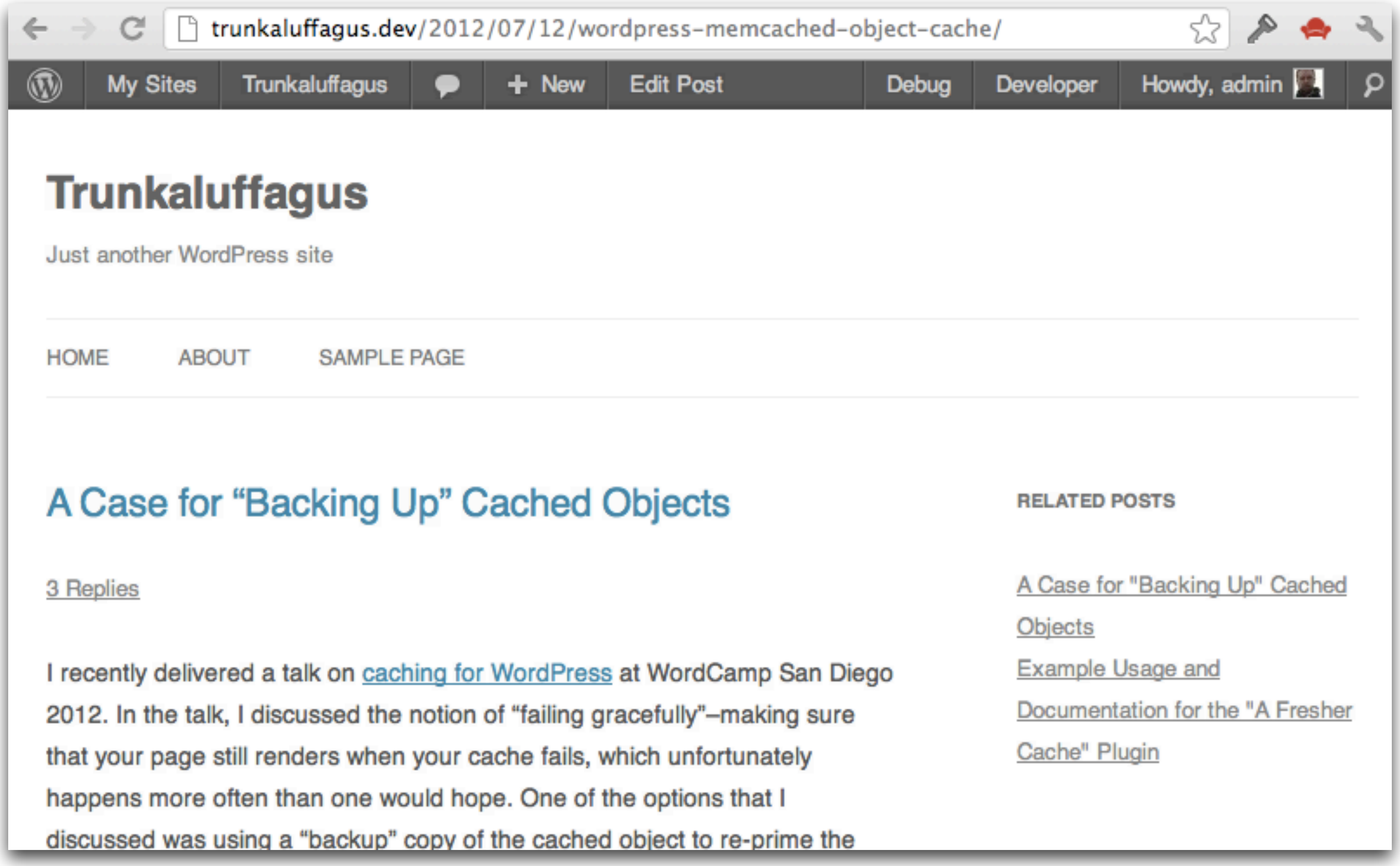
```
add_action( 'save_post', 'my_save_post', 10, 2 );

function my_save_post( $post_id, $local_post ) {
    /**
     * Check nonce, autosave, permissions,
     * validate and sanitize data.
     */

    update_post_meta( $post_id, '_my_meta', $data );
}
```

SCRETS-LATED POSTS

It's all the buzz



trunkaluffagus.dev/2012/07/12/wordpress-memcached-object-cache/

Trunkaluffagus

Just another WordPress site

HOME ABOUT SAMPLE PAGE

A Case for “Backing Up” Cached Objects

3 Replies

I recently delivered a talk on caching for WordPress at WordCamp San Diego 2012. In the talk, I discussed the notion of “failing gracefully”–making sure that your page still renders when your cache fails, which unfortunately happens more often than one would hope. One of the options that I discussed was using a “backup” copy of the cached object to re-prime the

RELATED POSTS

A Case for "Backing Up" Cached Objects

Example Usage and Documentation for the "A Fresher Cache" Plugin

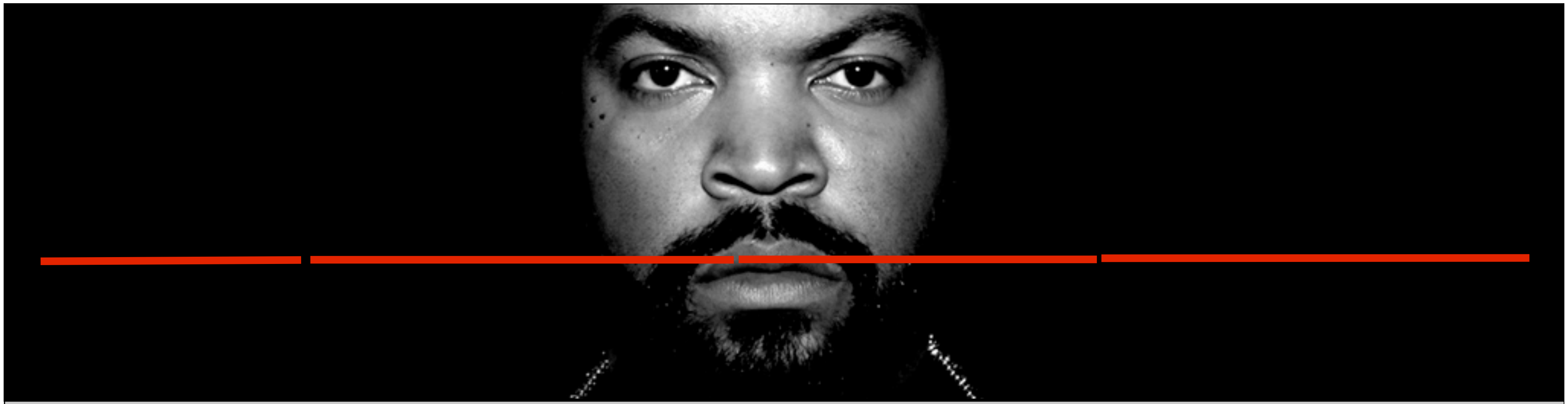
```
$args = array(
    'post_type' => 'post',
    'posts_per_page' => $number,
    'tax_query' => array(
        'relation' => 'OR',
        array(
            'taxonomy' => 'category',
            'field' => 'id',
            'terms' => $category_ids
        ),
        array(
            'taxonomy' => 'post_tag',
            'field' => 'id',
            'terms' => $tag_ids
        )
    ),
    'post__not_in' => array( get_the_ID() ),
    'orderby' => 'rand'
);
```

Bad Developer

```
<?php $posts = query_posts( $args ); ?>
<ul>
    <?php foreach ( $posts as $key => $post ) : ?>
        <li>
            <a href="<?php echo get_permalink( $post-
>ID ); ?>" title="<?php echo $post->post_title; ?>">
                <?php echo $post->post_title; ?>
            </a>
        </li>
    <?php endforeach; ?>
</ul>
```

Better Developer

```
<?php $posts = new WP_Query( $args ); ?>
<?php if ( $posts->have_posts() ) : ?>
<ul>
    <?php while ( $posts->have_posts() ) : $posts->the_post(); ?>
        <li>
            <a href="<?php the_permalink() ?>" title="<?
php the_title_attribute(); ?>">
               <?php the_title(); ?>
            </a>
        </li>
    <?php endwhile; ?>
</ul>
<?php endif; ?>
<?php wp_reset_postdata(); ?>
```



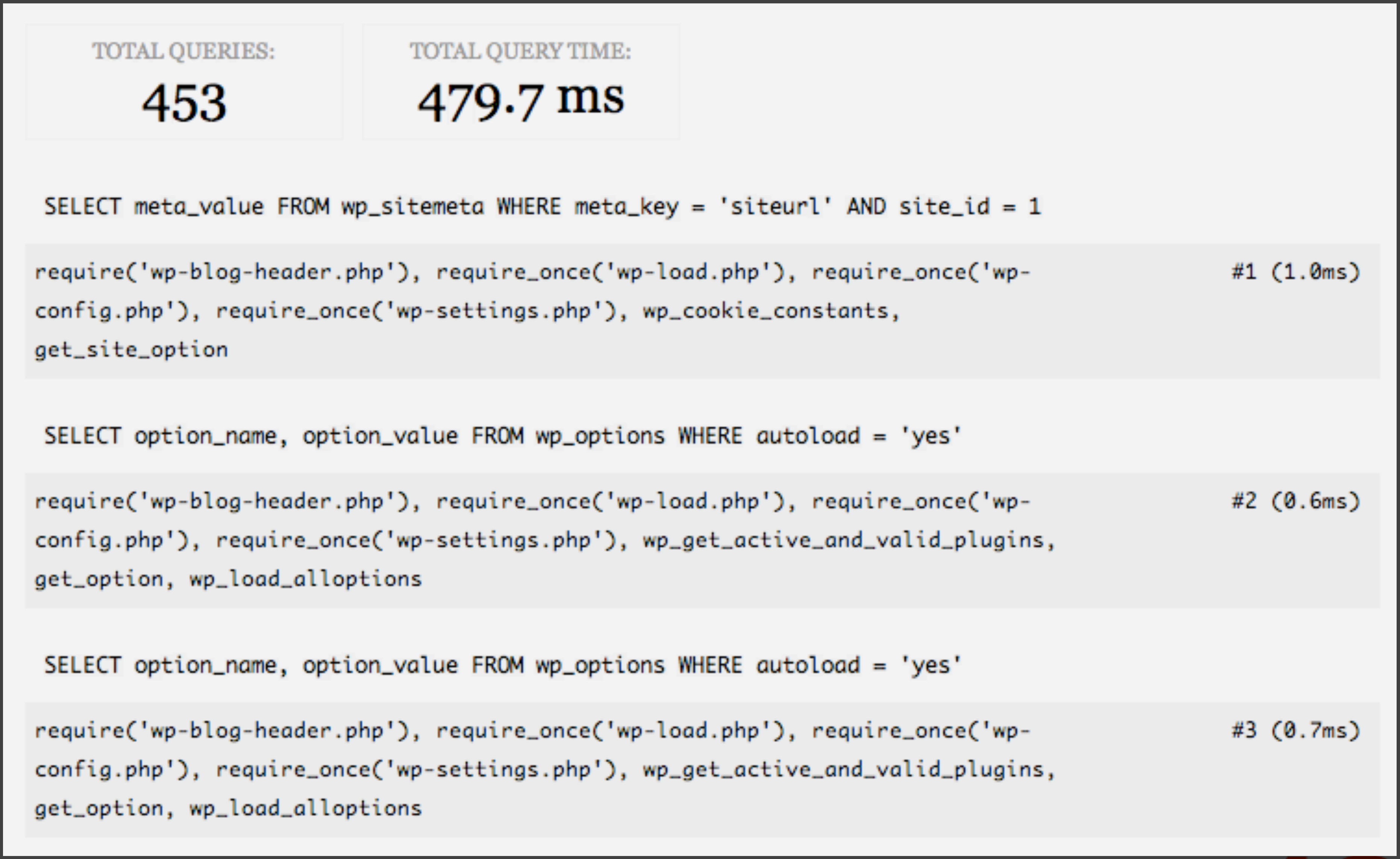
It is best practice to review your code for performance issues, while developing WordPress plugins and themes in order to avoid potential server malfunction that will result in your server making a “chickity” sound, as though the server is experiencing mechanical failure.

Things That Are Bad for Your Health

- Expensive queries
  - e.g., tax_query, meta_query, post__not_in, orderby = ‘rand’
- HTTP requests
- Intense PHP operations
  - e.g., image manipulation, a lot of regex, big loops

Check Yo Self

- Measure every query (within reason)
    - Debug Bar Extender: Queries Tab



TOTAL QUERIES:
453

TOTAL QUERY TIME:
479.7 ms

```
 SELECT meta_value FROM wp_sitemeta WHERE meta_key = 'siteurl' AND site_id = 1

require('wp-blog-header.php'), require_once('wp-load.php'), require_once('wp-          #1 (1.0ms)
config.php'), require_once('wp-settings.php'), wp_cookie_constants,
get_site_option

 SELECT option_name, option_value FROM wp_options WHERE autoload = 'yes'

require('wp-blog-header.php'), require_once('wp-load.php'), require_once('wp-          #2 (0.6ms)
config.php'), require_once('wp-settings.php'), wp_get_active_and_valid_plugins,
get_option, wp_load_alloptions

 SELECT option_name, option_value FROM wp_options WHERE autoload = 'yes'

require('wp-blog-header.php'), require_once('wp-load.php'), require_once('wp-          #3 (0.7ms)
config.php'), require_once('wp-settings.php'), wp_get_active_and_valid_plugins,
get_option, wp_load_alloptions
```

Check Yo Self

- Measure every query (within reason)
  - Debug Bar Extender: Queries Tab
  - Use a large database
- Time PHP operations
  - Use a debugging tool (xdebug) or manually time the operation
- Trust your gut

Improving Performance

- Cache Expensive Objects
  - Transients API: http://codex.wordpress.org/Transients_API
  - Cache on admin requests to protect against race conditions
- Refactor
  - Filter query results with PHP
  - Change data structure

SCRETS-LATED POSTS

It’s all the buzz

```php
$posts = new WP_Query( array(
    'post_type' => 'post',
    'posts_per_page' => $number,
    'tax_query' => array(
        'relation' => 'OR',
        array(
            'taxonomy' => 'category',
            'field' => 'id',
            'terms' => $category_ids
        ),
        array(
            'taxonomy' => 'post_tag',
            'field' => 'id',
            'terms' => $tag_ids
        )
    ),
    'post__not_in' => array( get_the_ID() ),
    'orderby' => 'rand'
) );
```

```
SELECT SQL_CALC_FOUND_ROWS  wp_posts.ID FROM
wp_posts  INNER JOIN wp_term_relationships ON
(wp_posts.ID = wp_term_relationships.object_i
d) INNER JOIN wp_term_relationships AS tt1 ON
(wp_posts.ID = tt1.object_id) WHERE 1=1  AND
wp_posts.ID NOT IN (78) AND ( wp_term_relatio
nships.term_taxonomy_id IN (5) OR tt1.term_ta
xonomy_id IN (11,13,15,16) ) AND wp_posts.pos
t_type = 'post' AND (wp_posts.post_status = '
publish' OR wp_posts.post_status = 'private')
GROUP BY wp_posts.ID ORDER BY RAND() DESC LIM
IT 0, 3
```

24 Rows in wp_posts

~0.6 - 0.8 Milliseconds!

~270,000 Rows in wp_posts

~6 - 7 **Seconds**!

Bad Developer

```
<?php $posts = new WP_Query( $args ); ?>
<?php if ( $posts->have_posts() ) : ?>
    <ul>
        <?php while ( $posts->have_posts() ) : ?>
        <?php $posts->the_post(); ?>
            <li><?php the_title(); ?></li>
        <?php endforeach; ?>
    </ul>
<?php endif; ?>
```

Better Developer

```
<?php
    $cache_key = 'screts-posts' . get_the_ID();
    $posts = get_transient( $cache_key );

    if ( false === $posts ) {
        $posts = new WP_Query( $args );
        set_transient( $cache_key, $posts, 86400 );
    }
?>

<?php if ( $posts->have_posts() ) : ?>
    <ul>
        <?php while ( $posts->have_posts() ) : ?>
        <?php $posts->the_post(); ?>
            <li><?php the_title(); ?></li>
        <?php endforeach; ?>
    </ul>
<?php endif; ?>
```

Data Management

Know Your Tables

- Clear
  - posts, postmeta
  - users, usermeta
  - comments, commentmeta
  - links (not for long: http://core.trac.wordpress.org/ticket/21307)
- Clear, but non consistent
  - terms, term...what?!?, term_relationships, term_taxonomy
- Murky
  - options

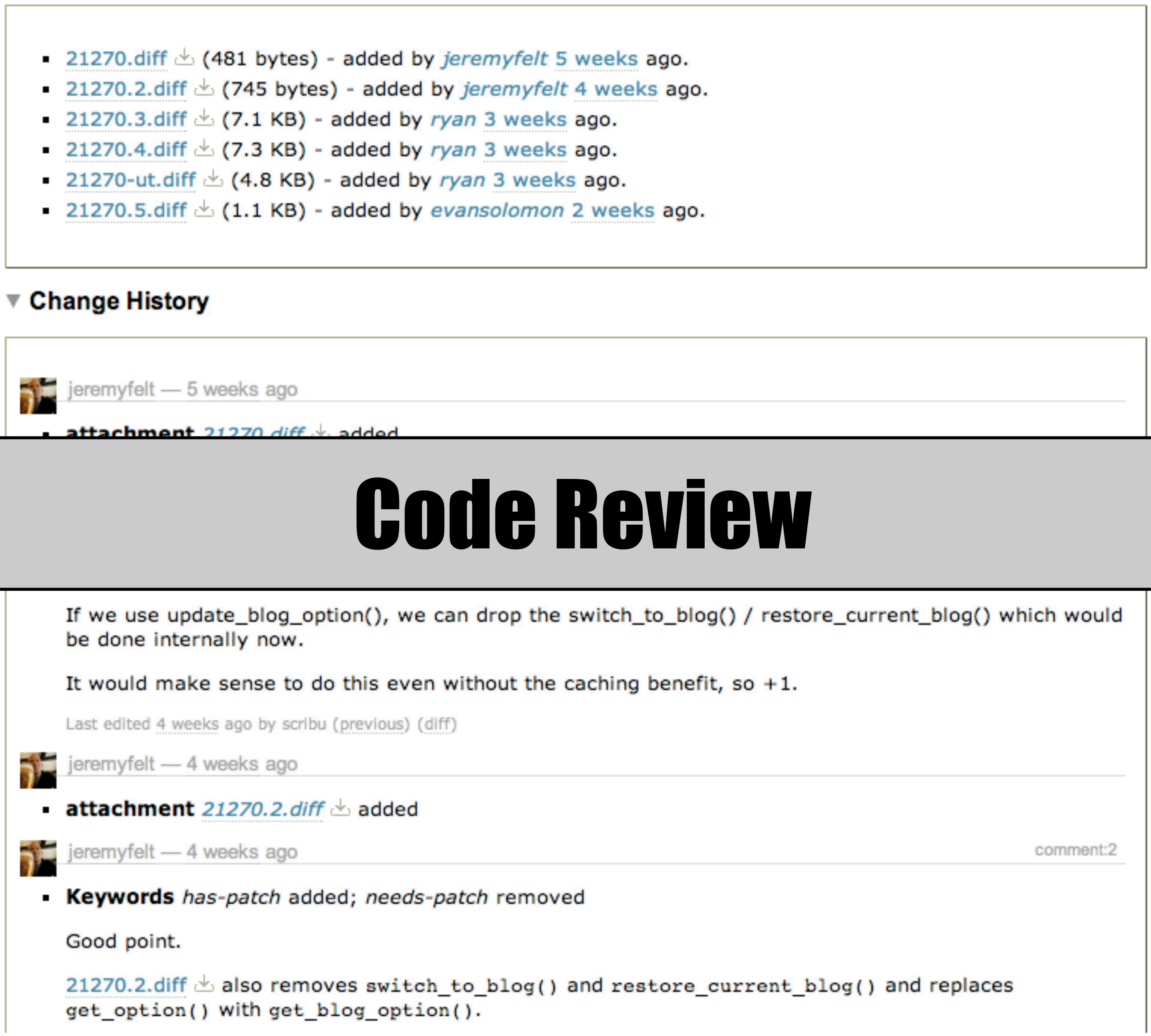
Options

- Ideal use: small amounts of "settings" data
- As a last resort, can contain other data
- Be mindful of autoload

Everything is a Custom Content Type...sorta kinda

- Custom **Post** Type is a misnomer
  - e.g., Twitter accounts, deprecated notices, layouts, orders, forums
- Know the limitations of WP_Query
  - e.g., inability to find individual value in metadata array using meta_query

Code Review



- 21270.diff (481 bytes) - added by *jeremyfelt* 5 weeks ago.
- 21270.2.diff (745 bytes) - added by *jeremyfelt* 4 weeks ago.
- 21270.3.diff (7.1 KB) - added by *ryan* 3 weeks ago.
- 21270.4.diff (7.3 KB) - added by *ryan* 3 weeks ago.
- 21270-ut.diff (4.8 KB) - added by *ryan* 3 weeks ago.
- 21270.5.diff (1.1 KB) - added by *evansolomon* 2 weeks ago.

Change History

jeremyfelt — 5 weeks ago

- **attachment** *21270.diff* added

If we use update_blog_option(), we can drop the switch_to_blog() / restore_current_blog() which would be done internally now.

It would make sense to do this even without the caching benefit, so +1.

Last edited 4 weeks ago by scribu (previous) (diff)

jeremyfelt — 4 weeks ago

- **attachment** *21270.2.diff* added

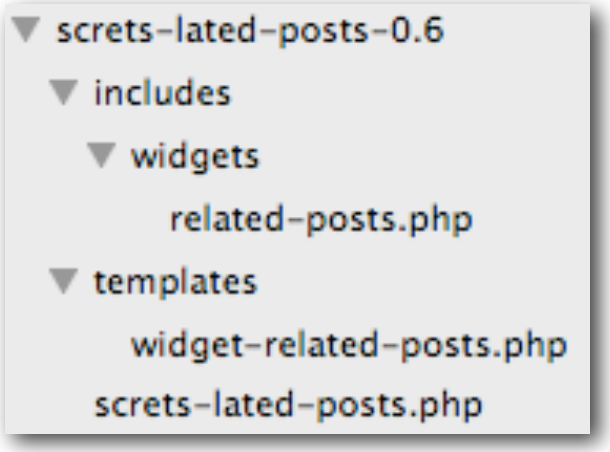
jeremyfelt — 4 weeks ago comment:2

- **Keywords** *has-patch* added; *needs-patch* removed

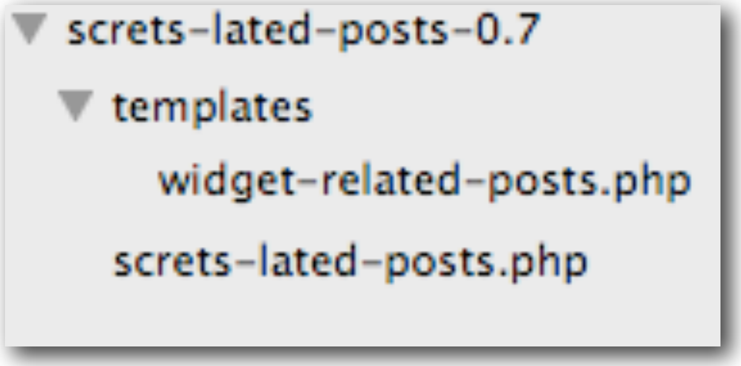
Good point.

21270.2.diff also removes `switch_to_blog()` and `restore_current_blog()` and replaces `get_option()` with `get_blog_option()`.

Before


```
▼ screts-lated-posts-0.6
  ▼ includes
    ▼ widgets
        related-posts.php
  ▼ templates
      widget-related-posts.php
    screts-lated-posts.php
```

After


```
▼ screts-lated-posts-0.7
  ▼ templates
      widget-related-posts.php
    screts-lated-posts.php
```

Meaningful Quotes Taken Out of Context

Success is the ability to go from one failure to another with no loss of enthusiasm.

Winston Churchill

Not doing it wrong is doing it wrong

Me

However

Insanity: doing the same thing over and over again and expecting different results

Unknown

Insanity: doing it wrong over and over again and expecting different results

Me + Unknown

Challenge: Be a Better Developer

- Improve **one** piece of code
- If your code is perfect, help **one** person improve his/her code

github.com/tollmanz/screts-lated-posts

@tollmanz

Bonus Material!!!!

EXTREME
OPEN SOURCE

Custom SQL

- Custom SQL is the fastest way to expose your database
  - More potential for error
- Performance issues may result due to custom SQL
  - Custom queries are uncached out of the box
- WordPress APIs provide more secure and performant database interaction

Evil Developer

```
global $wpdb;
$sql = 'SELECT * FROM wp_posts WHERE ID = ';
$sql = $_GET['id'];
$post_data = $wpdb->query( $sql );
```

Bad Developer

```
global $wpdb;
$sql = "SELECT * FROM $wpdb->posts WHERE ID = %d";
$id = (int) $_GET['id'];
$post_data = $wpdb->query( $wpdb->prepare( $sql, $id ) );
```

Better Developer

```
$post_data = get_post( $_GET['id'] );
```

Of Validation and Sanitization

- These are not the same thing
- Whitelist, blacklist, and clean data
- Both need to be done or at least considered
- Know when to use each

Validating

- Comparing data with a set of criteria to determine acceptability
- Reject non-conforming data
- Most important for data input

Sanitizing/Escaping Data

- Manipulating data for a specified use case
    - e.g., convert special characters for a post title when using in the title attribute
- Sanitize for the situation
    - e.g., URLs are escaped differently for different situations

WordPress is an Extreme Open Source Hater

- Given validation/sanitization functions out of the box (formatting.php)
- Use them!
- Do not assume that WordPress functions handle sanitization for you
    - e.g., the_title() handles escaping for general printing, but not for attributes

SCRETS-LATED POSTS

It’s all the buzz

Bad Developer

```php
public function form( $instance ) {
    $number = '';
    if ( isset( $instance['number'] ) )
        $number = $instance['number'];
?>
    <p>
        <label for="<?php echo $this-
>get_field_id( 'number' ); ?>">
            Number of items to show:
        </label>
        <input id="<?php echo $this-
>get_field_id( 'number' ); ?>"
            name="<?php echo $this-
>get_field_name( 'number' ); ?>"
            type="text" value="<?php echo $number; ?>"
            size="3"
        />
    </p>
<?php
}
```

Bad Developer

```
public function update( $new_instance, $old_instance ) {
    $number = '';

    if ( isset( $new_instance['number'] ) )
        $number = $new_instance['number'];

    return array( 'number' => $number );
}
```

Better Developer

```
public function form( $instance ) {
    $number = '';
    if ( isset( $instance['number'] ) )
        $number = $instance['number'];
?>
    <p>
        <label for="<?php echo $this-
>get_field_id( 'number' ); ?>">
            Number of items to show:
        </label>
        <input id="<?php echo $this-
>get_field_id( 'number' ); ?>"
            name="<?php echo $this-
>get_field_name( 'number' ); ?>"
            type="text" value="<?php echo absint( $number ); ?
>"
            size="3" />
    </p>
<?php
}
```

Better Developer

```
public function update( $new_instance, $old_instance ) {
    $number = '';

    if ( isset( $new_instance['number'] ) &&
        in_array( $new_instance['number'], range( 1, 20 ) ) )
        $number = absint( $new_instance['number'] );

    return array( 'number' => $number );
}
```