Luca de Alfaro @ UCSC

Search this site

Homework 4

Due: Tuesday November 29, 11pm
Submission instructions:

- **Assignment:** https://www.crowdgrader.org/crowdgrader/venues
- **Instructions:**
  - 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/ve and choose the link to submit. Upload the file web2py.app.start.w2p as an attachment.

The Assignment

In this assignment, you need to produce a single-page app, served by the default/index controller.
Ask users to login right away:

```
@auth.requires_login()
def index():
```

The app is a shopping cart app, and you can use our code in the cart-stripe-singlepage app as starting point.
While being served from a single page, it has to show THREE logical pages:

- A product list (as in the shopping cart app)
- The user's cart (as in the shopping cart app)
- The list of the user's past orders (this is new). You access this page by clicking a (new) icon on the top right that you will insert; please use the fa-clock-o icon.

The list of products

You can get the list of product on the server via:

```
import requests
r = requests.get("http://luca-
teaching.appspot.com/get_products")
```

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

You can then just return r.content, which is a string encoding the products in json, or you can do:

```
products = r.json()['products']
return response.json(dict(products=products))
```

You can also get the list of products directly from the client page, if you like.

Note that the list of products you get from the luca-teaching server does not include all fields; if you need additional fields, you can add them as done here.
In summary, you can likely use something like (I leave this up to you; this is a suggestion only, and I haven't tested it; it's just for your inspiration):

```
import requests
r = requests.get("http://luca-
teaching.appspot.com/get_products")
products = r.json()
for p in products['products']:
    p.desired_quantity = min(1, p.quantity)
    p.cart_quantity = 0
return response.json(dict( products=products, ))
```

Please *do not hardcode the list of products;* I will improve/change it before grading.

Stripe

To make it easier for you, no stripe integration is required. You can just assume that the Buy button buys the things in the cart, no checkout nor address etc. required.

The Order List

What is new compared to the app we did already is that you have to store the list of previous orders by the user.
This means that you need a table to store the user's orders, and when the user presses Buy, you have to store the content of the cart as something that has been bought, and you have to clear the cart.

To display the list of orders, you can do a nested loop. For each order, you create a header indicating the order date. Under that heading, you list the order, pretty much in the same way in which you list the cart (you can likely recycle then adapt the layout).

Summary of what you have to do

- Use stripe-cart-singlepage as the starting point branch. But note that you don't need to use its product table, since you are getting the products via a json call to http://luca-teaching.appspot.com/get_products. You also don't need all the stuff that has to do with Stripe integration (this to simplify the assignment).
- Create one table for the order history of a user. I have not checked, but perhaps this works:

```
db.define_table('orders',
    Field('user'),
    Field('created_on', 'datetime',
default=datetime.datetime.utcnow()),
    Field('order_json', 'text') # Order information, in json
)
```

- I think you need only the table above.
- Add a button to access the order history.
- Create a "virtual page" in index.html to display the order history. Just do a `<div v-if="page=='order_list'" id="order_list">` similarly to what was done for the other two pages.
- When a person clicks on "buy" in the cart page, you create a new order in the order history, and you clear the cart (and go back to the product list).
- When a person is on the order history page, fetch from the server the order history and display it.

Note that this has to be a **single-page app,** with no page loads at all except from the initial login.

Comments

You do not have permission to add comments.