## Contents

1 User's Guide 3
  1.1 Installation ................................................................. 3
  1.2 Configuration ............................................................... 3
  1.3 Usage ........................................................................... 4
  1.4 Example application ....................................................... 9

2 API Reference 11
  2.1 API Docs ....................................................................... 11

3 Additional Notes 13
  3.1 Contributing ................................................................. 13
  3.2 Changes ....................................................................... 15
  3.3 License ......................................................................... 15
  3.4 Contributors ............................................................... 16

Python Module Index 17
UI for Invenio-Search.

Further documentation is available on https://invenio-search-ui.readthedocs.io/
1. User’s Guide

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

1.1 Installation

Invenio-Search-UI is on PyPI so all you need is:

```
$ pip install invenio-search-ui
```

1.2 Configuration

Configuration for Invenio-Search-UI.

```python
invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_COUNT = 'templates/invenio_search_ui/count.html'
# Configure the count template.

invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_ERROR = 'templates/invenio_search_ui/error.html'
# Configure the error page template.

invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_FACETS = 'templates/invenio_search_ui/facets.html'
# Configure the facets template.

invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_LOADING = 'templates/invenio_search_ui/loading.html'
# Configure the loading template.

invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_PAGINATION = 'templates/invenio_search_ui/pagination.html'
# Configure the pagination template.

invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_RANGE = 'templates/invenio_search_ui/range.html'
# Configure the range template.

invenio_search_ui.config.SEARCH_UI_JSTEMPLATE_RANGE_OPTIONS = {'histogramId': '#year_hist', 'name': 'years', 'selectionId': '#year_select', 'width': 180}
# Configure the range template options.
```
1.3 Usage

UI for Invenio-Search.

Invenio-Search-UI is responsible for providing an interactive user interface for displaying, filtering and navigating search results from the various endpoints defined in Invenio-Records-REST. This is achieved through the usage of the Invenio-Search-JS AngularJS package and its configuration inside Jinja and AngularJS templates.

Although a default /search endpoint is provided, meant for displaying search results for the main type of records an Invenio instance is storing, it is also possible to define multiple search result pages by extending and configuring some base Jinja and AngularJS templates.

1.3.1 Initialization

Note: The following commands can be either run in a Python shell or written to a separate app.py file which can then be run via python app.py or by running export FLASK_APP=app.py and using the flask CLI tools. You can take inspiration from the Example Application on how the end result of this guide will look like.

To make sure that you have all of the dependencies used installed you should also run pip install invenio-search-ui[all] first.

First create a Flask application:

```python
>>> from flask import Flask
>>> app = Flask('myapp')
```

There are several dependencies that should be initialized in order to make Invenio-Search-UI work correctly.

```python
>>> from invenio_db import InvenioDB
>>> from invenio_pidstore import InvenioPIDStore
>>> from invenio_records import InvenioRecords
>>> from invenio_rest import InvenioREST
>>> from invenio_search import InvenioSearch
>>> from invenio_indexer import InvenioIndexer
>>> from invenio_theme import InvenioTheme
>>> from invenio_i18n import InvenioI18N
```

(continues on next page)
Register the JavaScript bundle, containing Invenio-Search-JS:

```python
>>> from invenio_assets import InvenioAssets
>>> from invenio_search_ui.bundles import js
>>> ext_assets = InvenioAssets(app)
>>> ext_assets.env.register('invenio_search_ui_search_js', js)
```

Before we initialize the Invenio-Search-UI extension, we need to have some REST API endpoints configured to expose our records. For more detailed documentation on configuring the records REST API, you can look into invenio-records-rest.

By default Records REST exposes a /api/records/ endpoint, which resolves integer record identifiers to internal record objects. It uses though a custom Flask URL converter to resolve this integer to a Persistent Identifier, which needs to be registered:

```python
>>> from invenio_records_rest import InvenioRecordsREST
>>> from invenio_records_rest.utils import PIDConverter
>>> app.url_map.converters['pid'] = PIDConverter
>>> ext_records_rest = InvenioRecordsREST(app)
```

Now we can initialize Invenio-Search-UI and register its blueprint:

```python
>>> from invenio_search_ui import InvenioSearchUI
>>> from invenio_search_ui.views import blueprint
>>> ext_search_ui = InvenioSearchUI(app)
>>> app.register_blueprint(blueprint)
```

In order for the following examples to work, you need to work within an Flask application context so let’s push one:

```python
>>> ctx = app.app_context()
>>> ctx.push()
```

Also, for the examples to work we need to create the database and tables (note, in this example we use an in-memory SQLite database):

```python
>>> from invenio_db import db
>>> db.create_all()
```

### Building Assets

In order to render the search results page, you will have to build the JavaScript and CSS assets that the page depends on. To do so you will have to run the following commands:
Record data

Last, but not least, we have to create and index a record:

```python
>>> from uuid import uuid4
>>> from invenio_records.api import Record
>>> from invenio_pidstore.providers.recordid import RecordIdProvider
>>> from invenio_indexer.api import RecordIndexer

>>> rec = Record.create({
    ...
    'title': 'My title',
    ...
    'description': 'My record description',
    ...
    'type': 'article',
    ...
    'creators': [{'name': 'Doe, John'}, {'name': 'Roe, Jane'}],
    ...
    'status': 'published',
    ...
}, id_=uuid4())

>>> provider = RecordIdProvider.create(object_type='rec', object_uuid=rec.id)

>>> db.session.commit()

>>> RecordIndexer().index_by_id(str(rec.id))
```

Feel free to create more records in a similar fashion.

1.3.2 Customizing templates

Search components

The building blocks for the search result page are all the `<invenio-search-...>` Angular directives defined in InvenioSearchJS. You can think of them as the UI components of a classic search page. All of the available directives are listed below (with their default template files):

- `<invenio-search-results>` - The actual result item display (`default.html`)
- `<invenio-search-bar>` - Search input box for queries (`searchbar.html`)
- `<invenio-search-pagination>` - Pagination controls for navigating the search result pages (`pagination.html`)
- `<invenio-search-sort-order>` - Sort order of results, i.e. ascending/descending (`togglebutton.html`)
- `<invenio-search-facets>` - Faceting options for the search results (`facets.html`)
- `<invenio-search-loading>` - Loading indicator for the REST API request (`loading.html`)
- `<invenio-search-count>` - The number of search results (`count.html`)
- `<invenio-search-error>` - Errors returned by the REST API, e.g. 4xx or 5xx (`error.html`)
- `<invenio-search-range>` - Date or numeric range filtering (`range.html`)
- `<invenio-search-select-box>` - Select box for further filtering (`selectbox.html`)

Each one of them accepts attributes for configuring their specific behavior. All of them though accept a `template` attribute specifying an Angular HTML template file which will be used for their rendering, thus defining the component's visual aspects.
In order for them to function, they need to be placed inside `<invenio-search>` tags, which also contain the configuration for the general search mechanics, like the REST API endpoint for the search results, HTTP headers for the request, extra querystring parameters, etc. You can read more about these directives and their parameters in the documentation of Invenio-Search-JS.

These components are placed and configured inside Jinja templates, where one has the choice to either override individual pre-existing Jinja blocks or even completely rearrange the way the components are organize in the template.

**Note:** You can find a full example of this type of configuration and templates in `search.html` and `static/templates`.

### Creating a new search page

Let’s create a new search page exclusively for records. For that we’ll need to add a new route to our application that will render our custom search page Jinja template, `records-search.html`:

```python
from flask import render_template

@app.route('/records-search')
def my_record_search():
    return render_template('records-search.html')
```

Then we need to extend `search.html`, inside our new `templates/records-search.html` template:

```html
{% extends 'invenio_search_ui/search.html' %}

{% set title = 'My custom records search' %}
```

Next we’ll have to configure the `<invenio-search>` root directive with our endpoint and some additional querystring parameters:

```html
... # Contents of templates/records-search.html #
{% block body_inner -%}
<invenio-search
    search-endpoint="/api/records/"
    search-extra-params="{"type": "article"}"
    search-hidden-params="{"status": "published"}"
    search-headers="{"Accept": "application/json"}"
>
{{ super() }}
</invenio-search>
{%- endblock body_inner -%}
...
```

The URL that will be displayed to the user at the top of the search page will look something like (note the missing “hidden” parameter `{status: 'published'}`):

```text
https://myapp.org/search?type=article
```

Our requests to the REST API though will look something like this:

---

**1.3. Usage**
# The hidden parameter

curl -H "Accept: application/json" \\
"https://myapp.org/api/records?status=published&type=article"

Now let’s modify what is displayed in case our REST API request returns an error status code (4xx or 5xx). We do so by creating a new Angular template in static/templates/error.html and passing it to the template parameter of the `<invenio-search-error>` directive.

In our Jinja template `records-search.html`:

```jinja
... 
{% block search_error %}
  <invenio-search-error
    template="{{ url_for('static', filename='templates/error.html') }}"
    message="{{ _('Search failed.') }}">
  </invenio-search-error>
{% endblock search_error %}
... 
```

In our new Angular template `static/templates/error.html`, we are going to add a link to some documentation page when a search error occurs:

```html
<div ng-show='vm.invenioSearchError.name'>
  <div class="alert alert-danger">
    <strong>Error:</strong> {{vm.invenioSearchErrorResults.message || errorMessage }}
    <small><a href="https://myapp.org/help.html"></a></small>
  </div>
</div>
```

Let’s modify the way our search result items are displayed. In order to do so we need to create a `static/templates/results.html` template and update the directive’s template attribute.

In our Jinja template `records-search.html`:

```jinja
... 
{% block search_results %}
  <invenio-search-results
    template="{{ url_for('static', filename='templates/results.html') }}">
  </invenio-search-results>
{% endblock search_results %}
... 
```

In our Angular template `static/templates/results.html`, we are going to display a link to the record’s actual page using the `ng-href` attribute and the links defined in the `record.links` object. We also display the creators of the record in a list by using the `ng-repeat` attribute and the `records.metadata.creators` array field.

```html
<ul>
  <li ng-repeat="record in vm.invenioSearchResults.hits.hits track by $index">
    <a ng-href="record.links.self">
      <h5>{{ record.metadata.title }}</h5>
    </a>
    <ul>
      <li ng-repeat="creator in record.metadata.creators">{{creator.name}}</li>
    </ul>
    <p>{{ record.metadata.description }}</p>
  </li>
</ul>
```
1.4 Example application

1.4.1 Installation process

Run ElastiSearch and RabbitMQ servers.
Create the environment and execute flask:

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

Run the server:

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

Search for example: `wall`.
To be able to uninstall the example app:

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

UI for Invenio-Search.

```python
class invenio_search_ui.ext.InvenioSearchUI (app=None)
```

Invenio-Search-UI extension.

- **Extension initialization.**
  - **Parameters** `app` – The Flask application.

- ** init_app (app) **
  - Flask application initialization.
    - **Parameters** `app` – The Flask application.

- **init_config (app) **
  - Initialize configuration.
    - **Parameters** `app` – The Flask application.

### 2.1.1 Bundles

UI for Invenio-Search.

```python
invenio_search_ui.bundles.catalog (domain)
```

- Return glob matching path to translated messages for a given domain.
2.1.2 Views

UI for Invenio-Search.

```python
invenio_search_ui.views.format_sortoptions(sort_options)
    Create sort options JSON dump for Invenio-Search-JS.

invenio_search_ui.views.search()
    Search page ui.

invenio_search_ui.views.sorted_options(sort_options)
    Sort sort options for display.
```

**Parameters** `sort_options` – A dictionary containing the field name as key and asc/desc as value.

**Returns** A dictionary with sorting options for Invenio-Search-JS.
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


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-Search-UI could always use more documentation, whether as part of the official Invenio-Search-UI 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-search-ui/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-search-ui` for local development.

1. Fork the `invenio-search-ui` repo on GitHub.
2. Clone your fork locally:
   
   ```bash
   git clone git@github.com:your_name_here/invenio-search-ui.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:
   
   ```bash
   mkvirtualenv invenio-search-ui
   cd invenio-search-ui/
   pip install -e .[all]
   ```
4. Create a branch for local development:
   
   ```bash
   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:
   
   ```bash
   ./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:
   
   ```bash
   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, 3.5 and 3.6. Check [https://travis-ci.org/inveniosoftware/invenio-search-ui/pull_requests](https://travis-ci.org/inveniosoftware/invenio-search-ui/pull_requests) and make sure that the tests pass for all supported Python versions.

3.2 Changes

Version 1.1.1 (released 2018-11-12)
- Includes missing assets for AMD build.

Version 1.1.0 (released 2018-11-06)
- Introduces Webpack support.

Version 1.0.1 (released 2018-03-23)
- facets: fix facets templates.

Version 1.0.0 (released 2018-03-23)
- Initial public release.

3.3 License

MIT License

Copyright (C) 2015-2018 CERN.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION 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
- Alizée Pace
- Diego Rodriguez
- Dinos Kousidis
- Esteban J. G. Gabancho
- Harris Tzovanakis
- Jiri Kuncar
- Lars Holm Nielsen
- Leonardo Rossi
- Nicola Tarocco
- Nikos Filippakis
- Sebastian Witowski
- Tibor Simko
## Python Module Index

<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>invenio_search_ui</td>
<td>4</td>
</tr>
<tr>
<td>invenio_search_ui.bundles</td>
<td>11</td>
</tr>
<tr>
<td>invenio_search_ui.config</td>
<td>3</td>
</tr>
<tr>
<td>invenio_search_ui.ext</td>
<td>11</td>
</tr>
<tr>
<td>invenio_search_ui.views</td>
<td>12</td>
</tr>
<tr>
<td>Function/Constant</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>catalog()</td>
<td>(in module invenio_search_ui.bundles), 11</td>
</tr>
<tr>
<td>format_sortoptions()</td>
<td>(in module invenio_search_ui.views), 12</td>
</tr>
<tr>
<td>init_app()</td>
<td>(invenio_search_ui.ext.InvenioSearchUI method), 11</td>
</tr>
<tr>
<td>init_config()</td>
<td>(invenio_search_ui.ext.InvenioSearchUI method), 11</td>
</tr>
<tr>
<td>invenio_search_ui</td>
<td>(module), 4</td>
</tr>
<tr>
<td>invenio_search_ui.bundles</td>
<td>(module), 11</td>
</tr>
<tr>
<td>invenio_search_ui.config</td>
<td>(module), 3</td>
</tr>
<tr>
<td>invenio_search_ui.ext</td>
<td>(module), 11</td>
</tr>
<tr>
<td>invenio_search_ui.views</td>
<td>(module), 12</td>
</tr>
<tr>
<td>InvenioSearchUI</td>
<td>(class in invenio_search_ui.ext), 11</td>
</tr>
<tr>
<td>search()</td>
<td>(in module invenio_search_ui.views), 12</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_COUNT</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_ERROR</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_FACETS</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_LOADING</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_PAGINATION</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_RANGE</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_RANGE_OPTIONS</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLATE_RESULTS</td>
<td>(in module invenio_search_ui.config), 3</td>
</tr>
<tr>
<td>SEARCH_UI_JSTEMPLET_SELECT_BOX</td>
<td>(in module invenio_search_ui.config), 4</td>
</tr>
<tr>
<td>sorted_options()</td>
<td>(in module invenio_search_ui.views), 12</td>
</tr>
</tbody>
</table>