invenio-theme Documentation

Release 1.2.0

Invenio Collaboration

May 12, 2020

Contents

Invenio standard theme.

Features:

- Default templates for displaying the page cover, settings, admin settings, breadcrumbs.
- Different templates for error codes.
- Jinja2 macro for displaying flashed messages.

Further documentation available at https://invenio-theme.readthedocs.io/

CHAPTER 1

User's Guide

This part of the documentation will show you how to get started in using Invenio-Theme.

1.1 Installation

Invenio-Theme is on PyPI so all you need is:

```
$ pip install invenio-theme
```

1.2 Configuration

Configuration for Invenio-Theme.

`invenio_theme.config.`**`ADMIN_BASE_TEMPLATE = 'invenio_theme/page_admin.html'`**
Base template for the administration interface.

The template changes the administration interface from using a standard Bootstrap interface to using AdminLTE 2.

The variable is defined in Invenio-Admin which will use the value defined here if Invenio-Theme is installed.

`invenio_theme.config.`**`BASE_TEMPLATE = 'invenio_theme/page.html'`**
Base template for user facing pages.

The template provides a basic skeleton which takes care of loading assets, embedding header metadata and define basic template blocks. All other user facing templates usually extends from this template and thus changing this template allows to change design and layout of Invenio.

`invenio_theme.config.`**`COVER_TEMPLATE = 'invenio_theme/page_cover.html'`**
Cover page template normally used e.g. for login and sign up pages.

`invenio_theme.config.`**`SETTINGS_TEMPLATE = 'invenio_theme/page_settings.html'`**
Settings page template used for e.g. display user settings views.

invenio_theme.config.**THEME_401_TEMPLATE = 'invenio_theme/401.html'**
The template used for 401 Unauthorized errors.

invenio_theme.config.**THEME_403_TEMPLATE = 'invenio_theme/403.html'**
The template used for 403 Forbidden errors.

invenio_theme.config.**THEME_404_TEMPLATE = 'invenio_theme/404.html'**
The template used for 404 Not Found errors.

invenio_theme.config.**THEME_429_TEMPLATE = 'invenio_theme/429.html'**
The template used for 429 Too Many Requests errors.

invenio_theme.config.**THEME_500_TEMPLATE = 'invenio_theme/500.html'**
The template used for 500 Internal Server Error errors.

invenio_theme.config.**THEME_BASE_TEMPLATE = None**
Template which all templates in Invenio-Theme all extends from.

Defaults to value of *BASE_TEMPLATE*.

invenio_theme.config.**THEME_BREADCRUMB_ROOT_ENDPOINT = ''**
The endpoint for the Home view in the breadcrumbs.

invenio_theme.config.**THEME_COVER_TEMPLATE = None**
Template which all cover templates in Invenio-Theme all extends from.

Defaults to value of *COVER_TEMPLATE*.

invenio_theme.config.**THEME_ERROR_TEMPLATE = 'invenio_theme/page_error.html'**
Base template for error pages.

invenio_theme.config.**THEME_FOOTER_TEMPLATE = 'invenio_theme/footer.html'**
Footer template which is normally included in *BASE_TEMPLATE*.

invenio_theme.config.**THEME_FRONTPAGE = False**
Enable or disable basic frontpage view.

invenio_theme.config.**THEME_FRONTPAGE_TEMPLATE = 'invenio_theme/frontpage.html'**
Template for front page.

invenio_theme.config.**THEME_FRONTPAGE_TITLE = 'Invenio'**
The title shown on the fronpage.

invenio_theme.config.**THEME_GOOGLE_SITE_VERIFICATION = []**
List of Google Site Verification tokens to be used.

This adds the Google Site Verfication into the meta tags of all pages.

invenio_theme.config.**THEME_HEADER_LOGIN_TEMPLATE = 'invenio_theme/header_login.html'**
Header login template, included in *THEME_HEADER_TEMPLATE*.

invenio_theme.config.**THEME_HEADER_TEMPLATE = 'invenio_theme/header.html'**
Header template which is normally included in *BASE_TEMPLATE*.

invenio_theme.config.**THEME_JAVASCRIPT_TEMPLATE = 'invenio_theme/javascript.html'**
Javascript assets template, normally included in *BASE_TEMPLATE*.

The default template just includes the Invenio-Theme JavaScript bundle. Set a new template if you would like to customize which JavaScript assets are included on all pages.

invenio_theme.config.**THEME_LOGO = 'images/invenio-white.svg'**
The logo to be used on the header and on the cover.

invenio_theme.config.**THEME_LOGO_ADMIN = 'images/invenio-white.svg'**
The logo to be used on the admin views header.

invenio_theme.config.**THEME_SEARCHBAR = True**
Enable or disable the header search bar.

invenio_theme.config.**THEME_SEARCH_ENDPOINT = '/search'**
The endpoint for the search bar.

invenio_theme.config.**THEME_SETTINGS_TEMPLATE = None**
Template which all settings templates in Invenio-Theme all extends from.

Defaults to value of *SETTINGS_TEMPLATE*.

invenio_theme.config.**THEME_SITENAME = 'Invenio'**
The name of the site to be used on the header and as a title.

invenio_theme.config.**THEME_TRACKINGCODE_TEMPLATE = 'invenio_theme/trackingcode.html'**
Template for including a tracking code for web analytics.

The default template does not include any tracking code.

1.3 Usage

Standard Bootstrap based theme for Invenio.

Invenio-Theme mainly consits of:

- *Templates and static files* - for integration with Bootstrap HTML, CSS and JS framework.
- *Error handlers* - for showing user-friendly 401, 402, 404, and 500 error pages.
- *Menu and breadcrumbs* - for basic site navigation using Flask-Menu and Flask-Breadcrumbs.

1.3.1 Initialization

First create a Flask application and initialize *InvenioTheme* extension.

```
>>> from flask import Flask, render_template_string
>>> from invenio_theme import InvenioTheme
>>> from flask_menu import register_menu
>>> app = Flask('myapp')
>>> theme = InvenioTheme(app)
```

Note: There is a new blueprint registered during initialization in order to enable loading of templates and static files from Invenio-Theme. The blueprint does not have any views registered.

In order for the following examples to work, you also need to load the Invenio-I18N and Invenio-Assets:

```
>>> from invenio_i18n import InvenioI18N
>>> from invenio_assets import InvenioAssets
>>> i18n = InvenioI18N(app)
>>> assets = InvenioAssets(app)
```

1.3.2 Basic customizations

Invenio-Theme makes it easy to perform some simple customizations such as changing logo.

Logo and name

The easiest customization you can do is changing the logo and name of your Invenio instance. Simply set the *invenio_theme.config.THEME_LOGO* and *invenio_theme.config.THEME_SITENAME* configuration variables:

```
>>> app.config.update({
... 'THEME_LOGO': 'images/invenio-black.svg',
... 'THEME_SITENAME': 'Awesome Site',
... })
```

The logo path is relative to your static folders. Also, if you don't want a logo displayed, you can simply set the value to `None`.

Search bar

If you don't like having the search bar available in the header, you can modify the behaviour by setting *invenio_theme.config.THEME_SEARCHBAR* to `False`:

```
>>> app.config.update({
... 'THEME_SEARCHBAR': False,
... })
```

Analytics tracking code

If you need to include an analyics tracking code in your HTML pages (e.g. to track analytics with Piwik or Google Analytics) you should set the *invenio_theme.config.THEME_JAVASCRIPT_TEMPLATE* variable to a template which includes the tracking code:

```
>>> app.config.update({
... 'THEME_JAVASCRIPT_TEMPLATE': 'invenio_theme/trackingcode.html',
... })
```

Note that `invenio_theme/trackingcode.html` is empty so you have to provide your own template with the tracking code.

Google Site Verification

Google Webmasters Tools allows you to inspect how Google is indexing your site. In order to see your site, you must provide proof of ownership which is done by e.g. including a token generated by Google in the meta-tags of your website. You can easily do this with Invenio-Theme theme. Simply update the *invenio_theme.config.THEME_GOOGLE_SITE_VERIFICATION* configuration variable with the list of tokens provided by Google:

```
>>> app.config.update({
... 'THEME_GOOGLE_SITE_VERIFICATION': ['...token...'],
... })
```

1.3.3 Templates

Templates in Invenio-Theme falls in two categories:

- **Page templates** such as the base, cover and settings templates which defines a template for an entire page.

- **Section templates** such as the header and footer templates which is included as part of a page templates.

Most simple visual changes can be done by just overriding a section template. Larger visual changes will likely require changes to one or more of the page templates.

Overriding templates

Most of the time all you need is to customize a small part of an existing template. This is best achieved by extending the Invenio-Theme template and overriding the template block you would like to customize.

For instance, say you wanted to include two logo files instead of only one in the header, then you could extend the header template and override the `brand` block like this:

```
{# mytheme/header.html #}
{% extends "invenio_theme/header.html" %}
{% block brand %} ... {% endblock brand %}
```

In order to make Invenio use this new header template you must set it in the configuration (assuming the template was saved in a template folder under `mytheme/header.html`):

```
>>> app.config.update({
... 'THEME_HEADER_TEMPLATE': 'mytheme/header.html',
... })
```

Warning: In general, if you override a template you should provide the same template blocks as the template you override. This is because other Invenio modules may rely on these template blocks,

Header section template

The header template (`invenio_theme/header.html`) is reponsible for rendering the navigation bar (including logo, search bar, menu and login/sign up buttons), flash messages and the breadcrumbs.

Change the template by updating *invenio_theme.config.THEME_HEADER_TEMPLATE*.

The template provide a number of blocks which can be overridden. The most important ones are (please see the template file for full details):

- `navbar` - Displays the navigation bar and inlcude other template blocks such as `navbar_header`, `navbar_nav` and `navbar_right`.
- `flashmessages` - Displays small notification messages such as "Successfully logged in."
- `breadcrumbs` - Displays the breadcrumbs.

Header login section template

The header login template (`invenio_theme/header_login.html`) is responsible for rendering the login/sign up buttons as well as a user menu if a user is logged in. It is included by the header template and does not define any template blocks.

Change the template by updating *invenio_theme.config.THEME_HEADER_LOGIN_TEMPLATE*

Footer section template

The footer template (`invenio_theme/footer.html`) is primarily reponsible for rendering the language selector and does not contain any template blocks.

Change the template by updating *`invenio_theme.config.THEME_FOOTER_TEMPLATE`*

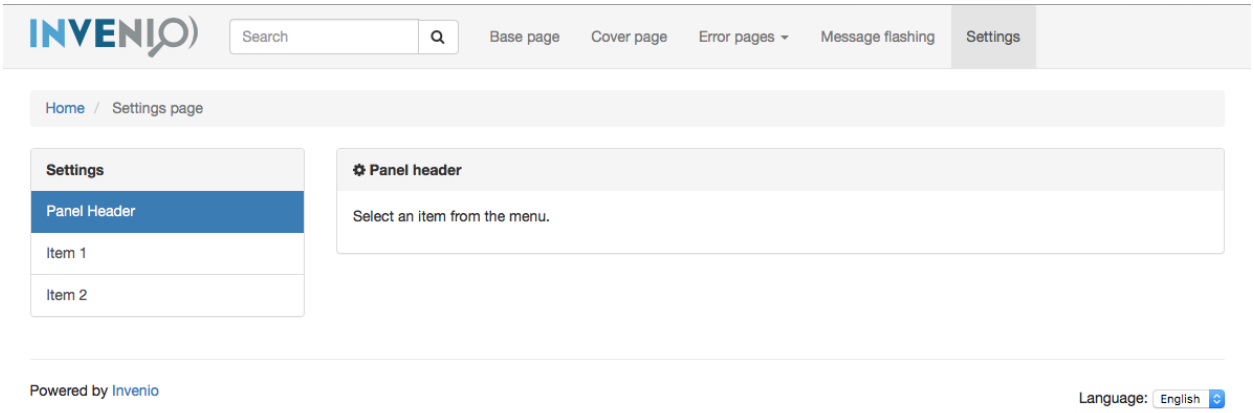
Javascript section template

If all you need to customize is to load some additional JavaScript, you can just override the JavaScript template (`invenio_theme/javascript.html`). This template is responsible for including site-wide JavaScript modules.

Change the template by updating *`invenio_theme.config.THEME_JAVASCRIPT_TEMPLATE`*

Settings page template

The settings template (`invenio_theme/page_settings.html`) is responsible for rendering user settings pages such as the change password dialog.



Change the template by updating *`invenio_theme.config.SETTINGS_TEMPLATE`*

The template provide a number of blocks which can be overridden. The most important ones are (please see the template file for full details):

- `settings_menu` - Renders the left-hand menu and include another template block `settings_menu_item` which renders an individual menu item.
- `settings_content` - Renders the right-hand content panel and include three template blocks for `settings_content_title`, `settings_body` and `settings_form`.

Cover page template

The cover template (`invenio_theme/page_cover.html`) has a simplified layout and is used for e.g. displaying the login dialog and sign-up page.

Change the template by updating *invenio_theme.config.COVER_TEMPLATE*

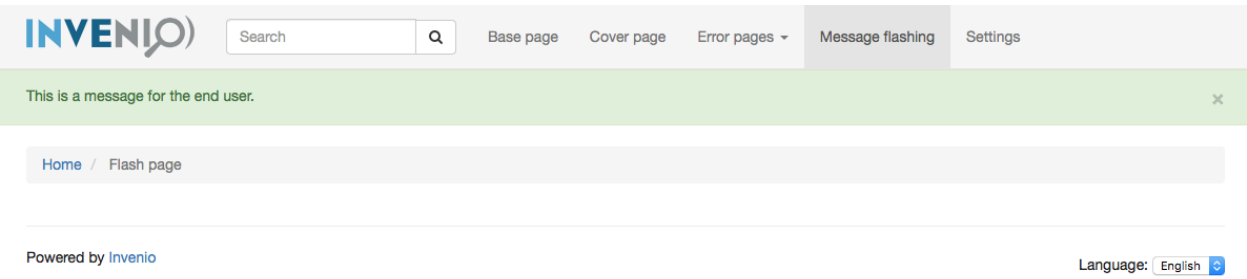
The template provide a number of blocks which can be overridden. The most important ones are (please see the template file for full details):

- `page_header` - Includes e.g. the logo and branding.
- `page_body` - Renders the page body which includes a panel. The panel can be customized via the template blocks `panel` and `panel_content` and by setting the template variable `panel_title`.
- `page_footer` - Displayed right after the page body.

Base page template

The base template (`invenio_theme/page.html`) is responsible for providing a basic page skeleton with template blocks that can be overriden by templates extending the base template.

Warning: If you override this template it is very important that you provide the same template blocks as Invenio is relying on the template block API provided in this template.



Change the template by updating *invenio_theme.config.BASE_TEMPLATE*

The template provide a number of blocks which can be overridden. The most important ones are (please see the template file for full details):

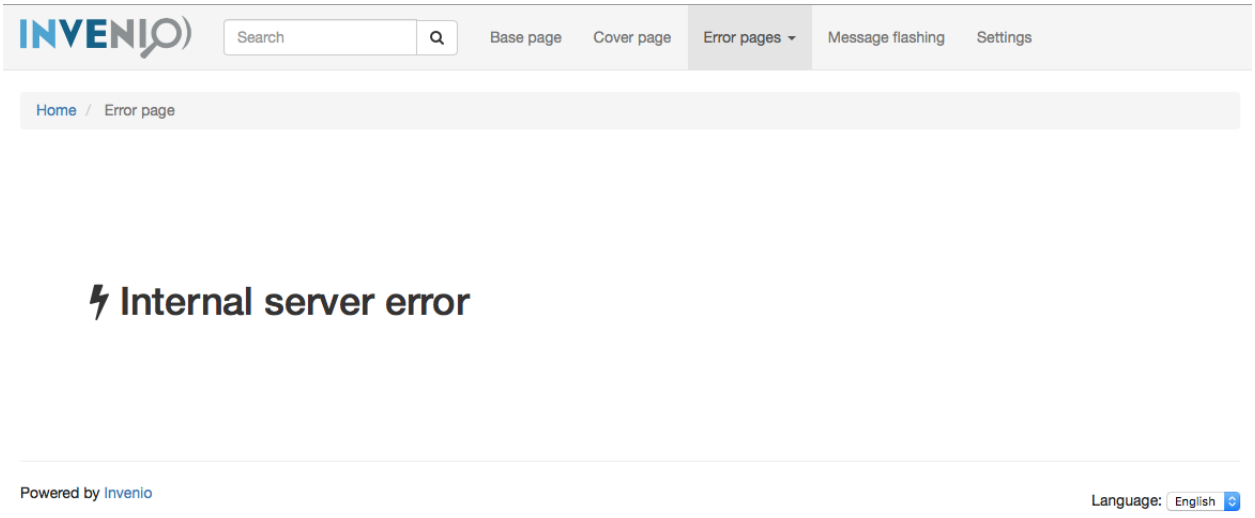
- `header` - Empty block you can override to add extra content in the head-tag.
- `css` - Block containing the link to stylesheets. By default just the Invenio-Theme CSS is included. Placed in the head-tag.
- `javascript` - Block containg the JavaScript tags. By default just Invenio-Theme JS is included. Placed in the very end of the page.
- `page_header` - Block which includes the header section template.
- `page_body` - Block containing the page content. Normally when using the template, this is the only block you need to override.
- `page_footer` - Block which includes the footer section template.

When using the template, a couple of template variables can be set t

- `title` - Sets the header title-tag.
- `description` - Sets the description meta-tag.
- `keywords` - Sets the keywords meta-tag.

Error page templates

The error template (`invenio_theme/page_error.html`) is used as basis for error pages such as unauthorized access, internal server error etc.



Change the template by updating *invenio_theme.config.THEME_ERROR_TEMPLATE*. If you wish to just override individual error pages you can set the variables:

- *invenio_theme.config.THEME_401_TEMPLATE*
- *invenio_theme.config.THEME_403_TEMPLATE*
- *invenio_theme.config.THEME_404_TEMPLATE*
- *invenio_theme.config.THEME_500_TEMPLATE*

Front page

The front page (`invenio_theme/frontpage.html`) is the homepage of the application. By default is `disabled`.

Enable the front page view by setting `THEME_FRONTPAGE` to `True` *invenio_theme.config.THEME_FRONTPAGE*

Change the template by updating: *invenio_theme.config.THEME_FRONTPAGE_TEMPLATE*

The template provide a number of blocks which can be overridden. The most important ones are (please see the template file for full details):

- `page_header` - Includes e.g. the logo and branding.
- `page_body` - Renders the page body which includes a panel. The panel can be customized via the template blocks `panel` and `panel_content` and by setting the template variable `panel_title`.

1.3.4 Developing Invenio modules

Note: This section is only relevant if you develop Invenio modules that need to be able to change style depending on the currently installed theme.

This section describes a set of patterns for Invenio modules that allows the module to be themeable without hard depdencies on Invenio-Theme, which is important as otherwise users of your module cannot change the default styling.

Base template pattern

The base template pattern allows your module to extend from the currently installed theme's base template.

First, create a dummy base template in your module. The dummy base template should define the same blocks as you plan to use from the Invenio-Theme template. In most cases this means just providing the `page_body` block:

```
{# invenio_foo/base.html #}
<html><body>
{% block page_body %}{% endbxlock %}
</body></html>
```

Note: Template paths need to be globally unique, so normally you should put your templates in a subfolder named according to your module. For instance if you are developing Invenio-Foo, then put your base template in `invenio_foo/base.html`.

Templates you develop in your module should extend from config variable instead of directly from the base template:

```
{# invenio_foo/myview.html #}
{% extends config.FOO_BASE_TEMPLATE %}
{% block page_body %} ... {% endblock %}
```

The configuration variable you then set during your Flask extension initialization, according to this pattern:

```
# invenio_foo/config.py
FOO_BASE_TEMPLATE = 'invenio_foo/base.html'
```

```
# invenio_foo/ext.py
from . import config

class InvenioFoo(object):
    # ...
    def init_config(app):
        # Set FOO_BASE_TEMPLATE to the theme's
        # base template if theme is installed
        app.config.setdefault(
            'FOO_BASE_TEMPLATE',
            app.config.get('BASE_TEMPLATE'),
        )
        # If no theme is installed, this will set
        # the base template to the module's
        # base template.
        for k in dir(config):
            if k.startswith('FOO_'):
                app.config.setdefault(
                    k,
                    getattr(config, k))
```

Template rendering pattern

In views where you render a template you should normally let the template path be defined by a configuration variable. Otherwise it is not possible for an instance to override the template.

```
>>> from flask import current_app, render_template
>>> app.config.update({
...     'FOO_MYVIEW_TEMPLATE': 'invenio_foo/myview.html'
... })
>>> @app.route('/myview')
... def myview():
...     return render_template(
...         current_app.config['FOO_MYVIEW_TEMPLATE']
...     )
```

Alternatively if only some parts of a template should be customizable you can apply the configration variable pattern to the include:

```
{% extends config.FOO_BASE_TEMPLATE %}
{% block page_body %}
{# ... #}
{%- include config.FOO_MYVIEW_TEMPLATE %}
{% endblock%}
```

Menus

Invenio-Theme defines are couple of menus that Invenio modules can plug into. For instance if you want to plug an menu item in the navigation bar you can do it like this:

```
>>> from flask_menu import register_menu
>>> @app.route('/myview2')
... @register_menu(app, 'main.myitem', 'My item', order=1)
... def myview2():
...     return ""
```

By default, any menu items that are children of `main` (such as the above example) will be displayed on the header of every page, via the `page_header` block.

The following menus exists:

- `main` - Header menu navigation (defined by the base template).
- `settings` - User settings menu bar (defined by the settings template).

To read more about the creation and usage of menus, see `flask_menu`.

Breadcrumbs

Breadcrumbs works similar to menus, just use the `register_breadcrumb()` instead.

```
>>> from flask_breadcrumbs import register_breadcrumb
```

Using this decorator, you can specify the position of the view in a hierarchical manner, as well as the title in the breadcrumb. By default, the current breadcrumb is displayed inside the `page_header` block in the base template.

```
>>> @app.route('/part1')
... @register_breadcrumb(app, '.', 'Index')
... def part1():
...     return ""
>>> @app.route('/part2')
... @register_breadcrumb(app, '.p2', 'Part 2')
... def part2():
...     return ""
>>> @app.route('/part3')
... @register_breadcrumb(app, '.p2.p3', 'Part 3')
... def part3():
...     return ""
```

To learn more about the usage of breadcrumbs, see `flask_breadcrumbs`.

User settings

If your module allows your users to configure some settings, you can provide a setting view that is plugged into the settings menu.

First, create your template for your view:

```
{% extends config.FOO_SETTINGS_TEMPLATE %}
{% set panel_icon="fa fa-cog" %}
{% set panel_title="Panel header" %}
{% block settings_form %}
...
{% endblock settings_form %}
```

Next, when creating the view register the view n the `settings` menu:

```
@app.route('/settings/')
@register_breadcrumb(app, '.settings', 'Settings page')
@register_menu(app, 'settings.item1', 'Item 1', order=2)
def settings_item_1():
    return render_template('invenio_foo/foo_settings.html')
```

Message flashing

At any time during a request, you may choose to record a message for the user in order to display it only once in the future. This is a great way for providing feedback to the user after an action, and Flask provides the method `flask.flash()` for this purpose. The first parameter is the message, while you may optionally supply a second parameter that serves as a category that gives context to your message.

Invenio-Theme provides a macro in `invenio_theme/macros/messages.html` called `flashed_messages` that iterates through stored flash messages for the current user and displays them in the current page. The category can be one of 'info', 'danger', 'warning' or 'success'. Using one of these will attach the appropriate Bootstrap class to the message. By default, the current flashed messages will be displayed inside the `page_header` block in the base template.

```
@app.route('/myview')
def settings_item_1():
    flash('Testing flashing', category='info')
    return render_template('...')
```

1.4 Example application

Install the Invenio default theme and build assets:

```
$ pip install -e .[all]
$ cd examples
$ ./app-setup.sh
```

Run the development server:

```
$ FLASK_APP=app.py flask run --debugger -p 5000
```

To be able to uninstall the example app:

```
$ ./app-teardown.sh
```

The example application demonstrates the default templates and the usage of menus and breadcrumbs. The views are registered to the application menu, which is visible on the page header. For more information about menus, see Flask-Menu.

The views displayed in the menu are:

- Base page - Demonstrates the empty **page** template, which has an empty **page_body** block.
- Cover page - Demonstrates the **page_cover** template, which displays the page logo.
- Error pages - Demonstrates the different views for each one of the common error codes.
- Message flashing - Demonstrates the message flashing functionality and flash template, with which you can display a flash message to the user using the `flask.flash()` method provided by Flask.
- Settings - Demonstrates the **page_settings** template. Inside, the submenu **settings** of the current menu is displayed as a list on the left of the page.

On the views that have it, the breadcrumb trail is also displayed below the menu. It displays the location of the current view in the hierarchy. The user may click on the links in the breadcrumb to navigate to the previous views. To read more about breadcrumbs, see Flask-Breadcrumbs.'

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

2.1 API Docs

Invenio standard theme.

class invenio_theme.ext.**InvenioTheme**(*app=None*, ***kwargs*)
Invenio theme extension.

Extension initialization.

Parameters

- **app** – An instance of Flask.
- ****kwargs** – Keyword arguments are passed to init_app method.

init_app(*app*, ***kwargs*)
Initialize application object.

Parameters **app** – An instance of Flask.

init_config(*app*)
Initialize configuration.

Parameters **app** – An instance of Flask.

2.1.1 Bundles

2.1.2 Handlers

Invenio error handlers.

invenio_theme.views.**index**()
Simplistic front page view.

invenio_theme.views.**insufficient_permissions**(*e*)
Error handler to show a 403.html page in case of a 403 error.

invenio_theme.views.**internal_error**(*e*)
Error handler to show a 500.html page in case of a 500 error.

invenio_theme.views.**page_not_found**(*e*)
Error handler to show a 404.html page in case of a 404 error.

invenio_theme.views.**too_many_requests**(*e*)
Error handler to show a 429.html page in case of a 429 error.

invenio_theme.views.**unauthorized**(*e*)
Error handler to show a 401.html page in case of a 401 error.

CHAPTER 3

Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

3.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

3.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/inveniosoftware/invenio-theme/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Invenio Theme could always use more documentation, whether as part of the official Invenio Theme docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inveniosoftware/invenio-theme/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.1.2 Get Started!

Ready to contribute? Here's how to set up *invenio-theme* for local development.

1. Fork the *inveniosoftware/invenio-theme* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/invenio-theme.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv invenio-theme
$ cd invenio-theme/
$ pip install -e .[all]
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

```
$ ./run-tests.sh
```

   The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -s
    -m "component: title without verbs"
    -m "* NEW Adds your new feature."
    -m "* FIX Fixes an existing issue."
    -m "* BETTER Improves and existing feature."
    -m "* Changes something that should not be visible in release notes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

3.1.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests and must not decrease test coverage.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.com/inveniosoftware/invenio-theme/pull_requests and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.2.0 (released 2020-03-20)

- Replaces Flask dependency with `invenio-base`.

Version 1.1.4 (released 2019-07-22)

- Introduce handling of the error 429.

Version 1.1.3 (released 2019-03-13)

- Restructure SCSS files, in order to allow easier customization and extension in overlays.

Version 1.1.2 (released 2019-02-15)

- Upgraded moment to 2.23.0

Version 1.1.1 (released 2018-12-05)

- Fixes issues with webpack and the AdminLTE theme.

Version 1.1.0 (released 2018-11-06)

- Introduce webpack support.

Version 1.0.0 (released 2018-03-23)

- Initial public release.

3.3 License

OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Note: In applying this license, CERN does not waive the privileges and immunities granted to it by virtue of its status as an Intergovernmental Organization or submit itself to any jurisdiction.

3.4 Contributors

- Alexander Ioannidis
- Alizee Pace
- Diego Rodriguez
- Eamonn Maguire
- Esteban J. G. Gabancho
- Harri Hirvonsalo
- Harris Tzovanakis
- Javier Delgado
- Jiri Kuncar
- Lars Holm Nielsen
- Leonardo Rossi
- Nikos Filippakis
- Orestis Melkonian
- Paulina Lach
- Sami Hiltunen
- Tibor Simko

Python Module Index

i

Index

U