Guide to Working with the Research Group's Website

Last updated: August 27, 2020

Contents

Getting started

The web page is built with `hugo`, a static site generator that can be downloaded at `https://gohugo.io`. This makes editing the web page very easy since `hugo` generates all the html, css and javascript files from simple markdown files; no html to has to be edited by hand.

Install `hugo` and `git` and setup an account with ssh access on the katana server (contact Joachim Simon). If everything is set up, clone the repository that contains all the files to your computer by

```
git clone URL numuq.
```

This will create a folder `numuq` that contains all the necessary files. Navigate into the folder (`cd numuq`) and run `hugo` with the command `hugo server`. This will start a local web server and open a local version of the web page in your web browser. If not, open your web browser and navigate to the url

```
http://localhost:1313.
```

You can now start editing the files; the web page will reload automatically. When you are finished editing the website stop the `hugo server` command and simply run `hugo` (without arguments). This will generate all the files that can then be published to the server. The publishing process is described in detail below. Note that this documentation is also part of the repository and if you find errors or want to extend it, please feel free to do so.

Recurring tasks

`hugo` makes editing and creating pages very easy. It is not necessary to write html, css or javascript; `hugo` uses `markdown` files that are easier to work with than html files. This also makes sure that everything is displayed correctly while you don't have to take care of the layout yourself. Below is a list of recurring tasks such as adding/ editing lectures or editing your personal page.

Before every task

Before you change anything, you have to update your local version to match the one on the server (since someone else could have changed something so that your local version and the version on the server do not match anymore).

Just navigate to the folder that contains the repository on your computer and run

```
git pull.
```

If you do not do this, errors might occur when you try to publish your changes and the changes are conflicting with the files on the server. You can then start the development server by running the command `hugo server`.

Publishing your changes/ Uploading the files to the server

When you are done adding new pages/ changing page content etc., you have to *commit* and *push* the files to the git repository on the server. This is done as follows: Stop the development server that you started earlier. Then, you can create the actual html/css/javascript files that will be uploaded to the server by simply running the command `hugo`. If no errors occurred, you can type `git status` to see the files `git` thinks have changed. These should be the actual markdown files you created or changed but also the corresponding html files in the folder `public/` (**This step is very important since otherwise the markdown files and corresponding html files do not match and your changes will not get published**).

If everything looks okay, type `git add -u`. This tells `git` that you want to track the changes you made to files that already existed in the repository before you started. If you also created new pages, you have to add them manually by typing for each new page (again: don't forget the newly created pages in the `public` folder)

```
git add path/to/new/page
```

You can again run `git status` to see if you have added all the files you added/modified.

Now you can commit the changes to the repository by typing

```
git commit -m "Add new page for course xyz"
```

The message after the parameter `-m` is the so called *commit message* and should describe your changes. Now you can upload the files to the server by running

```
git push
```

If everything is setup correctly, your files are now uploaded to the server and will automatically get published to the web. Note that it might take a few minutes until you see the changes; you might also want to reload the page using `Ctrl + Shift + R` (Linux/Windows) or `Cmd + Shift + R` (Mac) to force your web browser to fetch the updated version.

Adding a new semester

Adding a new entry on the teaching page is done by editing the file

`content/teaching/_index.md`.

The file is broken up into different semesters, which are called *terms* in this context. The first term in the file automatically gets interpreted as the current semester (see the Teaching page on the website to see the difference between the way the semesters are displayed). The remaining terms are rendered as "past terms". This makes adding new terms very easy. Simply add the new term above the term that is currently at the top. The syntax that defines a new term is

```
[[term]]
  title = "WiSe 2019/20"
```

Adding a new lecture/ seminar/ course

Under each term in the file `content/teaching/_index.md`, you can define as many lectures as you want. Note, that no matter if it is a lecture, a seminar or something else you want to add, in `hugo` it is always referred to as a lecture. Every lecture has the following properties

- A title/name called `title`
- A link to its LSF page called `lsf`
- A link to its moodle page called `moodle`
- A link to a separate course page called `page` (setting up a course page is described below)

Using these properties one can define a lecture using

```
[[term.lecture]]
  title = "Example Lecture"
  lsf = "https:://lsf.uni-heidelberg.de/xyz"
  moodle = "https://moodle.uni-heidelberg.de/xyz"
  page = "examplelecture"
```

None of the properties (except for the title) is mandatory. If the `page` property is set, the lecture is automatically turned into a link that links to a separate course page. Of course, this page has to be created which is done by creating a markdown file in the folder `content/teaching`. The name of the file has to equal the value of the page parameter. For example, in the example above we would need to create a file named `examplelecture.md`.

At the top of this file, enclosed in three plus symbols "+++" the title of the lecture has to be given. This can differ from the title given above so that on the teaching page a short name can be given (e.g. "Numerik 0" on the teaching page and "Einführung in die Numerik" on the detail page). Below the plus symbols arbitrary content can be added in markdown format. It is easiest to copy an existing file and edit it according to your needs.

Adding a page for a group member

Adding a new group member is done in two steps. First, we add the a new entry to the file `content/people/_index.md`. There are two types of entries, one for PhD students and one for postdocs. Both have the following three properties:

- The name of the member called `name`.
- A short description of the research area called `topic`.
- The name of the personal page called `page` (explained below).

The second step is to create and setup the personal page. To this end, we first create a markdown file with the name set to the value of `page` from above. This file contains additional information about the respective member such as contact details and (optionally) a picture. It is easiest to simply copy an existing page and edit it accordingly. Most of the parameters are self-explanatory expect for `pub_name` and `research_interest`.

`pub_name` Every group member's page contains a link to the publications page. The publications page allows the user to filter the entries (e.g. by author) and the value of `pub_name` sets this value for the respective group member when the user clicks on the "see publications" link of a member. Thus, this value should in most cases be set to the last name.

`research_interest` This parameter contains a more detailed description of the research interest. To add multi-line content, you can use the following syntax

```
research_interest = """
I am interested in

* Numerical analysis

* Uncertainty Quantification
"""
```

This would result in a list, similar to the following:

I am interested in

- Numerical Analysis
- Uncertainty Quantification

Using LaTeX

Generally, everywhere where you can put text, you can also put Latex code. Of course, only a subset of the Latex functionality is supported but simple equations or symbols can be used. Note, however, that due to the fact that `hugo` does not (yet) support the library that is responsible for converting the Latex code to HTML, you have to escape every backslash "\", i.e. you have to duplicate every backslash. Apart from that, you can use it as usual. To use display the maths in inline mode you have to enclose the Latex code by a dollar sign "$", to render an equation in display mode you have to use two dollar sings "$$". For instance, to achieve something like

$$\hat{\mathbb{E}}(f) = \frac{1}{N} \sum_{i=1}^{N} f(X_i)$$

you would type

```
$$
  \\hat{\\mathbb{E}}(f) = \\frac{1}{N} \\sum_{i=1}^N f(X_i)
$$
```