Technical University of Moldova

CIM Faculty

FAF

Report

Integrated Development Environments

Laboratory work #4

Done by: student, gr. FAF-151

Bîzdîga Stanislav

Verified by: Patrașco Alex

Chișinău - 2017

Topic: Web Development

Prerequisites:

- IDEs: (one of) **Sublime Text 3**,RubyMine, PyCharm, Komodo, Coda, phpStorm, etc.
- Languages: Ruby, **Python**, JS
- Technologies and Frameworks: Ruby on Rails, **Django**, Node.js && Express

Objectives:

- Make a simple personal Web Site.
- Familiarize with HTML and CSS
- Javascript interactions.

Tasks:

Mandatory tasks:(mark = 6)

Simple site with 3 static pages. √ (/zen , /zen/1+, /zen/fun , /admin)

Optional tasks:

Intermediate(each task is worth 0.5 points):

- Your site must keep all site data in a database. √ (sqlite3)
- Write some unit tests and provide coverage. --

Advanced(each task is worth 1.5 point):

- Your site must contain AJAX Requests. ? (jQuery counts?)
- Your controllers must implement XHR or JSON responses. Some Data are dynamically loaded to the page. --
- Consume some API in your web app, make it useful(ish). ? (jQuery counts?)

Work processing

In this laboratory work, I had my first encounter with web development. Having a basic try with django a few years back, I decided it would be the go-to choice.

First of all, I had to understand the basics. I was going to implement a MVC based arhitecture which means: Model View Controller - a software design pattern for developing web applications which is made up of the following three parts:

Model - The lowest level of the pattern which is responsible for maintaining data.

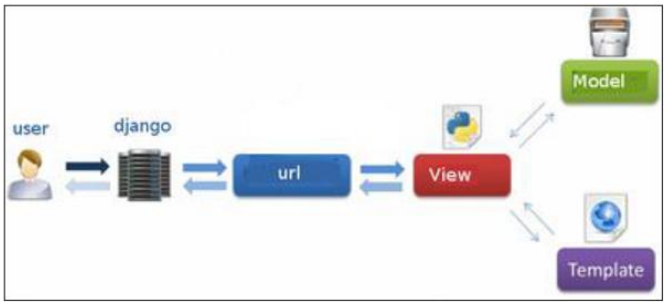
View - This is responsible for displaying all or a portion of the data to the user.

Controller - Software Code that controls the interactions between the Model and View.

Django manages this arhitechture by default since Django is a Python web framework like most other modern frameworks, thus it supports the MVC pattern. However, I ended up implementing an alternative, slightly different arhitecture: The Model-View-Template (MVT)

In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request:



The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

The first step to start a new Django project is to run in a CLI (ex.: cmd) the following command: `django-admin.py startproject project` which makes all the necessary initial directories and files.

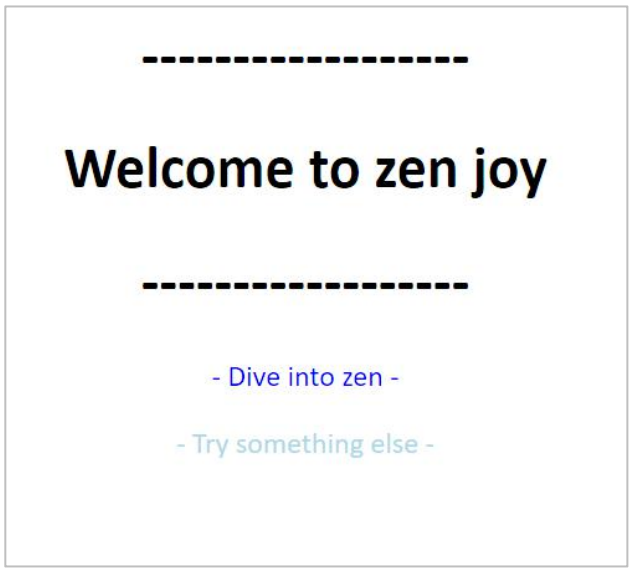
Next, I had to understand how to set up my project, which I did in the settings.py of my project. And once all was done, I could run my server locally. To do that I had to go to Command Line Interface (cmd) and type in: `python runserver manage.py` making sure I'm in the project directory. It would make a local server, that runs on the following address: `127.0.0.1:8000`

After that, I needed to understand the databases, particularly sqlite3 which django uses. Then I needed to figure out the HTML basics for the template use, some JavaScript, for implementing AJAX, which I'd implemented with ease thanks to jQuery (or would I?).

Product description:

Before I start, I feel the need to mention that the following site's goal is to be as much minimalist as possible. The reason being, Zen - it's a philosophy of mind stillness, clean of thougths, and full of calmness. Also, it's worth saying that I've uploaded the site to an international server for commodity.

Pictures:



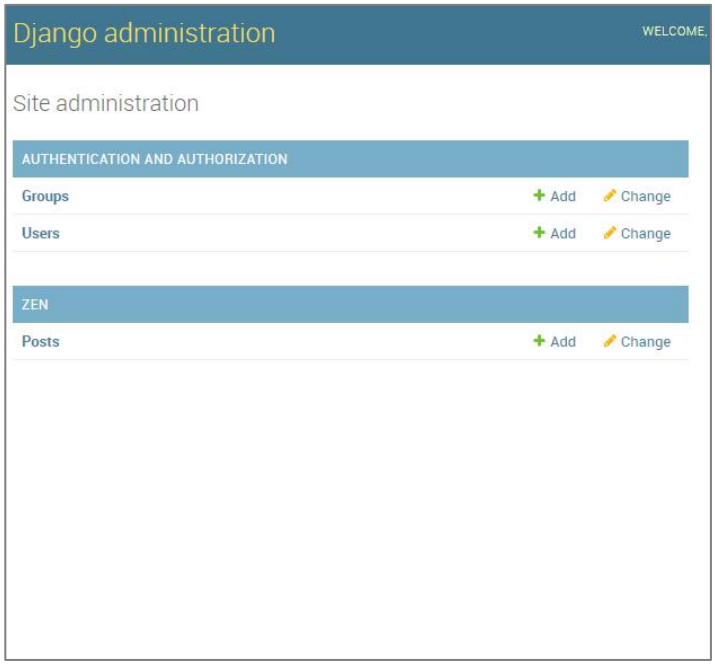
Main Page: http://azurium.pythonanywhere.com/

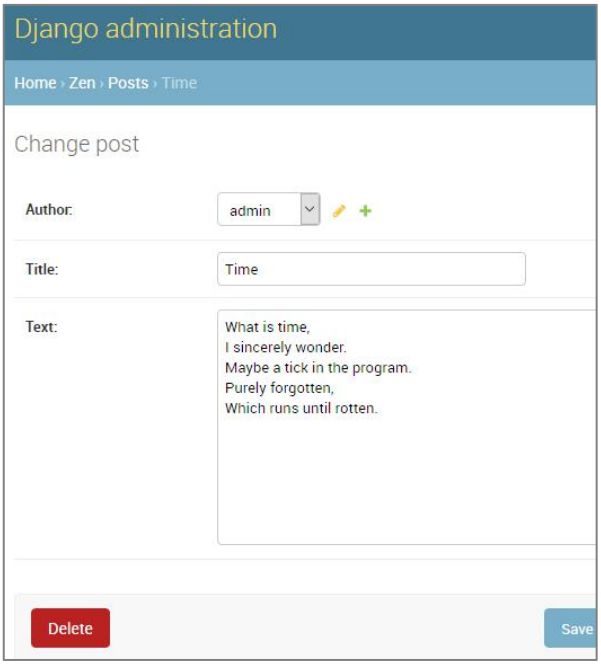
Features:

User may click **-Dive into zen-** to go to one part of the site, and **-Try something else-** for another.

The **zen** part is a little database that is composed of short texts that are meant to get the user into a zen state of mind, i.e. calm and relaxed, minimalistic, cleansing.

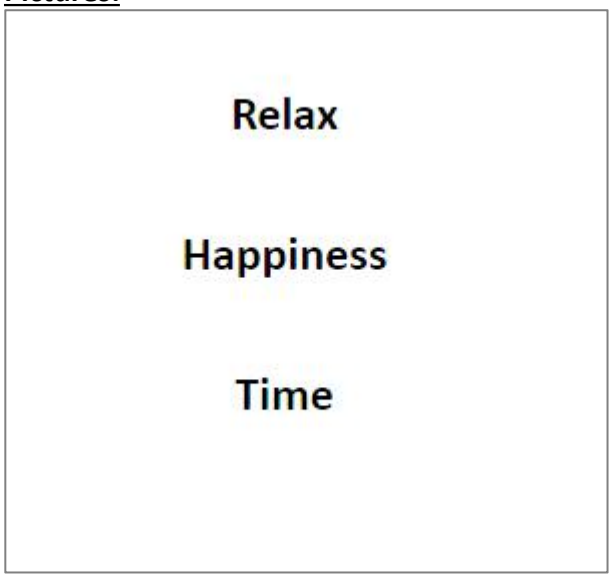
So far, only the admin can add such posts through accessing the admin page. Of course other super-users can be also manually added to have access to the admin page and add posts having their author name by their creation.





Admin Page: http://azurium.pythonanywhere.com/admin

Pictures:



Zen Page:
http://azurium.pythonanywhere.com/zen

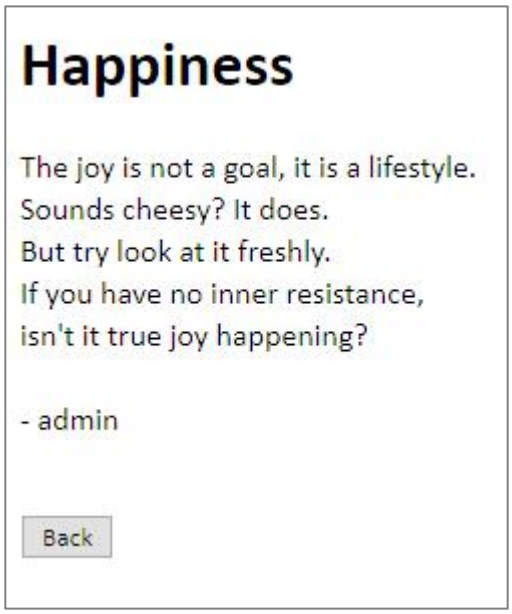
Features:

Here, we have a dynamic list that keeps going down in similar manner if you add more posts.

The user can click on the desired post name, and he will be redirected to it.

The content is centered, cleaned of href ugly purple and underlined style, and also of course, a cute new font.

I used inline CSS.

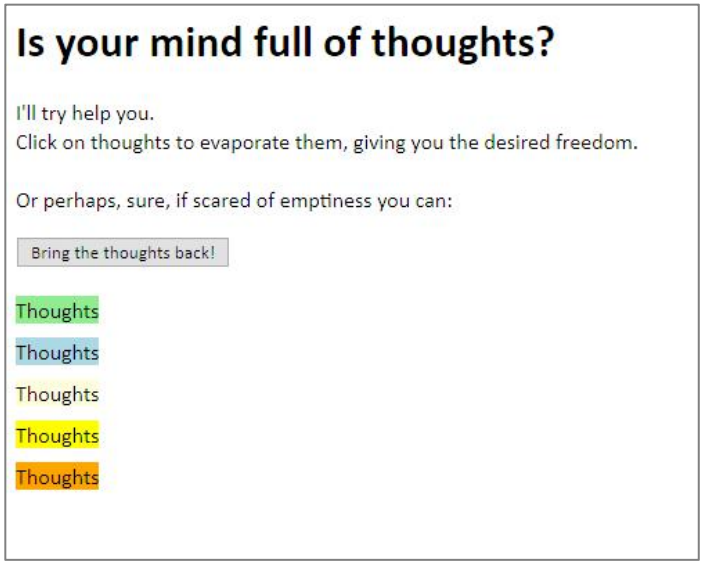


http://azurium.pythonanywhere.com/zen/2/

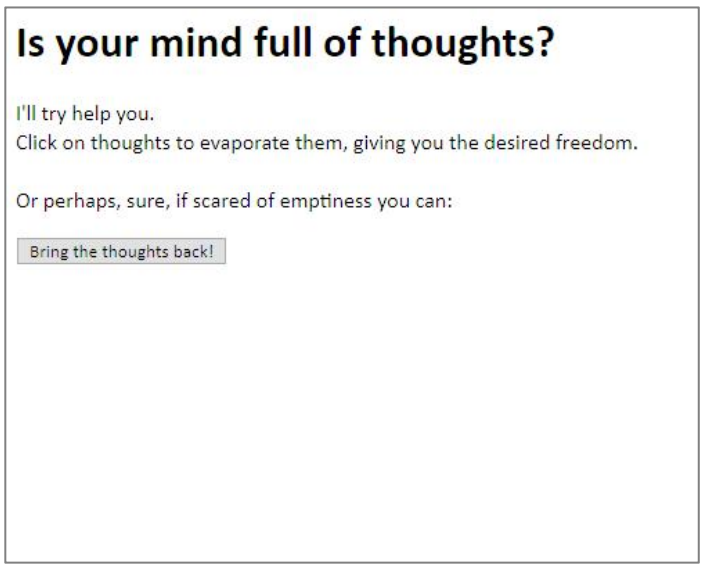
Here is one of the posts.
It's as any other post having its dynamic place, as a number after the **zen/** url space.
Also, here there's a back button to go back to **zen/** and be able to choose again among the list of zen-driven posts.

It displays the title, text and the author dynamically, without a need of manual tweaks. It's done by a html template with a for-loop inside.
(Didn't know you can do loops in html!)

Pictures:



Fun Page:
http://azurium.pythonanywhere.com/zen/fun



http://azurium.pythonanywhere.com/zen/fun
(after clicking on the "thoughts")

Features:

If user however clicks **-Try something else-** on the main page, he is taken here.

The **fun** part: this page is composed of short texts titled "thoughts" that are meant to be perceived as such - as in reality they are random and chaotic, they are represented in different colors and the user has the ability to click on them to make them go away, like he would during meditation obviously. This way, the user can get a feel of a zen state of mind.

Useful, isn't it?

You can also click the button to get the toughts come back, because let's be honest, they do come back A LOT.

These actions make the Thougts have animations. Firstly they disappear smoothly moving to the right. Then by the button click they return at their place, appearing slowly from the void.

These are functions that are made with jQuery. Cool thing.

Notes regarding the task:

I tried really hard to implement an ajax or xhr or json response but something was not right, I would even copy-paste a code following tutorials and it would not do the expected. For example I wanted to click a button to change a text in the page. It would respond with 200 (i.e state OK) while clearly it didn't find (should be 404) the source file(.txt, .php, .json) because it replaced the text with blank, and not only that, it doubled the content of the page. I tried so many ways around it, even jQuery's $.ajax(), and it gave the same outcome or even worse than that. In short though, AJAX is about loading data in the background and display it on the webpage, without reloading the whole page, so I hope that jQuery text fade in/out implementation counts as either an useful API or/and AJAX request.

Commentary:

I definitely learnt a lot, including the fact that I should have started with something easier than django, since it was my first web project. I also almost learnt the different response concepts (json, xhr, ajax) but sadly didn't get the required results due to some bugs.
I hope that the optional things would cover the tasks at least a little bit.
Try access the site! It's up and running :)

It was fun also to figure out how to manage the web app on the remote server, I connected it to my github and each time I would push my changes to git, I'd also have to run a bash at the pythonanywhere.com to pull the updates. And if I happened to delete some things (like the poll app I made following the django main tutorial), I would need to write on the server bash:

```
cd Lab4
git fetch --all
git reset --hard origin/master
```

Useful stuff!

And finally:

The Conclusion

IDE was a great course!
We had no time to laze around but needed to study different complex concepts alone.

Regarding the laboratory work:
This laboratory work was exciting. I studied about web development. Being a fan of python, I'm glad I learnt how to use it for this task, specifically, with **django**.

I was using a code editor called Sublime Text, rather than Notepad or Word. That is due to the fact that the code needs to be written in plain text, and the problem with programs like Word is that they don't actually produce plain text, they produce rich text (with fonts and formatting), using custom formats like RTF (Rich Text Format).

Also, code editors are specialized for editing code, so they can provide helpful features like highlighting code with colour according to its meaning, or automatically closing quotes for you, and auto-completing words.

I also learnt a lot of interesting things that are vital to web development.
I worked with SQL database, specifically sqlite3. I also worked with jQuery to make AJAX interactions way easier. Also I dove deep into html as templates.
It was a really really challenging and hard thing to be done in django, there's little documentation and PHP seemed to be the easier path to take. But I don't regret it.

Table of contents