Contribution Guide

Overview

Bug Fixes & Patches

The HUBzero Foundation accepts bug fixes and core patches. Submissions are reviewed by Foundation technical staff and, if accepted, they will become part of the HUBzero core distribution. When contributions are offered to the HUBzero Foundation, the copyright for the software must be reassigned to the HUBzero Foundation so that the changes can be managed as an indistinguishable part of the core distribution. The HUBzero Foundation reserves the right to reject submissions for any reason.

Adherence to Standards

The HUBzero Foundation has established coding conventions and guidelines for developing HUBzero components. All submissions to the core must adhere to the guidelines, coding patterns, and styles laid forth in the documentation. Contributions are reviewed by the Foundation team before being incorporated into the core distribution and those that do not meet the standards may be tweaked or rewritten as needed, or may be accepted pending required changes. Any contribution known to contain security vulnerabilities may be rejected entirely.

Hosted Hubs

All code extensions and alterations must adhere to the aforementioned coding conventions and guidelines for hubs hosted and maintained by HUBzero. Third-party extensions are allowable but must first be reviewed and accepted to ensure compatibility, functionality, and security concerns are addressed.

Guidelines

- Be sure that the code follows the development styles and conventions
- Make sure it works.
- If it requires internationalization, documentation, or help to be written, be sure you've done that before submission.
- Variables, subroutines, and comments in the code should be made in English. While the talents of the worldwide community is greatly appreciated, we cannot support code that we cannot read.

Subjective Requirements

Below are some subjective requirements that should be taken into consideration.

1. **Is this feature useful?**
   Consider whether this feature is useful to the current or future users. If there are relatively few people for who this is useful: what other reasons are there to include this?

If this feature is useful only to some, can it be made in a way that it can be switched off or easily removed, so that it's out of the way of the average user.

2. **Is this feature usable by the target audience?**
   Consider who is the target audience? Is it user-friendly for regular users, does a content manager in an avarage business environment have the skills to use it? Or is this directed to system administrators?
3. **What are the maintenance costs of it?**
   Strongly consider the time and effort spent in maintaining it.
4. **Does this fit in the direction HUBzero is going?**
   This may be something you want to discuss on the forum with other hub owners and users, but you can also submit inquiries to the HUBzero team.
5. **Is there already a similar feature?**
   If so: could this similar feature be adapted to suit you needs? Or does your contribution have all the features of the existing feature? And if so, is there a way to upgrade?

Code Guidelines

Hubzero maintains a set of guidelines and rules for code contributions. These are aimed at

creating a consistent, readable, security-minded, and maintainable codebase.

HUBzero Git Flow Slides

Standards and Code Styling

All code contributions must adhere to the style and conventions outlined in the following: Developer Conventions

PHP CodeSniffer

You can check your code using PHP CodeSniffer. This should be available in most repositories.

On the Amazon Instance or CentOS 6.5 installations, you may install this using sudo yum install phpcs Alternatively, you may install this from source: Code Sniffer - GitHUB

Once installed on your system you can clone the HUBzero Standards Repository from Hubzero Standards. It’s recommended to install the standards outside of the hub installation directory.

git clone -b hubzero-cms-2.1 https://github.com/hubzero/standards /home/<username>/standards

To run PHPCS, you would run an incantation similar to:

phpcs -np --standard=/home/<username>/standards/Php/ruleset.xml --ignore=*/test/*,*/assets/*,*/views/*,*/tmpl/*./path/to/code

Where <username> is in your Linux username (centos, more than likely if you're on an AWS instance).

Due to the size of the hubzero-cms codebase, it is recommended that you only run the check against files you've modified. Each file can be specified at the end of the command, replacing (./path/to/code), separated by spaces.

Suggested Workflow

Hubzero Web Developers have a couple of workflows which have developed organically over the years. In the future, we would like to use tooling to automate the development environment. Due to the complex configuration of all the hub software, starting from the VMware Image available for download.

A couple of developers have gotten together to put together a CMS-only environment using Vagrant: CMS-only Vagrant Environment. Instructions for using this environment are available in the repository's README file.

Configuration

The configuration necessary for web development is outlined below. There may be some variables such as login username, file locations, and users depending on the operating system. A person familiar with a LAMP environment should be able to ascertain these variables with relative ease. In all cases configuration secrets, such as MySQL root user password and admin user password are located in /etc/hubzero.secrets. The Hubzero command-line utility, hzcms, will create this file upon installation of the HUB.

External Contribution Process

Advanced Workflows

Unless you are confident in your Linux system administration skills, the Hub can be built via

Debian and RedHat packages. In the future, we anticipate moving to support RedHat / CentOS over Debian. In our effort to increase our presence on cloud service providers, RedHat / CentOS has been chosen to be implemented first.

Development Workflow on an AWS CentOS Image

Another approach, which is used internally at HUBzero, is to use the Amazon Marketplace Image.

1. Create a GitHub account and Fork the hubzero cms repository
2. ssh centos@hostname.aws.hubzero.org
    1. Log into the development environment.
3. cd /var/www
    1. Go to the web root directory.
4. cv hub hub.bak
    1. Backup the existing code which will be necessary to preserve the configuration files.
5. git clone https://github.com/gitusername/hubzero-cms.git hub
    1. Clone the development repository, naming it “hub” to preserve the preconfigured location that the web server (Apache) expects.
6. git remote add upstream https://github.com/hubzero-cms
    1. This adds the HUBzero repository as your upstream remote. This allows you to pull down the latest history.
7. cp hub.bak/app hub/
    1. Copy the configuration, contained within the app directory, to the new installation.
8. cd hub/core
    1. a. Go to the core directory.
9. php composer install
    1. Pull in external dependencies (including the HUBzero Framework).
10. cd . . /
    1. Go back to the /var/www/hub directory.
11. php muse migration -i -f
    1. Run database migrations.
12. cd /var/www
    1. Go to the /var/www/ directory
13. chown apache:apahce hub - R
    1. Change the owner and group of the hub directory to something that the web server agrees with.
14. chmod 775 hub - R
    1. Change the file mode back to something permissible for development purposes. In production environments this is a little more locked down.
15. git checkout <branch>
    1. At the time of writing: 2.1.0 is the stable branch.
        1. This branch really should have been called 2.1 and is treated as such. Each minor release is branched from a point in time on 2.1.0.

    2. master is the unstable branch.
16. git fetch --all
    1. Pull the latest code.
17. git pull --rebase upstream/<branch>
    1. Merges the latest changes from upstream into your local branch.
18. git checkout -b <feature-branch>
    1. This creates a feature branch. This is something completely arbitrary that means something useful to you as a developer.
    2. This will create a branch from whatever branch you select in step 14 and will have the latest history from steps 15 and 16.
19. <make some valid changes to the code>
    1. Fix a bug, for instance.
20. git add <file>
    1. Add the file(s) to the commit.
21. git commit -m "Descriptive message"
    1. Form the commit and add a descriptive message meeting the standardized message formatting requirements.
22. git push origin <feature-branch>
    1. Push your changes to your repository.
23. Open Pull Request on GitHub
    1. This starts the code review process. Another developer must approve the code submission(s) before they enter the mainline.
    2. It is highly recommended that you run PHPCS against your code before opening pull request. Styling errors may lead to a rejection of your code.
24. For further contributions, repeat steps 15 - 23

General Tips

- Explore your dev environment. It’s yours! You should have your own personal development environment. Use it to explore the Hubzero platform as a whole. Most of the Hubzero code is written in PHP or Python, which can be read via a text editor.
- Don’t be afraid to “trash” your development environment. Development environments should be messy. They’re your workspace to make, break, and create cool stuff. In VMware and other virtual machine applications, creating snapshots is a convenient way to return the machine to a working state.
- Ask questions. Feel free to ask Hubzero Developers questions through support tickets or our Contributor’s channel at https://hubzerocontributors.slack.com.
- Read the documentation. We try to do our best to keep the essentials in our documentation up-to-date. If something doesn’t seem to be working for you, let us know! We’ll help you out.
- Use Muse. Muse is the CMS’s command-line interface. There is a lot of functionality it brings to the table.
- Submit Bug Fixes. Come across an issue in core? Submit a ticket or contribute a fix !

The Hubzero codebase is extensive and there are a lot of features that need upkeep.

- Create Portable Code. Use relative paths, environment variables, and DNS entries (no IP addresses) to reference files and assets to ensure your project does not break when your code is migrated to another machine during maintenance.