Luca de Alfaro
@ UCSC

Search this site

Welcome!
Archived Projects
Classes
CMPS 121 - Mobile Applications
CMPS 183: Web Applications
CMPS 183 environment installation instructions
CMPS 183 Fall 2012
CMPS 183 Fall 2013
CMPS 183 Fall 2015
CMPS 183 Fall 2016
Homework 1
Homework 2
Homework 3
Homework 4
Lecture 1
Lecture 10
Lecture 11
Lecture 12
Lecture 13
Lecture 14
Lecture 15
Lecture 16
Lecture 17
Lecture 18
Lecture 2
Lecture 3
Lecture 4
Lecture 5
Lecture 6
Lecture 7
Lecture 8
Lecture 9
CMPS 183 Winter 2015

Classes > CMPS 183: Web Applications > CMPS 183 Fall 2016 >

Homework 2

Submission Instructions

Due: Friday October 14, 8pm.
Assignment site: https://www.crowdgrader.org/crowdgrader/venues/view_venue/

To submit, proceed as follows.

- First *pack* the web site. To this end, navigate to http://127.0.0.1:8000/admin/site (assuming you are serving the page from port 8000, the default), and for the Start application, click on Manage > Pack All. This will let you download a file called web2py.app.start.w2p. **Make sure your file name does not contain dashes '-'; underscores '_' are ok.**
- Then, go to this CrowdGrader assignment: https://www.crowdgrader.org/crowdgrader/venues/ , and choose the link to submit. Upload the file web2py.app.start.w2p as an attachment. Notes:
  - You need to log in using your @ucsc.edu account.
  - I used the class list as of Thursday September 29 to grant access to the CrowdGrader assignment. Let us know if you encounter problems.

You might want to consult the grading rubric while working on the assignment.

Assignment

You need to create a site composed of two pages.

Home Page

The home page behaves differently, according to whether the user is logged in or not.
Please look at the checklists application we are developing in class for an example of how to achieve this.

Not logged in

If the user is not logged in, at the top there should be a button to login (this is done for you in the starter code).
Below, the home page should display the list of the 20 most recent posts, with the most recent on top. You can find the instructions for doing so in the database abstraction layer chapter of the web2py book. For each post, you should display:

- Web2py logging
- Why web2py?
- CMPS 276 - Software Engineering
- CMPS 290G
- How to host a simple app on Google Appengine
- Raspberry PI Boot Camp
- **Collaborators**
- **Contact Information**
- **Creating and using a git repository**
- **Interested in UCSC?**
- **Is de Alfaro under A or D?**
- **Office Hours Appointments**
- **Past Projects**
- **Publications**
- **Publications (by topic)**
- **Resources**
- How to create a simple app on appengine
- **SlugIOT**
- **Talks**
- **TestDocs**
- **The Wikipedia Authorship Project**

External Links

- Resume
- Luca's personal home page
- CrowdGrader
- UCSC School of Engineering
- UC Santa Cruz

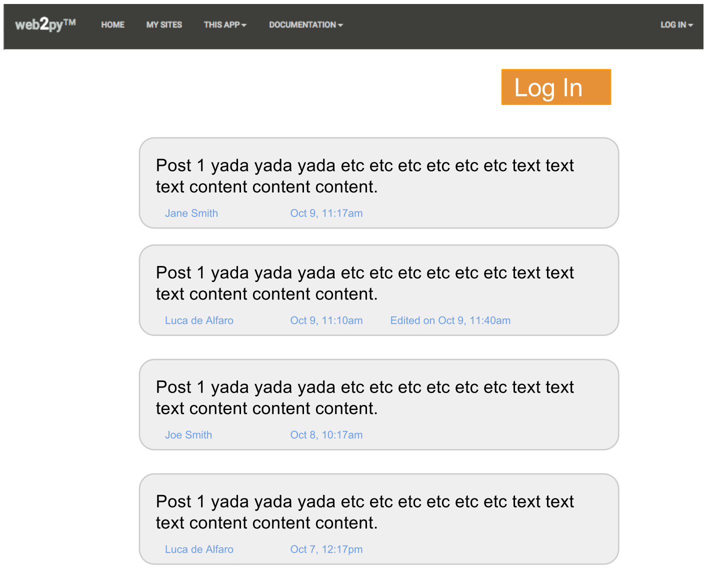
G+1 0

- The post content
- The author of the post (using first_name last_name). Note: see the function `get_user_name_from_email` in the starter code for how to get the user name given the email. There are better ways of doing this (using joins), but this will suffice for now.
- The date when the post was created
- The date when the post was edited, if different from the date when the post was created

See the drawing below for an example. Notes:

- You don't need to make your page look identical to this; this is a schematic representation only.
- The dates will display in UTC. This is intended. I will teach you later how to convert these to the user local time zone.
- Feel free to display the time in the best possible way, e.g., "12 minutes ago", "4 hours ago", "Yesterday 3pm", ... there are python libraries for converting from a datetime into a "humane" time representation. I leave this as an exercise.

Example of not logged in home page



Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Jane Smith Oct 9, 11:17am

Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Luca de Alfaro Oct 9, 11:10am Edited on Oct 9, 11:40am

Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Joe Smith Oct 8, 10:17am

Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Luca de Alfaro Oct 7, 12:17pm

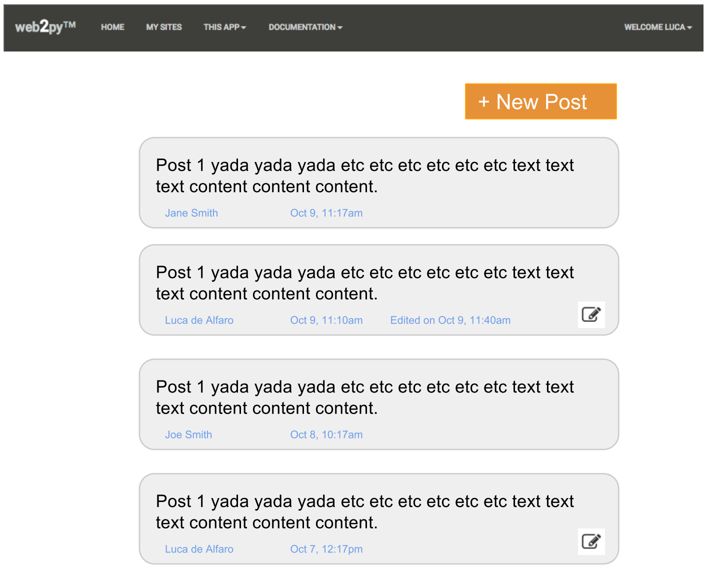
Logged in

If the user is logged in, the page should display at the top a button to add a new post (the button is there for you in the starter code). Below, as before, there should be the list of the 20 most recent posts, with the most recent on top.
For each post, you should display:

- The post content
- The author of the post (using first_name last_name)
- The date when the post was made
- The date when the post was edited, if different from the date when the post was created
- If the post was made by the author, a button to edit/delete the post.

See example below. Again, you don't need to make it look identical to this.

Example of logged in home page



Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Jane Smith Oct 9, 11:17am

Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Luca de Alfaro Oct 9, 11:10am Edited on Oct 9, 11:40am

Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Joe Smith Oct 8, 10:17am

Post 1 yada yada yada etc etc etc etc etc etc text text text content content content.
Luca de Alfaro Oct 7, 12:17pm

Note that in this example, the logged in user (Luca de Alfaro) can edit only the posts that he created, so the "Edit" icon is present only for those posts.
Note also how the edited date is displayed only when different from the creation date.

Page to Add/Edit/Delete Posts

This page should be accessed by a URL of type: `edit/<post_id>` , such that if the post_id is specified, the page will allow the user to edit or delete the post, and if the post_id is not specified, the page will allow the user to add a new post. You can create these URLs as follows:

- To create a post: `URL('default', 'edit')`
- To edit a post with id post_id: `URL('default', 'edit', args=[post_id])`

See here for more documentation on the URL helper. When the user accesses that page, the controller code should check the following:

- The user is logged in.
- If a post is specified, the post exists.
- If a post is specified, the post was created by the user who is logged in.

The page should display a form that enables the user to create / edit / delete the post; you can use SQLFORM for this. Once the post has been added / edited / deleted, you can redirect the user back to the home page. Bonus points: add a CANCEL button that enables the user to go back to the index without performing any edit.

Starter Code

You can find starter code in the `hw2` branch of our starter app. The code defines the database table for you, and it contains (empty) shells for the controllers. I did a bit of styling of the list of posts for you. Look in particular at `models/tables.py`, `controllers/default.py` and `views/default/index.html`, as well as `static/sass/_main.scss`.

Comments

You do not have permission to add comments.

Sign in | Report Abuse | Print Page | Powered By **Google Sites**