User Guide

https://csci-1301.github.io/about#authors

May 22, 2021 (01:35:38 AM)

Contents

This guide explains how this resource and its source code are organized, how it is built, and how to maintain this resource.

1 Resources Organization

This repository is organized as follows

path	description
`.github`	github templates and workflow files
`code`	code examples
`docs`	additional helpful documents
`img`	all images
`labs`	lab exercises
`lectures`	lecture notes
`problems`	practice problems

path	description
`templates`	templates and meta data files used for building this resource
`index.md`	website index page
`404.md`	website 404 page

Additional configuration files are at the root of the source code repository.

1.1 Locating course resources

How to obtain the latest version of this resource:

1. visit the accompanying website[1] - this website includes the latest version of the course textbook in all supported formats, links to labs, and all other available student resources
2. latest version is also available under releases[2] on Github

Copies of this resource will be made available through AU mirror website, box, and/or instructor's own website. Since manual effort is required to share the resources through these ways, these sources may be slightly behind latest version.

How to obtain much earlier versions of this resource:

1. Complete release history is accessible under releases[3] on Github
2. This resource will be periodically archived on Galileo

2 Editing Resources

If you are new to this project, first see Contributing Guidelines[4] to learn about the different ways you can contribute and how to join contributing teams.

2.1 Best practices for all forms of content

- **Structure for accesibility**
  - All resources are titled TODO: explain how
  - All resources are labelled when applicable
    - at minimum list prerequisites and security-related aspects
    - see Content Labelling for more details
- **Markdown**
  - text documents are written in readme files using standard markdown syntax
  - we will use a convention of always naming such files `readme.md` (lowercase)
- **Images**
  - Images require a descriptive alt tag for accessibility
  - Store images in the repository in `img` directory because pandoc will look for images there

[1] https://csci-1301.github.io/
[2] https://github.com/csci-1301/csci-1301.github.io/releases
[3] https://github.com/csci-1301/csci-1301.github.io/releases
[4] /contributing

- When refering to images in markdown, use path from root even if the image may appear broken locally

Syntax example. The quoted text is the alt tag and in parentheses is path to file

```
!["image of visual studio IDE"](./img/vs_ide.jpg)
```

- **Source code**
  - source code programs belong *primarily* in `code` directory
    - the code included in this directory should be a complete program
    - the program should compile and terminate
    - source code that is faulty, partial or does not terminate can be included in markdown as inline code block
    - we can automatically check these code snippets for syntactical correctness if these guidelines are followed
  - code snippet can be included in markdown documents using pandoc-include filter:
  - code blocks are by default annotated as `csharp` unless stated otherwise
  - syntax highlighting is applied automatically at build time
  - only include code in text form such that it can be copy-pasted for reuse

2.2 Creating new lectures

To create a new lecture `lecture xyz`:

1. Create a directory called `NNN_lecture_xyz` under `lectures` directory

   Following the existing pattern for naming convention which is lowercase and separation by underscores. The numbers `NNN` tell pandoc how to order book content. Use leading zero and increments of 10. Choose a number number based on where in the book the new lecture should appear.

2. under that directory, create a new file `readme.md` (lowercase). Write lecture notes in this file using markdown.

   We use filename readme because the build script looks for files matching this pattern. Pandoc appears to use a case-insensitive matching, at least currently, but to be safe use lowercase convention when naming this file to match the expected file pattern[5].

Following these steps will automatically include the new lecture in the book.

Do not include meta section in individual lecture files because these lectures will be concatenated by pandoc into single larger document. Any meta data in individual files would appear somewhere in the middle of the larger document, and as such will not be treated as front matter.

[5] https://github.com/csci-1301/csci-1301.github.io/blob/d0cca5dfab111ed9148256992b63fbed9c05b880/Makefile#L14

2.3 Creating new labs

1. Choose a short and unique name that describes the lab
2. Create a directory matching that name (use PascalCase since this is C#)
3. Under the lab directory create:
    1. `readme.md` (case sensitive) - write lab instructions in this file. You should include meta fields, at minimum a title
    2. (optional) if you want to include starter code with the lab,
        - create a subdirectory called `src`
        - create a subdirectory with the name of the solution you would like to use,
        - create a subdirectory with the name of the project you would like to use,
        - create a file called `Program.cs` in `src/<solution>/<project>/Program.cs` Write your code in this file, add a class file if you want in the same folder, but only include program files (not solution or project): they will be created automatically using the project and solution's name you specified, and hosted in the lab's folder as `solution.zip`.

If you follow these instructions the lab will be automatically built into a distributable format when you commit changes. It works as follows:

1. readme will be converted to index (html, pdf, odt)
2. contents of `src` will be converted to a standalone C# solution as a zip file. This solution will have the same name as the folder in the lab directory which is why directory names should use PascalCase.
3. all matching the described format will be included in `labs.zip` which you can find on the website or under releases

3 Content Labelling

Course resources are labelled with emoji shortcodes or text labels.

Each resource should, at minimum, list its prerequisites and security-related content.

3.1 Labelling with shortcodes

Use emoji shortcodes to label following course resources

Description	Shortcode	Icon
Security related aspects will be labelled as "security"	`:lock:`	
Optional parts will be labelled as "optional"	`:question:`	
Examples of common pitfalls	`:warning:`	

3.2 Labelling using text labels

1. Each resource will be labelled with prerequisites.

    This is a list of zero or more values. For zero prerequisites we write `None`. These requirements are expressed in the associated index of lectures/labs/problems (example[6]).

2. Lecture notes and slides will be labelled by related labs, and vice versa

[6] https://github.com/csci-1301/csci-1301.github.io/tree/main/lectures

These requirements are expressed in the associated index of lectures and labs (example[7]).

4 Styling and Templating

Templating files are under `templates` directory.

Templates directory specifies layout files and stylesheets used in the website. These layouts are applied by pandoc when resources are built.

For maintainability reasons it is preferable to apply templates during build time. This strategy makes it easy to edit templates later and apply those changes across all resources. Avoid applying templating to individual resource files whenever possible.

Currently templates directory contains the following:

- `default-code-class.lua` - pandoc filter for annotating code blocks, configured to default to C#, which then allows applying syntax highlighting to all code block.
- `templates/labs` - templates used for generating lab resources and associated pages
- `templates/web` - templates for website and HTML format resources.

5 Repository Maintenance

This repository uses following tools and technologies:

- git - version control
- Github - to make source code available on the web
- markdown, LaTeX - for writing the resources
- pandoc - for converting documents to various output formats
- make - for specifying how to build this resource
- github actions - to automatically build the resource
- github pages - to serve the accompanying website
- additional packages for specific tasks: texlive, Pygments, pandoc filters, lua filter[8], etc.
- Anchor.js[9] - for automatic links.

5.1 How build outputs are generated

The resource material is organized into specific directories (cf. resource organization). These resources are then compiled into templated documents in various formats using pandoc[10]. Different directories undergo different build steps as defined in the project Makefile[11] and generate various outputs. For example, lecture notes are compiled into a textbook and labs are packaged into individual labs. The makefile explains the exact steps applied to each type of resource.

[7] https://github.com/csci-1301/csci-1301.github.io/tree/main/lectures
[8] https://github.com/jgm/pandoc/issues/2104
[9] https://www.bryanbraun.com/anchorjs/
[10] https://pandoc.org/MANUAL.html
[11] https://github.com/csci-1301/csci-1301.github.io/blob/main/Makefile

5.2 Using Github actions with pandoc

This resource is built automatically every time changes are committed to the main branch of the repository. This is configured to run on Github actions[12]. There are currently two configured workflows[13]: one to build the resource and to deploy it, and a second one to check that any opened pull requests can be built successfully.

The build configuration uses texlive to keep the dependency installation time low. Similarly, the choice of Python packages is preferable for pandoc filters, because they are usually straightforward and fast to install. We want to avoid choosing packages that significantly increase build time.

Currently (May 2021) Github actions offers 2,000 free build minutes/month, which *should be* sufficient for the needs of this project and hopefully remains free in perpetuity (if it does not there are other alternative services). Going with one specific CI service over another is simply a matter of preference.

Following a successful build, the build script will automatically deploy the generated resources to an accompanying website hosted on github pages[14]. In the repository a special branch `gh-pages` represents the contents of the deployed website. It also allows maintainers to observe the generated build outputs.

5.3 How to create releases

Currently a github action is setup to do the following: whenever a new commit is made to the main branch, the action will build the resource and add the generated books as a pre-release under releases and tag them as "latest". If a subsequent commit occurs it will overwrite the previous latest files and become the new latest version. This cycle continues until maintainers are ready to make a versioned release (or "package").

Making a versioned release is done as follows:

1. Go to repository releases[15]
2. Choose latest, which contains the files of the latest build
3. Edit this release, giving it a semantic name and a version, such as `v1.0.0`. Name and version can be the same. (cf. semantic versioning[16])
4. Enter release notes to explain what changed since last release
5. Uncheck `This is a pre-release`
6. Update release

Following these steps will generate a new, versioned release. The versioned releases will be manually uploaded to university mirror site, box, and archived.

5.4 Building the resource locally

It is generally not necessary to build this resource locally unless the intent is to preview templating changes or to make changes to build scripts. For the purposes of editing content, it is sufficient to make edits to markdown files and committing those changes.

[12] https://github.com/features/actions
[13] https://github.com/csci-1301/csci-1301.github.io/actions
[14] https://pages.github.com/
[15] https://github.com/csci-1301/csci-1301.github.io/releases
[16] https://semver.org/

5.4.1 Installing required dependencies

To find the current list of dependencies needed to build this resource, refer to the build script install section[17] which lists all required packages need to build the resource. The exact installation steps vary depending on your local operating system.

In general the following dependencies are needed:

- pandoc
- texlive
- make
- python 3.+
- packages and filters: Pygments, pandoc-include, texlive-xetex texlive-latex-extra, lmodern, librsvg2-bin

5.4.2 Running the build

After installing all dependencies, from the repository root, run:

```
make
```

To see a list of other alternative build options run

```
make help
```

[17] https://github.com/csci-1301/csci-1301.github.io/blob/main/.github/workflows/build.yaml