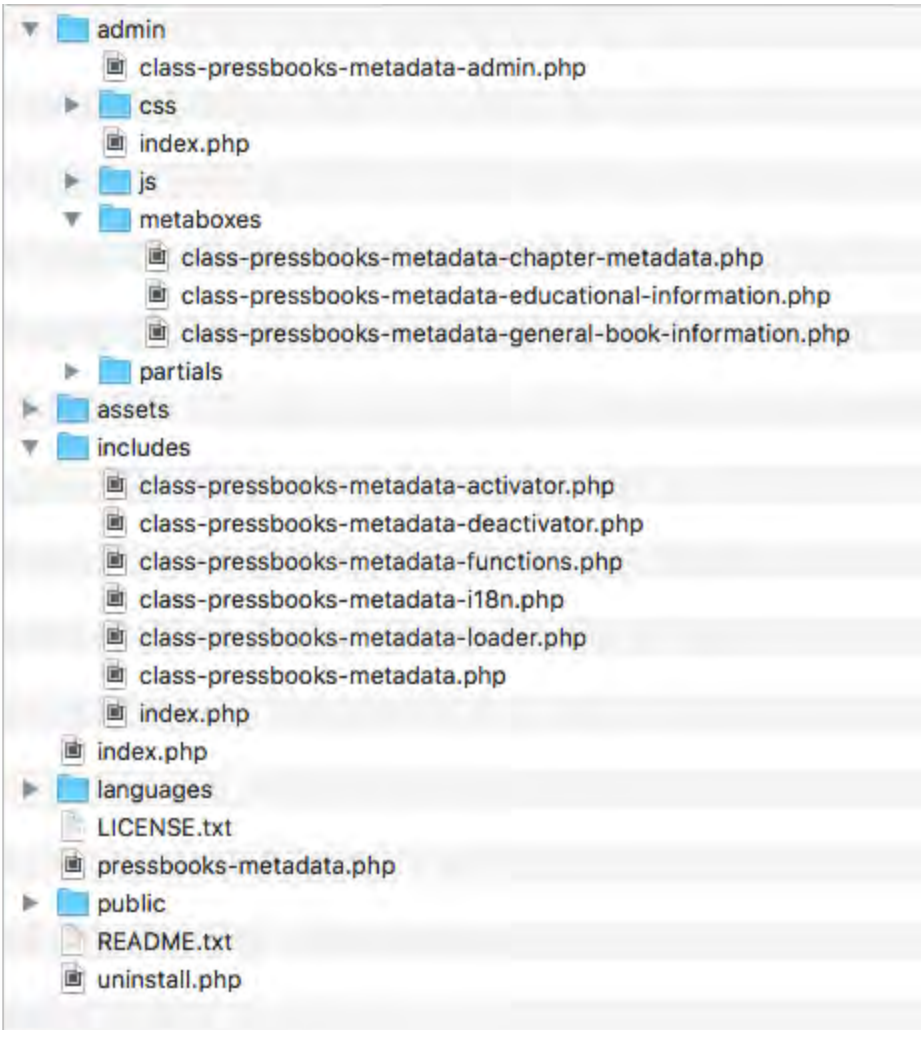
Metadata Plugin Documentation

(version 0.8)

What's the purpose of the plugin?

Pressbooks has by default a lot of metadata fields that can help Google and other search engines show our book more efficiently. The problem comes if your book is for educational purposes. Pressbooks-metadata, extends the functionality of Pressbooks and gives you the flexibility to add more metadata in your books, taking advantage of the LRMI schema markup.

Structure of the plugin



These are the files of the plugin. It uses the wordpress plugin boilerplate framework for better organization of the files and the code. Now we will explain the important files.

admin/metaboxes/

In the `admin/metaboxes` folder, we have 3 files. One for each metabox that we add information inside.

These metaboxes are:

- General Book Information (from Pressbooks)
- Educational Information (new)
- Chapter Metadata (from Pressbooks)

Let's see one of them.

```php
/**
* The function which produces the metafields to the existing
* metabox General Book Information of Pressbooks
*
* @since 0.8
*/
public function add_metadata(){

    //—— Genral Book Informaton metabox ——//

    //———————— metafields ———————— //

    // Illustrator
    x_add_metadata_field(   'pb_illustrator', 'metadata', array(
        'group'         => 'general-book-information',
        'label'         => 'Illustrator',
    ) );

    // Book Edition
    x_add_metadata_field(   'pb_edition', 'metadata', array(
        'group'         => 'general-book-information',
        'label'         => 'Book Edition',
        'description'   => 'The edition of the book. Example: First Edition or 1 or 1.0.0',
    ) );


}
```

This one is adding fields to a pre-existing metabox of pressbooks.
Let's decode the parameters of the function `x_add_metadata_field`

1. **Slug of the field** => This is the name that we will use to refer to this field. It is important for it to start with the prefix ‘pb_’, otherwise it will not work (This is because of pressbooks).
2. **Field’s position** => It is ‘metadata’ for the Book Info post meta and ‘chapter’ for the chapter’s post meta.
3. **Info Array** => In this array you can add information for the field like ‘group’, ‘label’, ‘description’ etc. You can see all the things you can do in the **custom-metadata** plugin. There is a file with examples called: `custom_metadata_examples.php`. The important ones are the ‘group’ and ‘label’. The group can be `‘general-book-information’, ‘copyright’, ‘about-the-book’, ‘additional-catalogue-information’, ‘chapter-metadata’,` according to which metabox you want to add the field to. If you make your own metabox, you can add your own group-slug.

admin / class-pressbooks-metadata-admin.php

Here are the important functions of the plugin. These are:

1. pmdt_init
2. pmdt_header_function
3. pmdt_footer_function

pmdt_init

This is a function that checks if we have pressbooks installed, and if yes the minimum requirements of it.

pmdt_header_function / pmdt_footer_function

This one adds the following code to the header of the website.

```php
/**
 * Used in the header of our site
 *
 * We can create a new Structured Data Type by adding a new type here, Check the link for an example
 * https://search.google.com/structured-data/testing-tool/u/0/#url=pressbooks.com
 * @since    0.6
 */
public function pmdt_header_function() {

    global $post;

    $pmdt_GS = new Pressbooks_Metadata_Functions();

    if ( is_home() ) {
        echo $pmdt_GS->pmdt_get_Root_level_metatags();
    } elseif ( is_front_page() ) {
        echo $pmdt_GS->pmdt_get_GoogleScolar_metatags();
    }
}
```

`If ( is_home() )` means if we are in the Root level (the catalog of the books). and `if ( is_front_page() )` means if we are in the Site level (the level of the book).

We print different code in the footer and the header because it was interrupting with the code of Pressbooks.

The reason we put the Root Page code in the header is here https://github.com/Books4Languages/pressbooks-metadata/issues/6

The reason we put all the other code in the footer is because if we want to extends Pressbook's Book and WebPage types, our code must be after their code.

includes/class-pressbooks-metadata-functions.php

In this file we have all the functions that actually produce the code. All of them work in the same way. We can see for example:

```php
/**
 * A function to retrieve the metatags we need for the Site level
 * of the website. In Pressbooks this is the book.
 *
 * @since 0.8
 */
public function pmdt_get_Site_level_metatags(){
    // array of the items that we need for the Book
    $book_data = array(
    // Here are the fields from General Book Information metabox.
        'illustrator'           => 'pb_illustrator',
        'bookEdition'           => 'pb_edition',
    //  Here are the fields from Educational Information metabox.
        'provider'              => 'pb_provider',
        'typicalAgeRange'       => 'pb_age_range',
        'learningResourceType'  => 'pb_learning_resource_type',
        'interactivityType'     => 'pb_interactivity_type',
        'timeRequired'          => 'pb_time_required',
        'license'               => 'pb_license_url',
        'isBasedOnUrl'          => 'pb_bibliography_url'
    );

    $html  = "<!-- Book additional microtags -->\n";

    $metadata = \Pressbooks\Book::getBookInformation();

    foreach ($book_data as $itemprop => $content){
        if ( isset( $metadata[$content] ) ) {
            //if the schema is timeRequired, we are using a specific format to display it,
            //like the example here: https://schema.org/timeRequired
            if ( 'timeRequired' == $itemprop ) {
                $metadata[ $content ] = 'PT'. $metadata[ $content ].'H';
            }
            $html .= "<meta itemprop = '" . $itemprop . "' content = '" . $metadata[ $content ] . "'>\n";
        }
    }

    return $html;
}
```

On the left side of the array is the itemprop we want to use, and on the right one the field's slug.
If you want to extend it, just add more itemprops and fields in the array.

Then for each property we have, it produces the microtag we need.