Advanced customization

Templates, stylesheets, translations, and more.

Overriding templates

Analytics

You can add analytics (e.g. Google Analytics) to all your pages by overriding the `analytics` block of the base template. In the `templates/` folder of your project, create a file called, `base.mhtml` with the following content:

```
<%inherit file="base/default.mhtml" />
<%block name="analytics">
  <!-- YOUR ANALYTICS JAVASCRIPT CODE GOES HERE -->
</%block>
```

Table of contents

If you want to change the appearance or functionality of the table of contents on each page, you can create a new template file in your project `templates/` directory, called `base/comp/toc.mc`. The easiest way is to copy the original version from the theme to the correct place in your template directory and modify it.

Other modifications

As the above examples illustrate, the templates for the WDocs theme are quite modular and easy to override or extend. Take some time to examine them to determine what the best way of achieving your desired result would be.

Stylesheets and Javascript

There are basically two ways of changing the CSS and Javascript that comes with the theme: overriding them by providing replacement versions, or extending them by adding other CSS or JS files and then adding `site.extra_css` or `site.extra_javasript` to your configuration file.

If you go the first route, create new files named `css/wdocs.css` and `js/wdocs.js` inside your `static` directory. (For the CSS file, you could also create a new SCSS file in `assets/scss/wdocs.scss` to achieve the same result.)

Automatic content filtering

Let's say you want to write your Markdown documents with internal links that reference the markdown documents themselves rather than point to the actual URLs that the generated HTML will have. In other words, you would have source documents with content like this:

```
Further information about this will be found in
[the details document](details.md).
```

Which would lead to HTML like this:

```
<p>
  Further information about this will be found in
  <a href="details.md">the details document</a>.
</p>
```

when you actually want

```
<p>
  Further information about this will be found in
  <a href="details/">the details document</a>.
</p>
```

There are several ways of achieving this, but one quick and simple one is to use the postprocessing functionality of wmk. Create a file called `py/wmk_autoload.py` in your project directory, containing something like this:

```
import re

def mdlinks(s, **kw):
    """
    Transform links to markdown (.md) files to ordinary links.
    """
    s = re.sub(r' href="index\.md', r' href="./', s)
    s = re.sub(r' href="([^"#]*)/index\.md(["#])', r' href="\1/\2', s)
    s = re.sub(r' href="([^"#]+)\.md(["#])', r' href="\1/\2', s)
    return s

autoload = {
    'mdlinks': mdlinks,
}
```

Then add the following to `content/index.yaml` to ensure that it applies to all pages that do not explicitly set `POSTPROCESS` themselves:

```
POSTPROCESS:
    - mdlinks
```

New translations

Let's say you want your site to be in a language different from the default English and that this language is not among those already provided by the theme. Let's take Esperanto (`eo`) as an example for the sake of concreteness. In this case you do the following:

1. Copy the file `wmk.pot` from the theme directory to the file `data/locales/eo/LC_MESSAGES/wmk.po` in your project (obviously you must create the folder as well).
2. Translate the strings in `wmk.po` in your favourite editor.
3. Inside the `LC_MESSAGES` subdirectory, run the command `msgfmt -o wmk.mo wmk.po`.
4. Set the value of `site.lang` to `eo`.
5. Recompile your website.

Your webiste will now be in Esperanto!