Git guide

Project 2 - How to use our git repository

Contents

1 Introduction

Git is a versioning system that can handle all the files of a project. It helps with tracking all the version and the modification by all the team members. Keep in mind that git is a very powerful tool and you should always pay attention when you use it.

If you have **ANY** doubt about GIT just ask the GIT Leader Riccardo (+39 3518878327) or the team leaders (Michele: +39 3487521909, Sofia: +39 3735325273).

2 How to clone the repo

In order to get the content of our git repository you need to clone it:

```
git clone <username>@atelier.inf.usi.ch:/home/lambrr/project_sa1.git
```

<username> is your short username (e.g. dallem)

3 File creation guidelines

Whenever you want to add a file, in order to keep the folder well organized you need to follow this guidelines:

- **Biography images**: The images of each Award winner's biography (e.g. Berners-Lee's biography) must follow the format:
  `./src/images/decades/<decade of the award>/<year of the winner's win>.(png or jpg)`

  e.g. ./src/images/decades/2010/2016.png (picture of Berners-Lee)

- **Award images**: The images of each Award winning reason (e.g. the invention of the World Wide Web) must follow the format:
  `./src/images/decades/<decade of the award>/award_<year of the award>.(png or jpg)`

  e.g. ./src/images/decades/2010/award_2016.png (image of the WWW)

- **HTML award page**: The HTML page for every year award (e.g. Award 2016 - Invention of WWW) must follow the format:
  `./src/html/awards/<decade of the award>/<award year>.html`

  e.g. ./src/html/awards/2010/2016.html

- **HTML decade page**: The HTML page for every decade (e.g. Decade 2010s) must follow the format:
  `./src/html/awards/<decade>/decade.html`

  e.g. ./src/html/awards/2010/decade.html

- **HTML biography page**: The HTML page for a winner biography (e.g. Berners-Lee biography) must follow the format:
  `./src/html/biographies/<winner_surname>.html`

  e.g. ./src/html/biographies/Berners_Lee.html

4 Sync your code

Before editing your local code, you have to be sure to sync your code with the remote code. In order to achieve that, you need to execute the command `git pull`, which synchronizes your local code with the remote. It's **very important** that you always pull before you start to edit your code.

5 Submit the code changes

Once you downloaded the directory you can start addeding pages, images and editing the files. **Remember** that you always need to follow the ***File creation guidelines*** (above) and the ***CSS guidelines*** (see specific pdf). When you finish editing your files you have to push to the git repo your changes in order to let the other members see your changes. The steps to push your changes are:

- **Add your changes**: `git add <list of files>`
  This command adds your changes to the current commit. You can add multiple files by adding a directory; you can also add multiple files by writing them inline. Remember to add **only** the files relevant for everyone (avoid adding temporary files, you personal notes, etc.).
  E.g. `git add ./file.html`
  `git add ./directory/`
  `git add ./file1.html ./file2.html ./file3.html`

- **Check the status of the commit**: `git status`
  This command checks the status of your commit, displaying the files not staged (files added to the commit but not still commited) and the files that are untracked (file that are not added to your commit). It also tells you if your local git repo is up to date with the remote repository. Use this command a lot: it's really useful.

- **Commit your changes**: `git commit`
  This command commits your changes, creating a commit in your local repository. This command commits only the files you added using `git add <files>`. This command opens the text editor with some lines commented with #. After this lines you can add your commit message.

It's **extremely** important that you write a complete and accurate message, specifying each modification you made. Use multiple lines to describe each change.

Wrong commit message: 'added a file'
Right commit message: 'Added file './.../file.html' containing the biography of Someone'

- **Push to the remote repository**: `git push`
  This command pushes your commits to the remote repository. Be sure that your commits are ok before pushing them using `git log`.

- **Check the log of the commits**: `git log`
  This command shows the log of the commit, with the commit messages. The '`origin/master`' commit is last remote commit, while the '`HEAD -> master`' commit is your last local commit.
  Once you push your commits to the remote repository, the `origin/master` commit and the `HEAD -> master` should be the same.

<u>DON'T:</u>

- Try to solve any problem by yourself: contact us!
- Create branches, force push or randomly merge
- Mess up with the GIT, it's extremely **risky**

If you have any doubt, question or problem don't try to solve it by yourself because you risk to ruin the ***whole*** group project. If you need anything just contact Riccardo, Michele or Sofia (phone numbers in the Introduction) and one of them will reply as soon as possible.