# CONTRIBUTING

## RESOURCES

If you wish to contribute to Zend Framework, please be sure to

read/subscribe to the following resources:

- [Coding Standards](https://github.com/zendframework/zf2/wiki/Coding-Standards)

- [Contributor's Guide](http://framework.zend.com/participate/contributor-guide)

- ZF Contributor's mailing list:

Archives: http://zend-framework-community.634137.n4.nabble.com/ZF-Contributor-f680267.html

Subscribe: zf-contributors-subscribe@lists.zend.com

- ZF Contributor's IRC channel:

#zftalk.dev on Freenode.net

If you are working on new features or refactoring [create a proposal](https://github.com/zendframework/ZendSkeletonApplication/issues/new).

## Reporting Potential Security Issues

If you have encountered a potential security vulnerability, please \*\*DO NOT\*\* report it on the public

issue tracker: send it to us at [zf-security@zend.com](mailto:zf-security@zend.com) instead.

We will work with you to verify the vulnerability and patch it as soon as possible.

When reporting issues, please provide the following information:

- Component(s) affected

- A description indicating how to reproduce the issue

- A summary of the security vulnerability and impact

We request that you contact us via the email address above and give the project

contributors a chance to resolve the vulnerability and issue a new release prior

to any public exposure; this helps protect users and provides them with a chance

to upgrade and/or update in order to protect their applications.

For sensitive email communications, please use [our PGP key](http://framework.zend.com/zf-security-pgp-key.asc).

## RUNNING TESTS

First, use [Composer](https://getcomposer.org) to install all dependencies:

```bash

$ composer install

```

Make sure that `zendframework/zend-test` is installed:

```bash

$ composer require --dev zendframework/zend-test

```

To run tests:

```bash

$ composer test

```

You can turn on conditional tests with the `phpunit.xml` file.

To do so:

- Copy `phpunit.xml.dist` file to `phpunit.xml`

- Edit `phpunit.xml` to enable any specific functionality you

want to test, as well as to provide test values to utilize.

## Running Coding Standards Checks

First, ensure you've installed dependencies via composer:

```bash

$ composer require --dev squizlabs/php\_codesniffer

```

To run CS checks only:

```console

$ composer cs-check

```

To attempt to automatically fix common CS issues:

```console

$ composer cs-fix

```

If the above fixes any CS issues, please re-run the tests to ensure

they pass, and make sure you add and commit the changes after verification.

## Recommended Workflow for Contributions

Your first step is to establish a public repository from which we can

pull your work into the master repository. We recommend using

[GitHub](https://github.com), as that is where the component is already hosted.

1. Setup a [GitHub account](http://github.com/), if you haven't yet

2. Fork the repository (http://github.com/zendframework/ZendSkeletonApplication)

3. Clone the canonical repository locally and enter it.

```bash

$ git clone https://github.com/zendframework/ZendSkeletonApplication.git

$ cd ZendSkeletonApplication

```

4. Add a remote to your fork; substitute your GitHub username in the command

below.

```bash

$ git remote add {username} git@github.com:{username}/ZendSkeletonApplication.git

$ git fetch {username}

```

### Keeping Up-to-Date

Periodically, you should update your fork or personal repository to

match the canonical ZF repository. Assuming you have setup your local repository

per the instructions above, you can do the following:

```bash

$ git checkout master

$ git fetch origin

$ git rebase origin/master

# OPTIONALLY, to keep your remote up-to-date -

$ git push {username} master:master

```

If you're tracking other branches -- for example, the "develop" branch, where

new feature development occurs -- you'll want to do the same operations for that

branch; simply substitute "develop" for "master".

### Working on a patch

We recommend you do each new feature or bugfix in a new branch. This simplifies

the task of code review as well as the task of merging your changes into the

canonical repository.

A typical workflow will then consist of the following:

1. Create a new local branch based off either your master or develop branch.

2. Switch to your new local branch. (This step can be combined with the

previous step with the use of `git checkout -b`.)

3. Do some work, commit, repeat as necessary.

4. Push the local branch to your remote repository.

5. Send a pull request.

The mechanics of this process are actually quite trivial. Below, we will

create a branch for fixing an issue in the tracker.

```bash

$ git checkout -b hotfix/9295

Switched to a new branch 'hotfix/9295'

```

... do some work ...

```bash

$ git commit

```

... write your log message ...

```bash

$ git push {username} hotfix/9295:hotfix/9295

Counting objects: 38, done.

Delta compression using up to 2 threads.

Compression objects: 100% (18/18), done.

Writing objects: 100% (20/20), 8.19KiB, done.

Total 20 (delta 12), reused 0 (delta 0)

To ssh://git@github.com/{username}/ZendSkeletonApplication.git

b5583aa..4f51698 HEAD -> master

```

To send a pull request, you have two options.

If using GitHub, you can do the pull request from there. Navigate to

your repository, select the branch you just created, and then select the

"Pull Request" button in the upper right. Select the user/organization

"zendframework" as the recipient.

If using your own repository - or even if using GitHub - you can use `git

format-patch` to create a patchset for us to apply; in fact, this is

\*\*recommended\*\* for security-related patches. If you use `format-patch`, please

send the patches as attachments to:

- zf-devteam@zend.com for patches without security implications

- zf-security@zend.com for security patches

#### What branch to issue the pull request against?

Which branch should you issue a pull request against?

- For fixes against the stable release, issue the pull request against the

"master" branch.

- For new features, or fixes that introduce new elements to the public API (such

as new public methods or properties), issue the pull request against the

"develop" branch.

### Branch Cleanup

As you might imagine, if you are a frequent contributor, you'll start to

get a ton of branches both locally and on your remote.

Once you know that your changes have been accepted to the master

repository, we suggest doing some cleanup of these branches.

- Local branch cleanup

```bash

$ git branch -d <branchname>

```

- Remote branch removal

```bash

$ git push {username} :<branchname>

```

## Conduct

Please see our [CONDUCT.md](CONDUCT.md) to understand expected behavior when interacting with others in the project.