igem-wikisync

Release 1.1.0-alpha3

Nov 08, 2020

Contents

iGEM WikiSync is an elegant and simple way to upload iGEM Wikis.

WikiSync eliminates the need to manually upload each file, replace each URL and copy paste your source code into a web form. Building and deployment can now be as simple as a `git push`, thanks to Travis.

Please head over to the *Overview* to read more about this project, or dive right in with the *Usage Guide*. We've also created a step-by-step *Tutorial* to get your wiki up and running quickly.

CHAPTER 1

Installation

```
pip install igem-wikisync
```

CHAPTER 2

Collaboration

Using this software or submitting issues and pull requests can count towards a collaboration for our teams. Please give us a shoutout at @igem_bits on Instagram if WikiSync has made your iGEM experience easier! For contibuting to this software or discussing further collaborations, please reach out to us at igembitsgoa@gmail.com.

CHAPTER 3

Contents

3.1 Overview

WikiSync is a Python library that allows you to easily upload your iGEM wiki. It eliminates the need to manually upload each file, replace each URL and copy paste your source code into a web form. Building and deployment can now be as simple as a `git push`, thanks to Travis.

All you need are five lines of code:

```
import igem_wikisync as sync

sync.run(
    team='your_team_name',
    src_dir='source_directory',
    build_dir='build_directory'
)
```

WikiSync goes through each media file or document in your wiki folder, and uploads them. It then goes through your source code (HTML and CSS files) and replaces all the URLs with those received after uploading files. It then uploads this modified source code as well. It also checks for broken links.

By automating this, allows you to leverage all the features of modern code editing software like Visual Studio Code. You can see how your wiki looks as you code and when you're done, you can effortlessly push your code to iGEM servers, by running just one command.

It also seamlessly integrates with continuous integration software, which allows your entire team to collaborate on the wiki, and finally upload it without any extra effort.

To get started, proceed to the *Installation* instructions. Then, head over to the *Usage Guide* or take a look at our *Tutorial* for step-by-step examples meant to help you make deployment as easy as a git push.

3.1.1 Features

1. Uploads media and documents

2. Checks for broken links
3. Replaces links in source code (HTML and CSS)
4. Uploads modified source code
5. Keeps track of uploads and only uploads changes
6. Renames media and documents according to iGEM specifications

Other Advantages of Using WikiSync

WikiSync allows you to leverage:

1. Modern IDEs like Visual Studio Code
2. Collaboration through Github.
3. Automatic deployment through Travis CI

3.2 Installation

WikiSync is a Python package, so it can be installed like any other.

Execute the following at the command line:

```
pip install igem-wikisync
```

WikiSync is supported only on Python 3.5+.

We've tested WikiSync on several operating systems across all supported Python versions. However, it's still in development and as we wait for iGEM teams to adopt it and provide feedback, we request you to kindly keep your copy of the software updated. You can do that by running the following command before you use WikiSync.

```
pip install -U igem-wikisync
```

If you don't have `pip` installed, please click here for instructions.

3.3 Tutorial

Table of Contents

Please reach out for any queries at ballaneypranav@gmail.com.

3.3.1 Uploading a Test Folder

If you'd like to test the functionality first, make a test folder with just a few files and try to upload that. The following example demonstrates that in more detail.

#1 **Start with the following directory structure:**

```
wiki/
    src/
        WS-basic/                   # just so that your main wiki is not affected
            index.html              # homepage
            css/
                style.css           # custom styles
                igem-reset.css      # Resets styles that iGEM injects
            js/
                main.js             # custom JS + iGEM reset
            Description/
                index.html          # Description page
        assets/                     # everything else must be in the assets folder
            WS-basic/               # just so that your main wiki is not affected
                img/
                    logo.jpg
                    background.jpg
    build/
        # this will be filled by WikiSync
    wikisync.py
```

The source code is inside `wiki/src/WS-basic/` instead of just `wiki/src/` so that any existing content on your wiki is not affected. Similarly, images are inside `assets/WS-basic/img/` instead of just `assets/img`.

Please download a zipped version of this code here.

#2 **Let's look at individual files now:**

`src/WS-basic/index.html`:

```
<html lang="en">

<head>
    <title>Testing iGEM WikiSync</title>
    <link rel="stylesheet" href="css/igem-reset.css">
    <link rel="stylesheet" href="css/style.css">
</head>

<body>
    <h1>iGEM Example Wiki</h1>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="./Description/">Description</a></li>
    </ul>
    <div class="container">
        <br><br>
        <img src="./assets/img/logo.png" alt="iGEM Logo" height=200 width=200>
        <br><br>
        <h1>Welcome to iGEM 2020!</h1>
        <p>This is a sample page, designed for a demonstration for iGEM WikiSync.</p>
    </div>
    <script src="./js/main.js"></script>
</body>
```

(continues on next page)

(continued from previous page)

```
</html>
```

src/WS-basic/css/style.css:

```
body {
    background-color: #f7feff;
    background-image: url(../assets/img/background.png);
}
```

#3 **Create "wikisync.py"**:

```
import igem_wikisync as sync

sync.run(
    team='your_team_name',
    src_dir='source_directory'      # folder where your wiki is stored
    build_dir='build_directory'     # folder where WikiSync will temporarily store␣
↪your wiki before uploading
)
```

#4 **Export your credentials as environment variables**:

On Windows Powershell:

```
$env:IGEM_USERNAME = 'youriGEMusername'
$env:IGEM_PASSWORD = 'youriGEMpassword'
```

You can verify by running:

```
Get-ChildItem Env:IGEM_USERNAME
```

On Mac or Linux:

```
export IGEM_USERNAME=youriGEMusername
export IGEM_PASSWORD=youriGEMpassword
```

You can verify by running:

```
echo $IGEM_USERNAME
```

#5 **Run** `wikisync.py`:

```
python wikisync.py
```

You should now see the following output:

```
> python wikisync.py
Done! Successfully uploaded:
    2 assets
    2 HTML files
    2 stylesheets
    1 JS scripts
Please look at the log for more details.
```

#6 **Let's look at the files WikiSync has written in** `build/` **now:**

build/WS-basic/index.html:

```
<html lang="en"><head>
    <title>Testing iGEM WikiSync</title>
    <link href="https://2020.igem.org/Template:BITSPilani-Goa_India/Test/css/igem-resetCSS?action=raw&amp;ctype=text/css" rel="stylesheet"/>
    <link href="https://2020.igem.org/Template:BITSPilani-Goa_India/Test/css/styleCSS?action=raw&amp;ctype=text/css" rel="stylesheet"/>
</head>

<body>
    <h1>iGEM Example Wiki</h1>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="https://2020.igem.org/Team:BITSPilani-Goa_India/Test/Description">Description</a></li>
    </ul>
    <div class="container">
        <br/><br/>
        <img alt="iGEM Logo" height="200" src="https://2020.igem.org/wiki/images/5/5a/T--BITSPilani-Goa_India--assets--img--logo.png" width="200"/>
        <br/><br/>
        <h1>Welcome to iGEM 2020!</h1>
        <p>This is a sample page, designed for a demonstration for iGEM WikiSync.</p>
    </div>
    <script src="https://2020.igem.org/Template:BITSPilani-Goa_India/Test/js/mainJS?action=raw&amp;ctype=text/javascript"></script>


</body></html>
```

`build/WS-basic/css/style.css`:

```
body {
    background-color: #f7feff;
    background-image: url(https://2020.igem.org/wiki/images/d/dc/T--BITSPilani-Goa_India--assets--img--background.png);
}
```

There are a few things to note here:

1. All the files have been uploaded and their URLs substituted in the code.
2. The filenames have been changed according to iGEM specification.
3. HTML files have been uploaded at `igem.org/Team:` but CSS and JS files have been uploaded at `igem.org/Template:`, and appended with the required URL parameters.
4. A file called `upload_map.yml` should have appeared in your directory. Read more about it the section about *Keeping Track of Changes*.
5. A file called `wikisync.cookies` should have appeared in your directory. Read more about in the section about *Maintaining a Session* and make sure you add it to your `.gitignore`.
6. A file called `wikisync.log` should have appeared in your directory. Read more about it in the section about *Logging*.

Note: We're working on some more tutorials. They will be up soon.

3.3.2 Collaborating with your Team using Github

Git: https://www.youtube.com/watch?v=USjZcfj8yxE&t=217s

Github: https://www.youtube.com/watch?v=nhNq2kIvi9s

3.3.3 Continuous Deployment with Travis

Travis: https://www.youtube.com/watch?v=g0KsiCj3CgQ&t=1s

You'll also need to add `GITHUB_USERNAME`, `IGEM_USERNAME` and `IGEM_PASSWORD` along with `GITHUB_TOKEN` as environment variables on Travis. We will have more details on the process up here soon.

Please read the *Continuous Integration* section in the *Usage Guide* for now. We will have this tutorial up soon.

3.3.4 Testing before Deployment with Github Pages

3.4 Usage Guide

Table of Contents

3.4.1 Getting Started

WikiSync runs through a Python script in your root directory, and looks for your iGEM username and password in your terminal session as environment variables.

Note: In this guide, we assume a familiarity and level of comfort with Python and command line software. If that doesn't sound like you, head over to our *Tutorial* where we explain everything you need to know.

In the Python script, it requires three parameters as input:

1. `src_dir`: Folder where your source code exists.
2. `build_dir`: Folder where WikiSync will save the modified code before uploading.
3. `team`: Your team name registered with iGEM.

Let's assume your wiki folder has the following structure:

```
wiki/
    index.html
    css/
        home.css
        content.css
    js/
        main.js
        content.js
    assets/
        img/
            logo.jpg
        video/
            intro.mp4
    Description/
        index.html
    Design/
        index.html
```

1. Since WikiSync saves the modified source code before uploading, the directory structure needs to change a little:

```
wiki/
    src/
        index.html
        # ... all the content from above
    build/
        # this is where modified code will be stored
```

2. Now, add the Python script, `wikisync.py`:

```
import igem_wikisync as sync

sync.run(
    team='your_team_name',
    src_dir='source_directory',      # 'src' in this case
    build_dir='build_directory'      # 'build' in this case
)
```

3. Export the following environment variables:

```
IGEM_USERNAME=youriGEMusername
IGEM_PASSWORD=youriGEMpassword
```

4. Run wikisync.py by executing:

```
python3 wikisync.py
```

Caution: We use environment variables for credentials so that they're not accidentally committed to Git. If you're using a bash script to export your credentials, please remember to add it to `.gitignore`.

And that's all! Your wiki has been deployed to iGEM!

Read on to see how WikiSync performs optimizations by storing cookies and uploading only the files that have changed.

3.4.2 Maintaining a Session

WikiSync stores cookies so you don't have to login on every run. This reducing network overhead and also makes the overall operation faster.

Cookies are stored in a file called `wikisync.cookies` in the directory where WikiSync is run.

> **Caution:** It is strongly recommended that you add `wikisync.cookies` to `.gitignore`.

3.4.3 Keeping Track of Changes

After each run of WikiSync, it creates a file called `upload_map.yml` in the directory where it was run. This is a list of files it has encountered and uploaded till now, along with their URLs and MD5 hashes. This ensures that existing files are not uploaded again, but their URLs still can be substituted in the code. MD5 hashes allow it to check for changes within existing files, so it can upload the modified versions.

This is also useful in case connection to iGEM servers is lost while uploading. WikiSync saves the intermediate state in the upload map, so you can resume from that point when the internet connection is restored.

The upload map can (and should) be tracked by a version control system, to allow *continuous integration* and deployment through Travis. This also helps you get a bird's eye view of the upload operation without having to read the log.

The upload map should never be edited manually. If this file is deleted/damaged, WikiSync will upload each file again, which can overload the iGEM servers unnecessarily. This can be especially troublesome when all the teams try to upload their content, close to the Wiki Freeze.

3.4.4 Tracking Broken Links

As your wiki grows into several pages and hundreds of links spread across them, it can be hard to find broken links. WikiSync tries to make this easier by checking for broken (internal) links. This functionality is enabled by default to enforce good practice, but it can be disabled. Look at the configuration options to know more about this.

Note: Broken link warnings can be silenced by passing `silence_warnings=True` in the call to `wikisync.run()`.

3.4.5 Logging

WikiSync prints a log of all the operations it carries out, allowing you to oversee them. This log is present in the `wikisync.log` file. You can search for specific events using the following keywords:

Under construction.

Coming up in a few days.

This file doesn't contain any sensitive information, and can be committed to git.

3.4.6 Continuous Integration

Since WikiSync can upload your entire wiki automatically, this job can now be fully integrated into your version control system itself. Travis CI can now deploy to iGEM just as easily as it can deploy to Github Pages.

Note: In this guide, we assume a familiarity and level of comfort with version control systems and continuous integration. If that doesn't sound like you, head over to our *Tutorial* where we explain everything you need to know.

Please find here a Travis configuration file that you can directly include in your project

You'll also need to add `GITHUB_USERNAME`, `IGEM_USERNAME` and `IGEM_PASSWORD` along with `GITHUB_TOKEN` as environment variables on Travis. We will have more details on the process up here soon.

3.5 Reference for Developers

3.5.1 igem_wikisync

igem_wikisync.wikisync.**build_and_upload**(*files*, *browser*, *config*, *upload_map*)

Replaces URLs in files and uploads changed files.

Parameters

- **files** – Custom file cache
- **browser** – mechanicalsoup.StatefulBrowser instance
- **config** – Configuration for this run
- **upload_map** – custom upload map

Returns Dictionary with no. of 'html', 'css' and 'js' files uploaded

igem_wikisync.wikisync.**cache_files**(*upload_map*, *config*)

Loads filenames into memory, along with setting up appropriate objects to generate URLs and hashes as required.

Parameters

- **upload_map** – custom upload map
- **config** – configuration for this run

Returns *cache* – dictionary with html, css, js and other file objects

igem_wikisync.wikisync.**get_browser_with_cookies**()

Creates a mechanicalsoup.StatefulBrowser() instance with cookies loaded from file, if exists.

Returns *browser* – mechanicalsoup.StatefulBrowser() instance cookiejar: browser cookiejar that can be saved after logging in

igem_wikisync.wikisync.**get_upload_map**()

Opens existing upload_map.yml or creates and empty upload map.

Upload map is a dictionary that contains previously uploaded html, css, js and other files, along with their URLs and hashes.

igem_wikisync.wikisync.**run**(*team: str*, *src_dir: str*, *build_dir: str*, *year=2020*, *silence_warnings=False*, *poster_mode=False*)

Runs iGEM-WikiSync and uploads all files to iGEM servers while replacing relative URLs with those on the iGEM server.

Mandatory Arguments: team: iGEM Team Name src_dir: Path to the folder where the source files are present build_dir: Path to the folder where the built files will be stored before uploading

Optional Arguments: year: Subdomain for igem.org. Current year by default. silence_warnings: Broken link warnings are not printed to console if true. The log still contains everything. poster_mode: Run WikiSync in poster mode.

- Renames files to T–[TeamName]–Poster_[filename].extension
- Adds the poster template the HTML file
- Fails if any other HTML/CSS/JS file is provided

Returns *1* – Incorrect input in function call. 2: Connection problem. 3: Invalid upload map. 4: Failed to write/upload file.

igem_wikisync.wikisync.**upload_and_write_assets**(*other_files*, *browser*, *upload_map*, *config*)

" Uploads and writes all files and stores URLs in upload_map.

Parameters

- **other_files** – dictionary containing OtherFile objects
- **browser** – mechanicalsoup.StatefulBrowser instance
- **upload_map** – custom upload map
- **config** – custom configuration options

Returns Number of files uploaded

Raises SystemExit on failure

igem_wikisync.wikisync.**write_upload_map**(*upload_map: dict*, *filename='upload_map.yml'*)

Writes upload map to file.

3.6 Contributing

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

3.6.1 Bug reports

When 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.

3.6.2 Documentation improvements

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

3.6.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/igembitsgoa/igem-wikisync/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 code contributions are welcome :)

3.6.4 Development

To set up *igem-wikisync* for local development:

1. Fork igem-wikisync (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:igembitsgoa/igem-wikisync.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes run all the checks and docs builder with tox one command:

```
tox
```

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

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

3.7 Authors

- Pranav Ballaney - https://github.com/ballaneypranav

3.8 Changelog

3.8.1 1.1.0-alpha3 (2020-11-08)

- Poster mode allows uploading multiple code files as long as they start with /Poster

3.8.2 1.1.0-alpha1 (2020-11-05)

- Fix typo

3.8.3 1.1.0-alpha0 (2020-11-01)

- Alpha release that supports poster mode.
- Has not been tested because wikis have been frozen.
- Use at your own risk!

3.8.4 1.0.0 (2020-10-27)

- iGEM Judging Release!

3.8.5 0.0.15 (2020-10-21)

- Bug fixes

3.8.6 0.0.14 (2020-10-21)

- Support srcset

3.8.7 0.0.13 (2020-08-27)

- Bug fixes

3.8.8 0.0.6 (2020-08-02)

- Allow silencing warnings
- Add WikiSync to upload description

3.8.9 0.0.5 (2020-07-30)

- Check file size before uploading
- Don't change URL if file not found
- Add year configuration option
- Enforce asset paths to start with 'assets/'
- Improve logging and change log and cookie filenames
- Classify logging messages
- Refactor upload_map
- Don't rename files if they already follow iGEM spec
- Check for filename too large
- Print summary after execution

3.8.10 0.0.4 (2020-07-25)

- Drop jsmin.
- Add version specifiers to dependencies.

3.8.11 0.0.3 (2020-07-25)

- Ensures that directories exist before writing files.

3.8.12 0.0.2 (2020-07-25)

- Assets are also written to disk in build_dir.

3.8.13 0.0.1 (2020-07-25)

- Build directory doesn't get cleared on every run.

3.8.14 0.0.0 (2020-07-25)

- First release on PyPI.

Python Module Index

i

Index