November 11th, 2018

Preparing website for deployment

# Frontend fixes and changes

Revolution Slider

I dealt with the optimization of the revolution slider, making sure that all different screen sizes show optimized content. In order to change the layout for responsiveness I had to read the documentation of the revolution slider to make sense of the syntax used in the HTML code. The documentation was found in the Unify folder, namely in the vendor/revolutionslider/documentation folder. The documentation had an entry about responsive setup which explained the otherwise obscure syntax found in the HTML code:

<https://www.themepunch.com/revsliderjquery-doc/responsive-setup/>

Once the syntax was understood, I proceeded to optimize the size and position of images and text, while testing the result in 3 different screen sizes (mobile iPhone6/7/8, 11.5” laptop screen, and 1080p screen).

About us image

Changed the stock image to a GIF, which I feel transmits the process of building software better. Subject to change.

Service Descriptions

I changed the service descriptions to better reflect our services and strong points. The features remain ordered by length for a better visual display.

Client logos

The client logos were mistakenly changed in a for loop. This section will remain static, as it goes to show who we have worked with, and needs only contain logos.

Portfolio

The all portfolio button was removed since it is redundant and compromises the aesthetics of the site.

Admin Footer

I removed the unnecessary icons and links from the footer and changed the name to Forcewing.

**Total hours worked: 2h**

# Backend fixes and changes

Save picture

The func.py file contains the method that is used for saving pictures. The function save\_picture() is used for the blog, which uses square dimensions by default. This is not appropriate for the portfolio functionality where images have varying dimensions (since they are taken with screenshots or snipping tools). For this reason I created another function called save\_picture\_port() which simply saves the image without processing it further.

Blog

I removed some code that created directories from the new and update blog functionality. In my understanding there is no need for a directory tree under blog\_pics, as it simply makes the problem more complicated and harder to read the code. All blog photos will be stored in blog\_pics with their original name intact, and this should be enough to offer full functionality, without complicating the matter further. However, directory manipulation code should be saved for future reference, by finding it in earlier commits.

A question arises whether two separate forms are necessary for having CRUD functionality to Blog and Portfolio models. I refer to BlogForm and BlogFormUpdate. Reasoning should be given to support this decision. If not necessary, a refactoring is required.

New portfolio

The code is littered with new folders the purpose of which is not obvious. I implemented the client logo functionality and changed the code in many places. The code needs a full refactoring and a well-documented way to add multiple pictures to a table.

Some of the main changes have occurred in the portfolio model. A new field has been added for the portfolio images, thus condensing the new portfolio functionality in one step. The ImagesPortfolioForm has been removed. In the routes.py the naming convention has been adapted for a clearer code understanding. In both creating and updating the portfolios, the pictures must be present. When updating a portfolio, the images will be replaced with new ones. After updating the page will now be redirected to the portfolio section instead of the blog section.

Tag

The tag name is used to classify portfolio, mainly so the visitor is able to filter by tag. This is done through the data filter attribute in the HTML code:

*data-filter=".webapp"*

For this filter to work correctly the tag name must be lowercase with no spaces. This presents a conflict in the cases where the tag needs to be upper case with spaces, such as:



To solve this dilemma, we need two sets of tag identifiers. The proposed solution made use of converting all tags to lowercase with no spaces when the home page is called. This would sometimes trigger a database error due to the tables being locked and couldn’t be updated. To go around this problem, the tags were set to lowercase no space when they were created, instead of everytime the home page is called.

Showing portfolio on home page

In the model we have added a MainImage field in Portfolio, in order to facilitate showing the hero image on the home screen, since uploading multiple files has no clear order in the database.

Related items

This functionality is simple to do. After some googling, we import

*from* sqlalchemy.sql.expression *import* func

and

rand\_portfolios = Portfolio.query.order\_by(func.random()).limit(3).all()

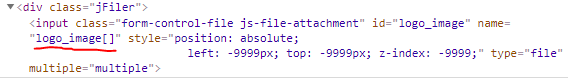
Next and Previous functionality left as exercise.

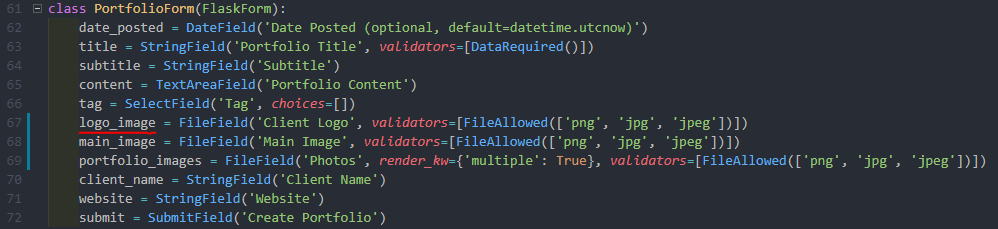
**Total hours worked: 5h**

General notes:

The naming convention should be properly followed. One particular instance is the liberal use of the term image\_file in 3 different forms with different purposes. This leads to extreme confusion and should be avoided in the future. Functional names like portfolio\_images, blog\_image, client\_logo\_image should be used instead (a functional name reveals the function in which the variable is used).

A persistent unwanted effect pops up in the js-file-attachment. Whenever we want to upload a picture using this library, we need to manually change the name of the input using JQuery. There is perhaps a better suited alternative. More testing and exploring should be done on that front, or fallback to using a different javascript upload library. The behavior can be seen in the following screenshots:





The JQuery code is found at the end of the main template, and without it the images do not get uploaded:

