Assignment 4

CSCI 4332

Due Nov. 15, 2015

Summary

In this assignment, you will modify our article database system from lecture to add some new features.

Your submission will consist of a zip file containing both your code and a PDF file `README.pdf` describing how you solved the problems.

For this assignment, you will be using **PostgreSQL**.

Do not modify any of the database tables to enable this project, as we need to be able to run it on our own copy of the database.

Your code should run on the department Linux servers with a virtual environment built from Python 3.4. We will grade it in a compatible environment.

Getting Started

Download the zip file `articleserver.zip`. This contains the database dump and source code. Unpack this and use it to get started.

After you have unpacked the code, you need to import the data into our PostgreSQL database on the department server.

You can do this by running the following command on Zeus (or similar):

```
zcat scidb-dump.sql.gz |psql -h postgresql.cs.txstate.edu
```

To keep from bumping in to your fellow students trying to run Flask servers on the same port on Zeus and other department servers, please configure your application to listen on a different port. Add the following line to your `settings.py`:

```
SERVER_NAME = 'localhost:5XXX'
```

Replace `XXX` with the last 3 digits of your student ID.

Part 1: Displaying Article Info (20 pts)

Right now, our article lists just show a bulleted list of article titles. This is not a very useful format for viewing articles. Modify the list of articles per issue to display citations in the following format:

> Halfaker, Aaron, Geiger, R. Stuart, and Terveen, Loren G. 2014. Snuggle: designing for efficient socialization and ideological critique. In *Proceedings of ACM CHI 2014 Conference on Human Factors in Computing Systems.*

In addition, modify the article display to display the article's basic information in this format as well, so that we have a citation rather than our current bulleted list of authors below the heading.

A few notes:

- The title should be a hyperlink to the article's page (`/articles/<aid>`).
- If the issue has a title, print the issue title as above; if the issue has no title, create a title from the publication title, issue volume, and issue number.
- If there are only 2 authors, separate them by 'and' with no comma; if there are 3 or more authors, separate them by commas with a comma and the word 'and' before the last one (Oxord comma).

  Note: This requirement is less important than the rest of the feature. If you implement it correctly, but just have all authors separated by commas regardless of number, you'll get 18 of the 20 points.

In your `README.pdf`, describe the following:

1. The query or queries you used to retrieve the data (include the SQL and a brief description of what the query does).
2. What parts of the formatting you did in SQL, what parts in Python, and what parts in the Jinja2 template.

I recommend doing this with a Jinja2 Macro to make easy to reuse your template logic in other parts of the program. You can create a macro by creating a new file called `widgets.html`, in the `templates` directory, with the following content:

```
{% macro citation(article) %}
{{ article.title }} <!-- all your HTML and template code for displaying an article goe
{% endmacro %}
```

Then, in the `article.html` or elsewhere, you can write:

```
{% import 'widgets.html' as widgets %}
<p>Cite as:</p>
<p>{{ widgets.citation(article) }}</p>
```

This feature lets you reuse template snippets throughout your application.

Part 2: Article View (30 points)

Right now, we do not have any way to list the articles written by an author.

1. Create a new type of page, with URL `/authors/<int:aid>`, to display the articles written by an author. This page should give the author's name in the heading at the top, a count of the number of articles they have written, and then list all their articles in the citation format from Part 1 in decreasing order by date.
2. Modify the citation search to display turn each author's name into a hyperlink pointing to their author page. This means that the `article` objects/maps you prepare for the template will need to contain a list of authors, each of which has both the author name and author ID. Alternatively, you can pre-render the author list HTML in Python before sending it to the template.

In your `README.pdf`, include the query you use to retrieve the results for an author page.

Part 3: Author Leaderboard (10 points)

Create a new page, `/authors/leaders`, that displays a list of authors with the number of papers they have published, with the authors with the highest publication counts first. Create a link to this page on the front page of the application, called 'Authors by publication count'. Each author name should be a link to their author page.

Extra Credit: Pagination (5 points)

For 5 points of extra credit, *paginate* the author leaderboard. The leaderboard, when you visit it, should display 50 authors and a 'Next Page' link. The 'Next Page' link should show the next 50 authors.

Your URL format can use a *query parameter*, so the URL for page 2 will look like `/authors/leaders?page=2`. Consult the Flask documentation for how to read query parameters.

Pages 2 and later should also have a 'Previous Page' link that goes back to the previous page.

Submitting

Submit your ZIP file to TRACS.