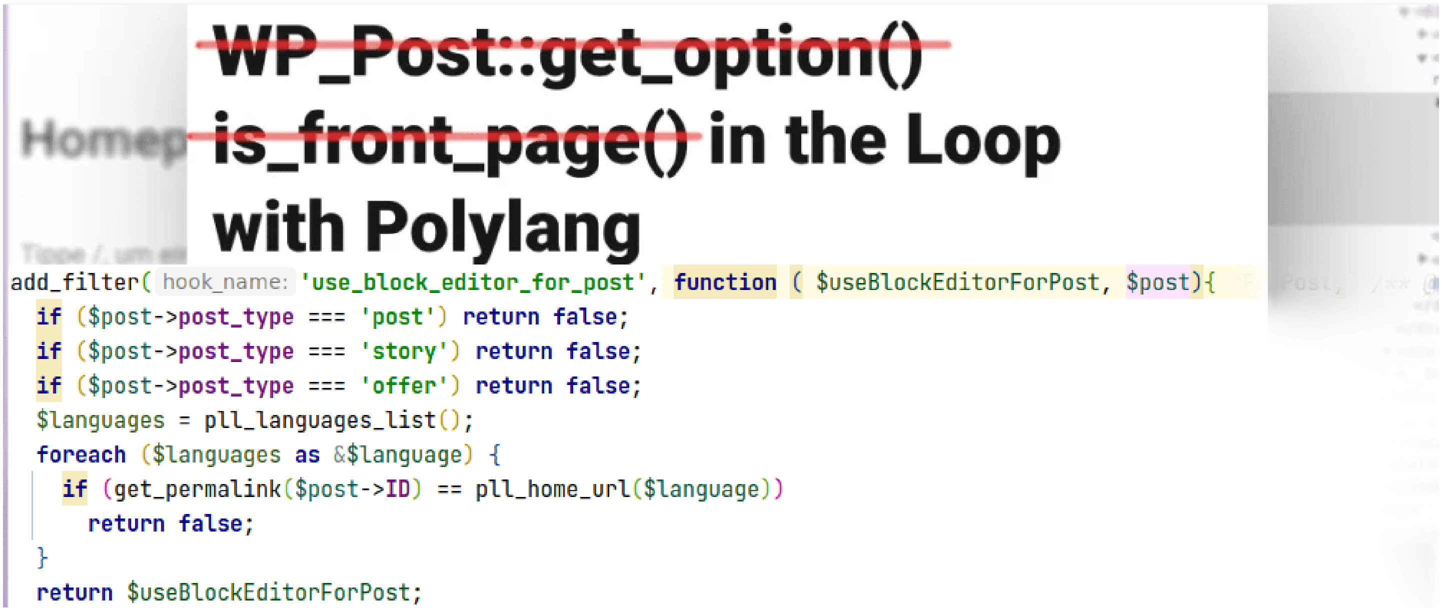


~~WP_Post::get_option()~~
~~is_front_page()~~ in the Loop
with Polylang

```
add_filter( hook_name: 'use_block_editor_for_post', function ( $useBlockEditorForPost, $post){
  if ($post->post_type === 'post') return false;
  if ($post->post_type === 'story') return false;
  if ($post->post_type === 'offer') return false;
  $languages = pll_languages_list();
  foreach ($languages as &$language) {
    if (get_permalink($post->ID) == pll_home_url($language))
      return false;
  }
  return $useBlockEditorForPost;
```

Edit Manage Stats

Ingo Steinke
Posted on Aug 2

❤️ 6 🦄 2 🤯 1 🙌 1 🔥 2

Filter WP Post Frontpages in the Loop with Polylang

#webdev #wordpress #php

Using WordPress as a Developer (10 Part Series)

Did you ever try something like `WP_Post::get_option()->is_front_page()` in the loop with Polylang? No, that's not the correct syntax, but that's what I tried and googled before coming up with a working solution.

Let me show you my initial requirement, different approaches and the reasons why they didn't work. But you can also use the links in the table of contents to jump to the solution directly.

Table of Contents

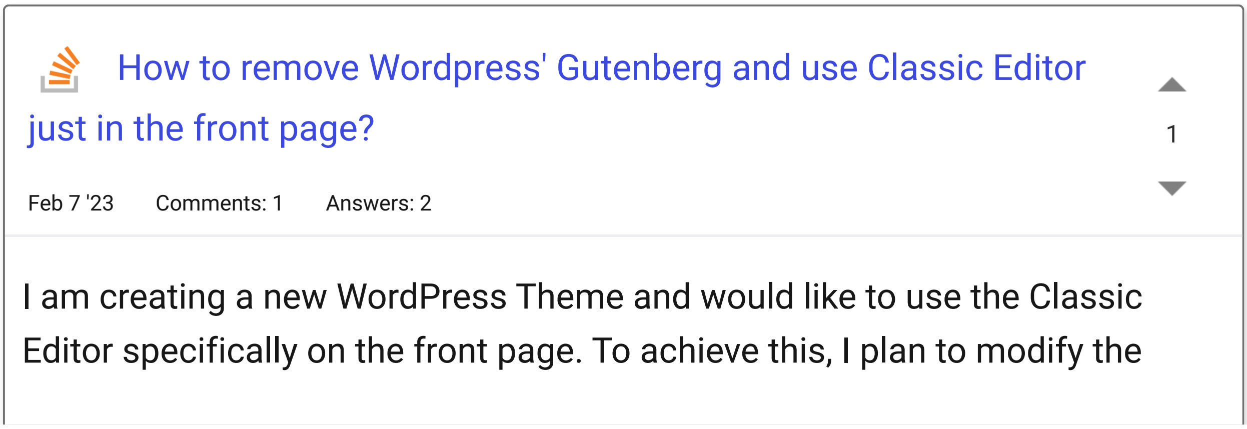
Requirements: multiple Editors, multiple Languages

Requirement: I want to provide the WordPress block editor for editing pages, unless it's the front page. (The front page has a custom design and content using custom fields and post types.)

Additional requirement: There is more than one language, thus more than one frontpage, so we cannot simply compare the current post's `ID` to `get_option( 'page_on_front' ))` in the backend.

Blocking the Block Editor conditionally

You probably also found some questions like this one that proves I am not the first and not the only one to ask.



How to remove Wordpress' Gutenberg and use Classic Editor just in the front page?

▲ 1 ▼

Feb 7 '23 Comments: 1 Answers: 2

I am creating a new WordPress Theme and would like to use the Classic Editor specifically on the front page. To achieve this, I plan to modify the functions.php file with a few lines of code instead of relying on a plugin.

I would like to maintain the Gutenberg editor...

Open Full Question

Using Gutenberg Block Editor depending on Post Types

Based on post or page types, we can add a filter in `functions.php`

```
/* use block editor only for page type "page" */
add_filter('use_block_editor_for_post_type', function (
  $useBlockEditorForPost,
  $post
){
  if ($post->post_type === 'post') return false;
  if ($post->post_type === 'story') return false;
  return $useBlockEditorForPost;
},
  /** @var int $priority */ 10,
  /** @var int $accepted_args */2
);
```

Excluding the Home Page or Front Page?

Home Page vs. Front Page

A default WordPress home page is a page that displays the latest blog posts. We can configure our site to show a static WordPress page instead which can also be a home page then. We can also provide a custom HTML template called `front-page.php` (according to the template hierarchy in the WordPress "Codex" manual).

But should we use `is_home()` or `is_front_page()` to check whether a page is a home page, or what is the correct best practice in my case, to extend my filter function and switch off the block editor when editing the home page.

There is a more general filter called `use_block_editor_for_post` which is not restricted to the post type. Also easy to find out: `is_home()` and `is_front_page()` don't work for the current post inside a loop, at least not without providing a post ID. Without looking up, I would try either `$post->is_front_page()` or `is_front_page($post->ID)` ? So let's see...

```
add_filter('use_block_editor_for_post_type', function (
  $useBlockEditorForPost,
  $post
){
  if ($post->post_type === 'post') return false;
  if ($post->post_type === 'story') return false;
  if (is_front_page($post->ID)) return false;
```

Oh well, okay, after trying and reading about 10 or 20 obscure Q&A discussions on StackExchange, I find another approach:

get_option('page_on_front')

```
  if ($post->ID &&
      $post->post_ID > 0 &&
      $post->ID === (int) get_option( 'page_on_front' )
     ) return false;
```

Even if that works, how is it supposed to return more than one post ID when used on an admin page? Remembering my requirements: in a localized multi-language scenario, I have up to as many front pages as I have languages.

The Loop vs. the current Query

Many built-in WordPress core functions like `is_home()` or `is_front_page()` assume we want to check for the current post's (or page's) properties, implicitly available in a global context or inside a loop (or rather [THE Loop]) after calling `the_post` or how it's called. There are some "typical" getter function pairs like `the_title()` and `get_the_title($post_ID)` where the former prints the current posts's title to stdout, whereas only the latter accepts an optional `ID` and can be assigned to a variable, but many boolean built-ins have no official counterparts that accept an ID and can be used inside a custom loop.

Conclusion: how to check if the current page (inside a multilingual loop) is the front page in WordPress

Or, more technically, speaking, how to filter `WP_Post` by property `is_front_page()` in the loop with Polylang, or, as we will see, how to do this without using `is_front_page()` at all.

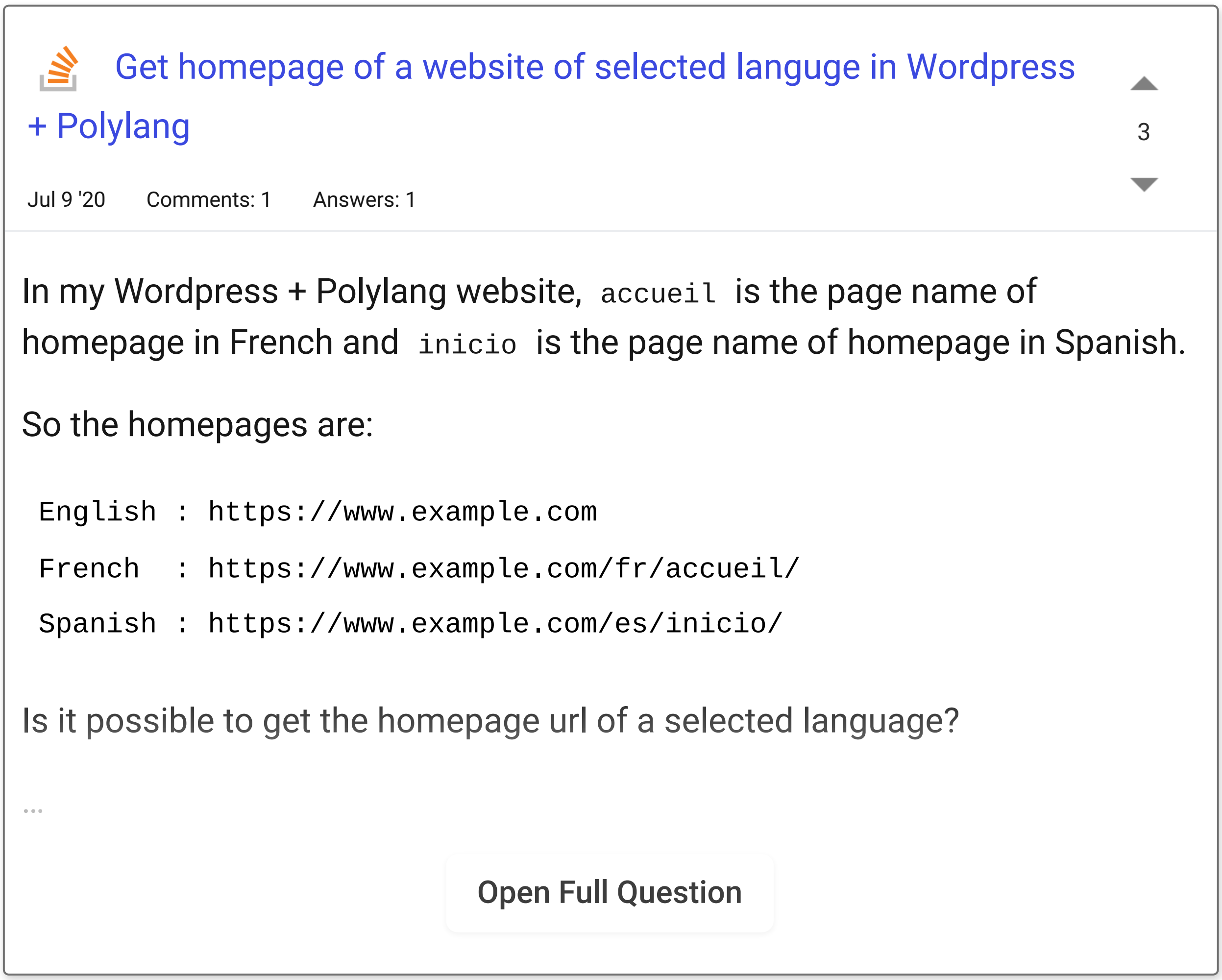
Based on [another StackOverflow answer], we can iterate over all languages and compare each localized front page with the current page.

Naive, pragmatic, and careless about performance (in the backend at least, or if it's cached)

Naive and pragmatic, I know that there are nothing but two languages on my website, so let's start with a hard-coded array and focus on the new function and find out if it works at all.

I don't even care about performance, because it's in the admin backend and if it wasn't, I wouldn't care either as I make sure to cache the results of overengineered processing anyway so that my page visitors get a static HTML webpage from the cache quickly, unless we actually need something uncacheable like personalized data for logged in users.

get homepage of a website of selected language in Wordpress + Polylang



Get homepage of a website of selected language in Wordpress + Polylang

3

Jul 9 '20 Comments: 1 Answers: 1

In my Wordpress + Polylang website, `accueil` is the page name of homepage in French and `inicio` is the page name of homepage in Spanish.

So the homepages are:

```
English : https://www.example.com
French  : https://www.example.com/fr/accueil/
Spanish : https://www.example.com/es/inicio/
```

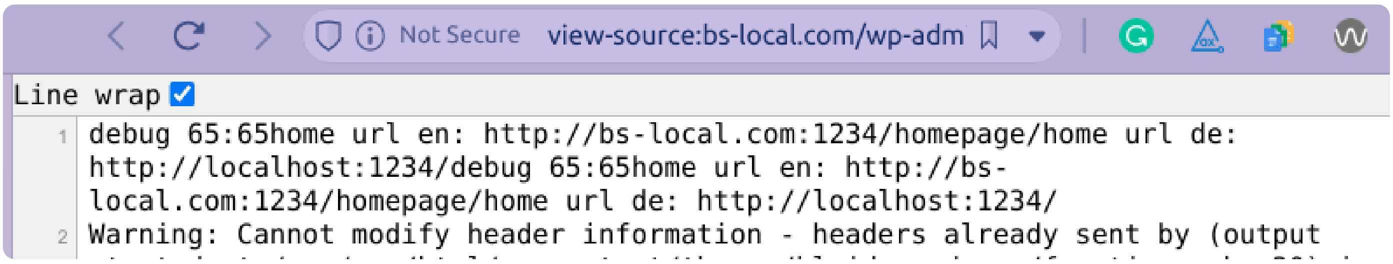
Is it possible to get the homepage url of a selected language?

...

When copying code from the accepted answer, I already sense that I'm but half-way there, as it isn't called `pll_home_page` but `pll_home_url`. Let's just echo the values and see where to go from there.

```
// (string) pll_home_url(  $lang = '' );
echo 'home url en: ' . pll_home_url( 'en' );
echo 'home url de: ' . pll_home_url( 'de' );
```

If we don't `return false` in our filter function, Gutenberg's JavaScript SPA will hide our echo message, but we can still find it when inspecting the page source:



Line wrap ☑

```
debug 65:65home url en: http://bs-local.com:1234/homepage/home url de:
http://localhost:1234/debug 65:65home url en: http://bs-
local.com:1234/homepage/home url de: http://localhost:1234/
Warning: Cannot modify header information - headers already sent by (output
```

Here they are:

```
home url en: http://bs-local.com:1234/homepage/
home url de: http://localhost:1234/
```

This looks somewhat unexpected but it's correct. As German (`'de'`) has been set as the default language and my permalink settings omit anything else but the page title, the English homepage url does not contain ab `/en/` slug.

So can we possibly compare the current page's url (or, technically speaking, [the current post's permalink](#)) with the known language variations?

```
  echo ' ;post permalink: ' . get_permalink($post->ID);
```

and when it matches, return false to disable the block editor.

```
  $languages = array( 'de', 'en' );
  foreach ($languages as &$language) {
    if (get_permalink($post->ID) == pll_home_url($language))
      return false;
  }
```

Finally, let's use the actual languages so our theme still works when the site owner wants to add a third language. We can use Polylang's [`pll_languages_list`](#) function.

Final Filter Function: use_block_editor_for_post

Here is my final `functions.php` filter function to keep the block editor enabled for pages except for the front page(s) in any language and except for posts and my custom post types:

```
add_filter('use_block_editor_for_post', function (
  /** @var bool */ $useBlockEditorForPost,
  /** @var WP_Post */ $post
){
  if ($post->post_type === 'post') return false;
  if ($post->post_type === 'story') return false;
  $languages = pll_languages_list();
  foreach ($languages as &$language) {
    if (get_permalink($post->ID) == pll_home_url($language))
      return false;
  }
  return $useBlockEditorForPost;
},
  /** @var int $priority */ 10,
  /** @var int $accepted_args */2
);
```

This will block the block editor for the front pages but we can still use it on any other page.

Customizing the Classic Editor's Styles and Toolbars

We can also customize the classic (tinyMce) editor, add editor styles (`add_editor_style` in an `admin_init` action) and class names (in JavaSript: `acf.add_filter( 'wysiwyg_tinymce_settings'`) and customize the classic editor toolbars (in PHP: `add_filter('acf/fields/wysiwyg/toolbars'`) to make it both easy and safe for website owners to edit their site content when using a hybrid WordPress theme (classic + block editing, as I explained in an earlier post about classic themes with block patterns in WordPress in 2023.

Classic Themes with Block Patterns in WordPress

Ingo Steinke • May 23

#wordpress #webdev #tutorial

Using WordPress as a Developer (10 Part Series)