Environment guide-2016

GIT

Description

A modern, decentralized version control system.

Initial tasks (needs to be performed the first time when you start using GIT)

1. Install from http://git-scm.com/downloads
2. Create an https://www.assembla.com account. Please choose a username that contains your real name for easier identification.
3. Instructor adds you with read permissions to the https://git.assembla.com/epam-ui-2016.git repository.
4. Create a new GIT repository in Assembla. Let's suppose we created a repository named epam-ui-2016-john-doe.
5. Tell your GIT command who you are:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

6. Clone training source code from https://git.assembla.com/epam-ui-2016.git and go into the folder that is created as the result of cloning.

```
$ git clone https://git.assembla.com/epam-ui-2016.git
$ cd epam-ui-2016
```

7. Switch to the branch that contains the files for the current day. Let's suppose it is called lesson-1.

```
$ git checkout lesson-1
```

8. Create a remote that points to your own repository.

```
$ git remote add origin-own https://git.assembla.com/epam-ui-2016-john-doe.git
```

9. Push this branch to your own repository.

```
$ git push origin-own lesson-1
```

10. You can add new commits. When you want to push your new commits to your own remote repository, see Step 9.

Daily tasks (needs to be performed every day except the first)

1. Get the new branch for the current day.

```
$ git pull origin
```

2. Switch to the branch that contains the files for the current day. Let's suppose it is called lesson-2.

```
$ git checkout lesson-2
```

3. Push this branch to your own repository.

```
$ git push origin-own lesson-2
```

4. You can add new commits. When you want to push your new commits to your own remote repository, see Step 3.

Working from your home computer

1. If you would like to work on the code at home, you can clone your own repositry or if it is already cloned just pull the new branches.

```
$ git clone https://git.assembla.com/epam-ui-2016-john-doe.git
$ cd epam-ui-2016
```

or

```
$ git pull origin
```

2. Switch to the branch you would like to work on.

```
$ git checkout lesson-1
```

3. You can add new commits, then push them to your own repository.

```
$ git push origin lesson-1
```

Note: We did not create an other origin here because we are only communicating with one repository.

Commiting code changes

Here are some example commands that you will probably need. These commands are not described here in full detail, if you need more information about them, you can find a lot of good documentation online.

- Check what changed since your last commit and what changes are on the staging area.

```
$ git status
```

- Add changes to the staging area.

```
$ git add <file>
$ git add <directory>

// Add all changes to the staging area.
$ git add -A
```

- Commit staged changes.

```
$ git commit -m "This is the commit message"
```

- Unstage all changes.

```
$ git reset
```

- Remove a file and add this change to the staging area.

```
$ git rm <file>
```

Reference

- http://git-scm.com/book/en/v2/Getting-Started-About-Version-Control
- http://git-scm.com/docs
- http://marklodato.github.io/visual-git-guide/index-en.html
- http://ndpsoftware.com/git-cheatsheet.html
- http://justinhileman.info/article/git-pretty/git-pretty.png

NodeJS

Description

A javascript interpreter utilizing the Google v8 js engine, which is able to run javascript code outside browsers.

It is used for starting up the training backend application, also using its package manager npm to automatically setup the environment.

Tasks

1. Install nodejs from http://nodejs.org
2. Install training server code and development tools dependencies

```
$ npm install
```

3. Install gulp globally, so it can be run from anywhere

```
$ npm install -g gulp
```

4. Examine `package.json for training project settings`
5. (Optionally) Start node server by executing

```
$ node server.js
```

Quit by entering `CTRL+C`

Gulp

Description

Gulp is a build tool for javascript, it is used to define tasks which help executing the reoccurring tasks of a developer.

In the training project gulp is used for:

1. Compile Sass files into CSS
2. Check javascript errors
3. Watch sass and javascript and react when they have changed
4. Start up node server
5. Restart node server if `server.js` changes
6. Clean CSS files (rarely)

Tasks

1. Execute all tasks except cleaning CSS files (those are not triggered by the default task)

```
$ gulp
```

2. Recompile Sass files to CSS files (ie: sass task)

```
$ gulp sass
```

3. Check javascript files for syntax and common errors

```
$ gulp lint
```

4. Delete all CSS files.

```
$ gulp clean-sass
```

5. Monitor server.js and restart server if it is changed.

```
$ gulp nodemon
```

6. Start watching JS and SCSS files for changes and run lint/sass tasks if they are changed.

```
$ gulp watch
```

7. Examine `gulpfile.js`