Storyblok Integration for Magento 2

Our Magento 2 integration allows developers and digital agencies to create content-rich pages that are easily editable using the Storyblok interface.

Why Storyblok?

Storyblok allows you to manage content through a CMS that is intuitive and easy to use. It offers features like:

- Visual Editor
- Content Types with Blocks
- Custom Fields
- Internationalization Support
- Content Scheduling, and more!

It can be used as an **alternative to Magento Commerce's Page Builder** as it provides all of its features and, combined with our integration module, it allows for a more pleasant developer experience when it comes to creating custom blocks.

Installation

1. Install via composer `composer require medialounge/magento2-storyblok-integration`
2. Run `php bin/magento setup:upgrade`

Getting Started

Before you can start using our module you will need to create an account in Storyblok, you can start by using their free tier but you will definitely want to upgrade to one of their paid plans if you want to use the more powerful features.

After creating your account make sure to create a new space so you can start configuring the module in the Magento admin area.

Configuration

There are only 3 settings you will need to setup after installing the module, you can access them by going to **Stores** → **Configuration** → **Media Lounge** → **Storyblok**:

Enable

Whether the module is enabled or not.

API Key (required)

Your Storyblok API Key, you can get it from your Storyblok space by going to **Settings** → **API-Keys**.

Make sure the "**Access Level**" is set to "**preview**".

Webhook Secret (required)

Random string that we will use to authenticate that requests are coming from Storyblok, we recommend using a 20+ characters alphanumeric string.

After this is set make sure to use the same value in your Storyblok Space by going to **Settings → General → Webhook secret**.

Creating our first page

With the module configured we can start creating content in Storyblok, go to **Content → + Entry** and give it any name you want.

You should now see a preview of the page in your Magento 2 store, for initially it will be empty so let's create our first block.

Creating our first block

A block is an individual piece of content that can be reused throughout the whole site, you can create one from the page preview screen by clicking **+ Add block** and giving it a name.

With our block created you can then define a schema for it which is just one or more fields that will be used to store the block's content. There are multiple field types available by default in Storyblok (text, textarea, wysiwyg, image, etc.) and you can know more about them in their Field Type Documentation.

After adding some custom field types and including the new block in our page we should see a debug message displayed where the block should be, this is a hint from the module to tell us that this is a new block and we have not yet created a template file in our theme to render it so let's go ahead and do that.

Creating the template for our first block

Out of the box **our module doesn't include blocks**, in fact we don't even add any extra CSS or JS to your Magento 2 store! We give you the freedom of customising your content any way you want using the technology stack of your choice.

The debug message above shows all the data that we have available as well as the path where it expects the PHTML template file to be in, so we just need to create a new file in **MediaLounge_Storyblok/templates/story/block-name.phtml** in our custom theme.

Inside this template we have access to the Storyblok fields as part of the block's data so you can access them by using the `$block->getData()` method (or magic methods if you prefer).

MediaLounge_Storyblok/templates/story/block-name.phtml

```
<?php /** @var MediaLounge\Storyblok\Block\Container\Element $block */ ?>

<div>
    <h1><?= $block->getTitle(); ?></h1>
    <img src="<?= $block->getImage()['filename'] ?>" alt="<?= __('New Luma
Collection') ?>">
</div>
```

> *Hint: You can see all the block's available data using `var_dump($block->getData())`*

Helper Methods

The template block uses the `MediaLounge\Storyblok\Block\Container\Element` class which also extends `Magento\Framework\View\Element\Template` so we have access to all the Magento block methods that we are used to.

On top of this we expose a few helper methods to make working with Storyblok content a bit easier:

$block->renderWysiwyg(array $arrContent)

Arguments	Description
$arrContent	Content of "Richtext" field

When using "Richtext" fields this method will ensure that HTML elements are rendered.

Usage:

```
<?php /** @var MediaLounge\Storyblok\Block\Container\Element $block */ ?>

<div>
    <?= $block->renderWysiwyg($block->getDescription()) ?>
</div>
```

$block->transformImage(string $image, string $param = '')

Arguments	Description
$image	Image URL
$param	Transformation parameters for Storyblok Image Service

Storyblok offers an Image Service that allows you to transform image's size, format, quality amongst other things. This method provides a convenient way of interacting with it so we are able to modify images on the fly.

Usage:

```
<?php /** @var MediaLounge\Storyblok\Block\Container\Element $block */ ?>

<div>
    <img width="1280" height="460" src="<?= $block->transformImage($block->getImage()
['filename'], '1280x460/filters:quality(80):format(webp)') ?>" alt="<?= __('New Luma
Collection') ?>">
</div>
```

> *Hint: This can be really powerful when combined with the `srcset` attribute or `<picture>` element!*

Creating nested blocks

Sometimes it is useful to nest blocks inside other blocks, eg: we could have a "Gallery" block with a "Title" and "Description" fields, as well as an "Media" field where we could nest "Image" or "Video" blocks.

Storyblok allows us to infinitely nest blocks and our integration module supports this functionality, all we need to do is use the `$block->getChildHtml()` method in the

location of our template where we want its child blocks to appear.

```
<?php /** @var MediaLounge\Storyblok\Block\Container\Element $block */ ?>

<div>
    <h2><?= $block->getTitle(); ?></h2>
    <p>
        <?= $block->getDescription(); ?>
    </p>

    <?= $block->getChildHtml() ?>
</div>
```

There's nothing else to it, now any child component will be rendered inside this block using its own template.

> *Hint: Depending on the situation you might want to avoid nesting components and instead prefer to render them in the parent template. The child blocks are just an array of data still available in the parent template so it is a valid option to do so.*

Putting it all together

From here on it's just a matter of repeating the same process for all custom blocks and by adding some CSS and JS we can create from simple to very complex layouts using the Storyblok interface.

Publishing Content

When we are ready to make our Storyblok content available in our Magento 2 store we just need to publish it by clicking the **Publish** button in the top right.

Under the hood Storyblok will make a `POST` request to our website and our module will use this to clear the cache for that specific Story ID, this is where the **Webhook Secret** setting is used so only requests coming from Storyblok are allowed to do this.

Make sure to set the following URL in your Storyblok Space under **Settings** → **General** → **Story published & unpublished and Datasource entry saved**:
`http://yourmagento.url/storyblok/cache/clean`

> *This needs to be a publicly accessible URL, if you want to test this in your local environment you can use a service like ngrok to expose your local domain to the world.*

Using Storyblok outside pages

We don't need to use Storyblok just to manage content for a whole page, we can also use it to make specific sections within our pages editable. This can be useful when we need elements like header menus, footer links, product-specific promotions, etc, where we would normally use Static Blocks or Widgets.

First, we need to create a new **Entry** in Storyblok but instead of using the "page" content type you should choose to create a new one that better matches the content that you're trying to create.

> *Take note of the **Slug** value as we will need this later.*

After this Storyblok will open the content type in the preview window as a regular page, this is NOT what we want but let's ignore it. For now, define the schema as you normally would.

Next we want to tell Storyblok what page should be displayed in the preview, you can change this in the sidebar by going to **Config** → **Real Path**. This depends on the kind of content that you want to manage, eg: to display the homepage use `/`, for the cart page use `/checkout/cart` and so on. Save the change and reload the page.

We should now see the **Real Path** in the preview screen but our new block is nowhere to be seen yet. We need to tell Magento where we want our block to be displayed and this is done using regular Layout XML directives:

```
<referenceContainer name="footer">
    <block class="MediaLounge\Storyblok\Block\Container"
name="storyblok.custom.block">
        <arguments>
            <argument name="slug" xsi:type="string">our-block-slug</argument>
        </arguments>
    </block>
</referenceContainer>
```

As long as we use `MediaLounge\Storyblok\Block\Container` as the block `class` and pass the Storyblok `slug` as an argument we can display custom blocks anywhere we want, in this example we chose to display it inside the `footer` container.

Finally we just repeat the same process as before:

1. Create the PHTML template
2. Style content
3. Publish

Custom Field Types

One of the most powerful features of Storyblok is that it lets us create custom field types to enhance the content editing experience. We can create our own website-specific fields or reuse them across multiple projects, they can also be published on the Storyblok Marketplace so they are available to anyone using the platform.

SEO

The "meta-fields" custom field is an example of an extra block that can be used in our Storyblok Space and our Magento 2 module provides support for it out of the box.

Configuration

We can add our SEO field type as part of our page's schema, make sure to select "Plugin" under the Field **Type** and "meta-fields" as the **Custom Type**.

Now the page's title and meta description will be editable from Storyblok.

Sitemap

Storyblok pages will be added to the default Magento sitemap generator. You can change the *Frequency* and *Priority* under **Store** → **Configuration** → **Catalog** → **XML Sitemap** → **Storyblok Options**.

Custom Field Integrations

With some field types we can allow content from Magento to be editable in Storyblok, this can be useful for example when you want to display a list of featured products. The product data is stored in Magento but if we want to specify what products to show we can use a "text" field to allow for a comma-separated list of SKUs to be entered, then in our template we can use a `Helper` or `ViewModels` to parse this value and only show the products that we need.

> *Hint: The same concept can be used to display any kind of data, eg: customers, orders, social network posts, etc.*

However we can use custom fields to improve this type of content, for the example above we could create a new custom field plugin to query the Magento Products API so we can search against it.

Storyblok gives us almost unlimited freedom over the look and feel of these fields so we can customise them any way we want, with our integration module we hope you're able to build great content-rich experiences in your Magento 2 stores!