Contributing via Git and Github

Git Manual

This is not a manual explaining how to use every feature of the Git program or the Github services with the forked and cloned OFBIz repositories.
If you are looking for that go to:

- Git: documentation
- Github: help

Work in Progress

This document is a work in progress.

This document is intended to capture all aspects related to contributing via Git and Github. While being WIP this document can contain questions (gathered from experience, and posts via mailing lists and the Slack channel) to be addressed.

Suggested approaches

By Jacopo: https://rocketmq.apache.org/docs/pull-request/

Related mail threads

- Git commit workflow for ofbiz
- OFBiz contributions & Github Pull Requests

Git Repositories

The project maintains several repositories:

Name	Description	Public location
ofbiz-framework	The repository containing all base components	https://github.com/apache/ofbiz-framework
ofbiz-plugins	The repository containing several plugins	https://github.com/apache/ofbiz-plugins
ofbiz-site	The repository containing the page of the official OFBiz site	https://github.com/apache/ofbiz-site
ofbiz-tools	The repository containing several functions and services to maintain the project and its works	https://github.com/apache/ofbiz-tools

Setting up your own Git Clone

Forking the OFBiz repositories

Forking the OFBiz repository in Github is - in essence - having your clone of the OFBiz repository in the Github environment, thereby being publicly available to the community and others.

This is done by clicking on the '**Fork**' button on the repository's page in Github (see public locations above).

From here you can clone or download it to your local development environment (and use it within your preferred IDE). You can do this via the green '**Clone or Download'** button on your repository page, or directly in your preferred location (folder) on your local environment, with a command as decribed in the 'Cloning your Github fork' section.

Cloning your Github fork

A Git clone is your remote git clone existing in your local environment.

Cloning is done through following commands in your cli:

Repo	command
ofbiz-framework	`git clone https://github.com/<your_github_id>/ofbiz-framework`
ofbiz-plugins	`git clone https://github.com/<your_github_id>/ofbiz-plugins`
ofbiz-site	`git clone https://github.com/<your_github_id>/ofbiz-site`
ofbiz-tools	`git clone https://github.com/<your_github_id>/ofbiz-tools`

Remark: I*t is advised to clone ofbiz-plugins within your ofbiz-framework clone as 'plugins' to ensure that you can both work on your plugin changes and use it in your ofbiz-framework environment. This can be done with following Git command:*

```
git clone https://github.com/<your_github_id>/ofbiz-plugins.git plugins
```

Invoking the git clone command as outlined above will ensure that your local clone has link to your Github repository. By invoking the **git remote -v** command you get an overview of all associated remote repositories.

Note that doing so will not ensure a synchronisation between the 2 repos (framework and plugins) when switching to release branches and you will need to manually switch plugins when switching framework. You will else encouter compiling issues due to the differences of codes in framework and plugins.

Keeping your local clone in sync with changes in the official repositories

Accepted commits will be merged regularly in one of the branches in the repository by one of the committers. In order to ensure that you're working with the latest from the official repository (to avoid merge conflicts), it is of the utmost importance that you regularly bring the changes from the OFBiz repository to your local clone.

In order to do this you need to set up a connection to the official repository. This is done by executing one of following git commands in your environment (in the folder of your local clone):

1. for ofbiz-framework: git remote add OFBiz https://github.com/apache/ofbiz-framework.git
2. for ofbiz-plugins: git remote add OFBiz https://github.com/apache/ofbiz-plugins.git
3. for ofbiz-site: git remote add OFBiz https://github.com/apache/ofbiz-site.git
4. for ofbiz-tools: git remote add OFBiz https://github.com/apache/ofbiz-tools.git

and update your local clone via **git fetch**, followed by **git rebase.** You may also use the convenient shortcut **git pull,** but then you need to use it with the *--rebase* parameter (git pull --rebase) or set rebase once for all: *git config --global pull.**rebase** true*

If you are a fan of the GitHub webapp UI, here is an article to keep your fork up to date https://github.com/KirstieJane/STEMMRoleModels/wiki/Syncing-your-fork-to-the-original-repository-via-the-browser
It works, but it's seems easier to use the CLI to directly update you local working copy: https://gist.github.com/CristinaSolana/1885435

Working on your enhancements for the project

Enhancements to the codebases of our project always start from somewhere. For our project this driven by one or more open issues (tickets) existing in our **Jira**. See also the section '**Reporting Code Issues**' below.

New development branches

When you are about to start working on a ticket, it is advised to create a new development branch based on one of the official branches. It is advised to name the branch in reference to the ticket, eg. the ticket id. As an example:

```
git checkout -b OFBIZ-12345
```

From Work-In-Progress to commit

Each time you save a code artefact in your development environment it isn't submitted as a commit to the branch. Instead it is saved as - until the commit action - a work in progress, and pushes to your remote repository (e.g your repository on Github) will not contain your saved and uncommitted changes.

In order to get your saved changes committed to the branch you have to invoke the **git commit** command. On committing your changes you are advised to set a commit message in accordance with the commit protocol as explained in OFBiz commit message template.

Squashing your commits

When working on a ticket in your local development branch it may be so that you have multiple commits before it is ready to be contributed to the project. In such a case and as a last step you should squash all your commits.

Remark: *When collaborating with fellow community members, you should **NEVER** squash the commits of those authors as it will rewrite history and the attribution is lost.*

Getting your contributions accepted

In order to get your contributions (commits) accepted by the project's committers, ensure that:

1. your code changes are in accordance to the project's coding guidelines, see Coding Conventions;
2. you have tested your changes from the cli with and they pass the integration tests (through the *testIntegration* Gradle task)
3. your commit message is in accordance with the project's commit-message, template, see OFBiz commit message template.

Making your commits available to community members (pushing to Github)

After having committed your changes to your local branch you may want to make those publicly available - in your public repository on Github - to your fellow community members for evaluation. This is done via the **git push** command, e.g

```
git push origin OFBIZ-12345
```

The example above sends your committed changes to your public repository on Github, as your local clone was established from there (see the '**Cloning your Github fork**' section above).

Pull Requests

> **Github Guide**
>
> Github workflow explanation: https://guides.github.com/introduction/flow/
>
> **TBD**: Pull request template: https://help.github.com/en/github/building-a-strong-community/creating-a-pull-request-template-for-your-repository

Pull requests are the means to inform fellow community members that your changes are available for review.

Pull requests (e.g. from Github) can be handled in the same way as patch files in a Jira ticket. In general, patch files are downloaded from the ticket and then applied to a branch in the development environment (e.g. your IDE) for evaluation and testing purposes.

Creating a Pull Request

After having you pushed your development branch to your public repository on Github, you are ready to submit a Pull Request. The best approach is to go into the branches section, where your development branches will be visible.

There you select the development you want to generate the Pull Request from, by clicking the '**New Pull Request**' button. This will open a dialog page where information regarding the Pull Request can be submitted. The dialog page allows you to select the branch (in most cases this should be the 'trunk' branch, which is the master branch of the repository) the Pull Request is applicable for.

Remark: *When the Pull Request is addressing an existing ticket in Jira, it is advised to put the ticket Id and the subject of the ticket in the subject of the request. This will ensure that the Pull Request is added as a link to the ticket and the 'Work Log' section of the ticket is updated*.

It is also advised that, for each of the tickets the pull request relates to, the ticketId (OFBIZ-xxxxx) is referenced with a explaining description. This ensures that the Pull Request can be associated in Jira to the correct ticket(s).

Handling a Pull Request (for committers)

A submitted PullRequest can be treated in a somewhat similar way as a patch file available in the Jira ticket. But, instead of having to create a new test branch in the local test environment where the patch file is uploaded, with a PullRequest this can be checked out directly in the local test environment as a new branch.

If after reviewing/testing the proposed commit doesn't pose a threat to breaking the code, and can be brought into one branches of the repository, the commit can be merged.

The commit can be merged into the correct branch(es) of the repository, after reviewing and testing the submitted change (validating that it doesn't pose a threat to breaking the existing code).

Remark

In order to see the PullRequest submitted to the official Github-repository of the project, it is required to have a location definition in your local clone to the official Github repository. Additionally you need to have following line added in that location: *fetch = +refs/pull/*/merge:refs /remotes/OFBiz/pr/**

Following example shows the excerpt of a git configuration for this:

```
remote "OFBiz"
url = https://github.com/apache/ofbiz-framework.git

fetch = +refs/pull/*/merge:refs/remotes/OFBiz/pr/*

fetch = +refs/heads/*:refs/remotes/OFBiz/*
```

Pushing commits

The Pull Request locally checked out must **NOT** be pushed back to the remote, as this will create an undesirable 'HEAD' pointer to the Pull Request.

Collaborating with community members

Collaborating to enhance the works of the project is paramount to its health. The project therefore welcomes community members to collaborate on larger issues. This can be easily done with Git/Github. When you want to work with a fellow community member on a larger issue, you can set up your local clone to also take in changes from the public fork of that community member.

Git GUI Clients

Instead of working via the cli (terminal) that comes with your OS, you can also use dedicated Git GUI clients (software packages) to execute git commands.

An overview of clients available (for your os platform) can be found here: https://git-scm.com/downloads/guis

Of course, you can also use the git functionalities available in your IDE of choice (Apache Netbeans, Eclipse, IntelliJ, etc.)

Reporting Code Issues

The project's tool to report and track progress on issues with code in the various repositories is Jira (https://issues.apache.org/jira/projects/OFBIZ).

While Github also offers functionality to report issues, there is no bidirectional way to sync these with Jira. Therefore, it is advised to register issues in Jira as tickets there appear in the project's mailing lists to reach its community members in a better way. And reference the ticket in save and commit actions (as per the commit-message template).