Project Automation

Why is it so hard for me to work on anything without having an overly elaborate system in place first?

2025-01-22
Website Link (http://mayaferrandiz.com/project-automation)

Specifications

Materials: primary, secondary, consumables

Dimensions: cm w x cm h x cm d

Weight: kg

Requirements

Setup & Maintenance

This is a tool to automate the creation and management of art and personal projects. It creates local project directories to store work-in-progress files and media, GitHub repositories, and Things 3 projects with consistent organization and metadata. Once ready, it also facilitates publication to a Jekyll website in the form of project-specific posts and an overall roadmap.

(Almost) all content on this website is published and maintaned using this script.

Setup

1. Clone the repository:

```
git clone git@github.com:yourusername/project-automation.git
cd script
```

1. Create and activate a virtual environment:

```
python -m venv .venv
source .venv/bin/activate  # On Windows: venv\Scripts\activate
```

1. Install dependencies:

```
pip3 install -r requirements.txt
```

1. Create your environment file:

```
cp .env.example .env
# Edit .env with your values
```

Environment Variables

- `WEBSITE_DOMAIN`: The domain on which your Jekyll website will ultimately be served
- `PROJECT_BASE_DIR`: Path for where you would like to create and manage projects
- `JEKYLL_DIR`: Path to where your Jekyll site content is located
- `GITHUB_USERNAME`: Your GitHub username
- `GITHUB_TOKEN`: Your GitHub personal access token
- `ENABLE_THINGS3`: Enable/disable Things 3 integration (true/false)

Usage

Create a new project:

```
python -m script.src.main

# Alternatively...
python -m script.src.main --command create
```

List all projects:

```
python -m script.src.main --command list
```

This will show all projects with their display names, canonical names, creation dates, and status.

Publish project updates:

```
# Publish all out-of-date projects
python -m script.src.main --command publish --all

# Publish specific project
python -m script.src.main --command publish --project project-name

# Publish website (for any non-project website changes)
python -m script.src.main --command publish --website
```

Rename a project:

```
# Start rename process (will prompt for new name)
python -m script.src.main --command rename --project project-name
```

The rename command will: - Update the Things 3 project name - Rename the local directory - Update all project metadata - Update README headers - Rename the GitHub repository - Update Jekyll site files

Project Structure

Each project is created with the following structure:

```
project-name/
├── src/              # Source code
├── docs/             # Documentation
├── hardware/         # Hardware-related files
├── media/           # Published media (tracked in git)
│   ├── images/
│   ├── videos/
│   └── models/
├── media-internal/  # Internal work-in-progress media (git-ignored)
│   ├── images/
│   ├── videos/
│   └── models/
├── content.md       # Project documentation
└── metadata.yml     # Project metadata
```

The script uses the content.md + anything in the /media folder as the basis for publishing to any platform. It uses this information to generate a README.md for Github and Jekyll post using platform-specific formatting.

Repository Structure

```
project-automation/
├── src/
│   └── automation.py
│   └── config.py
│   └── constants.py
│   └── file.py
│   └── github.py
│   └── jekyll.py
│   └── main.py
│   └── things.py
```

```
│   └── utils.py
├── templates/
│   ├── content.md           # Template for project READMEs
│   ├── metadata.yml        # Template for project metadata
│   ├── gitignore          # Template for project .gitignore
├── .env.example            # Example environment variables
├── .env                    # Local environment variables (create your own)
├── .gitignore
├── requirements.txt        # Python dependencies
└── README.md              # This file
```

Features

- Creates standardized project structure
- Initializes git repository and pushes to GitHub
- Creates Things 3 project in specified area
- Manages both published and internal media
- Syncs project metadata with Jekyll portfolio site

Development

To modify templates or add features: 1. Templates are in the `templates/` directory 2. Script files are in `src/` 3. Add any new dependencies to `requirements.txt`

Things 3 Integration

The script integrates with Things 3 on macOS: - Creates projects in a specified area (configured as "🎨 Art") - Updates project names when using the rename command - Manages project organization automatically

Naming Convention

Projects use two types of names: - Display Name: User-friendly name with emoji (e.g., "🌱 Project Name") - Canonical Name: URL-safe, lowercase name (e.g., "project-name")

The script automatically handles conversion between these formats.