CODING NOMADS

15) User Profiles Lesson

Add User Profile Information

13 min to complete · By Brandon Gigous

Contents

The app that you've been making in this Flask course is a social app. What's more ubiquitous in a social app than user profile pages? Not much, so in this section you'll be diving into making a profile pages for users, allowing them to edit their profile, and allowing admins to edit profiles, too. In this lesson, you'll start with the profile page.

Adding User Profile Information

The first thing you'll need to do is to add columns to your `User` model. Adding additional user information to the database makes displaying each user's unique information onto a template much easier:

```
class User(UserMixin, db.Model):
    # ...
    name = db.Column(db.String(64))
```

```
    location = db.Column(db.String(64))
    bio = db.Column(db.Text())
    last_seen = db.Column(db.DateTime(), default=datetime.utcnow)
```

These new fields will store a user's name, location, a small autobiography, and date of last visit. The `bio` field is type `db.Text` and these types are used for longer strings. The `last_seen` field is of type `db.DateTime`, and its default keyword argument takes a *function*. That's why the `utcnow` function isn't called, it will be called when the default column value is assigned upon the creation of a new `User`. All of these fields, except `last_seen`, are up to the user to provide.

Pinging Your Users

After reading the word "ping" you may have thought back to the days of online gaming where you'd try to join the servers with the lowest ping. At least I was, but the ping in this case is just a simple function you'll make for the user. Whenever they make a new request on your server, you'll call this function to update the user's `last_seen` field:

```
class User(UserMixin, db.Model):
    # ...
```

```
    def ping(self):
        self.last_seen = datetime.utcnow()
        db.session.add(self)
        db.session.commit()
```

The best place to call this new function? Well, which handler gets called every time a request is made? That's right, the new `before_request()` function that you made in the previous section.

```
@auth.before_app_request
def before_request():
    if current_user.is_authenticated:
        current_user.ping()
        if not current_user.confirmed \
                and request.endpoint \
                and request.blueprint != 'auth' \
                and request.endpoint != 'static':
            return redirect(url_for('auth.unconfirmed'))
```

Now every time a request is made and the user is authenticated, it will update that user's `last_seen` time. There's of course no point in doing it if there's a user that's not signed in!

Get Mentorship

Flask-Moment

Before you start fleshing out your user pages, you'll need to take a *moment* to think about how you're going to deal with time. Every user on your site is probably not in the same time zone, which complicates the way you'll show them what time another user was last seen, or just dates and times in general. Typically, what servers do is use Coordinated Universal Time (UTC) that is independent of time zones. But this time is not something you want your users to see. It may be confusing for them to have to do the math in their head for what time something happened in *their* time.

Since probably no one likes programming time zone logic, some other folks made the sacrifice and created Moment.js to share with the world. It's an open source JavaScript library that is able to take the UTC time from the web browser, originally sent by the server, and converts it to local time for the user.

Lucky for you, you don't need to worry about JavaScript this time. There's Flask-Moment for that, a Flask extension that integrates Moment.js with Jinja templates. So, grab it with:

```
(env) $ pip install flask-moment
```

It's initialized like so:

```
from flask_moment import Moment
# ...
```

```
moment = Moment()
# ...

def create_app(config_name):
    # ...
    moment.init_app(app)
```

Almost done, just one more thing before you get started with your user pages. Flask-Moment depends on jQuery.js as well as Moment.js. What that means is they need to be somewhere in the HTML document of your templates. The easiest way to make that happen is to use the extension's helper functions. These functions reference the tested versions of the libraries, but since Bootstrap already uses jQuery.js and includes it into your HTML documents, only Moment.js needs to be added to the document.

To put it in your own documents, all you need is this one simple trick (in your `templates/base.html`):

```
{% block scripts %}
{{ super() }}
{{ moment.include_moment() }}
{% endblock %}
```

Then, whenever you need to display a time, Flask-Moment has you covered with a `moment` object conveniently automatically passed into templates. You'll see an example in just a bit.

User View Function

Perhaps you already made a view function for user accounts. Let's make it better by adding one that grabs a user from the database, or bust:

```
@main.route('/user/<username>')
def user(username):
    user = User.query.filter_by(username=username).first_or_404()
    return render_template('user.html', user=user)
```

With this dynamic route, the view function will look for the user specified and if not will return a 404 response with Flask-SQLAlchemy's `first_or_404()` query object method.

User Template

```
{% extends "base.html" %}
{% block title %}{{super()}} User {{user_name}}{% endblock title %}

{% block navbar %}
    {{ super() }}
{% endblock navbar %}

{% block page_content %}
{{ super() }}
<div class="page-header">
    <h1>{{ user.username }}</h1>
    <table class="table">
        <tbody>
        <tr>
        <th scope="row">Name</th>
        <td>{% if user.name %}{{ user.name }}{% endif %}</td>
        </tr>
        <tr>
        <th scope="row">Location</th>
        <td>{% if user.location %}{{ user.location }}{% endif %}</td
        </tr>
        {% if current_user.is_administrator() %}
            <tr>
            <th scope="row">Email</th>
            <td><a href="mailto:{{ user.email }}">{{ user.email }}</
            </tr>
        {% endif %}
        <tr>
        <th scope="row">Bio</th>
        <td>{% if user.bio %}{{ user.bio }}{% endif %}</td>
        </tr>
        <tr>
        <th scope="row">Last seen</th>
        <td>{{ moment(user.last_seen).fromNow() }}</td>
        </tr>
```

```
            </tbody>
        </table>
    </div>
    {% endblock %}
```

Other than using a Bootstrap table for the profile fields, this is your run-of-the-mill template that you've seen or made a few times now. However there are a couple things to note. The first is that the user's email is only exposed if an admin is signed in. Additionally, it's a "mailto" link so that an admin can more easily email the user if necessary. The second is the use of the `moment` object mentioned before. To display the displayed user's last seen time in the current user's local time, `user.last_seen` is passed to the `moment` object. then `fromNow()` is invoked to calculate the time between then and now.

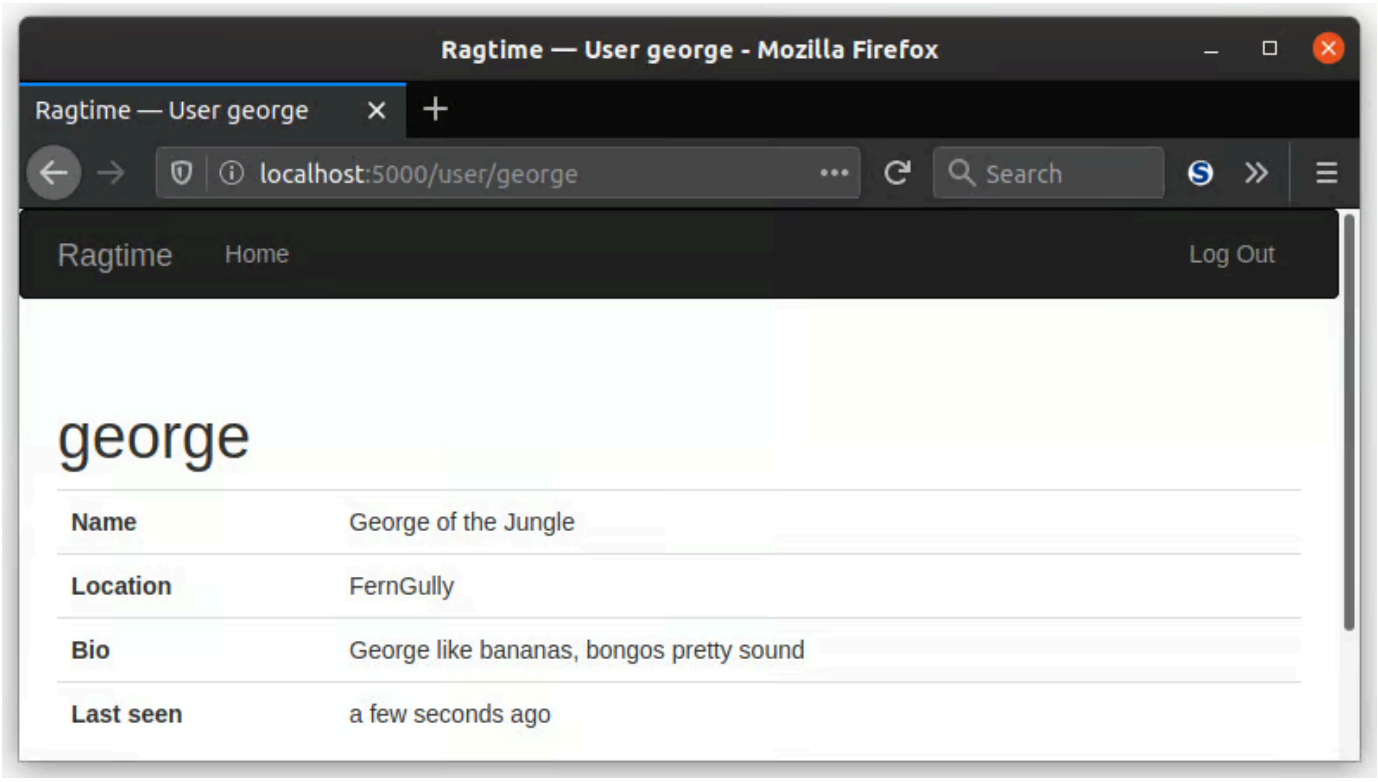
Profile Link

Since a user will probably want easy access to their profile, add a link to the navigation bar:

```
{% if current_user.is_authenticated %}
<li>
    <a href="{{ url_for('main.user', username=current_user.username)
        Profile
    </a>
</li>
{% endif %}
```

Any unauthenticated users will also see the navigation bar, but there's no reason for them to have a profile page, and thus there's no link for them to a profile page. That's why the check is made here.

Your profile and navbar should look something like this:



Fantastic, you're off and ready with a user profile page! Like any social media site, if you spell something wrong or forget to add a detail, it's good to be able to edit your own

profile. So that's what you'll let users do in the next lesson!

Summary: Adding Info to a User Profile in Python + Flask

- To add user profile information, you can add columns to your user model like name, location, bio, and last_seen
- Ping is a function you'll make for the user. Whenever they make a new request on your server, you'll call this function to update the user's `last_seen` field
- Moment.js is an open source JavaScript library that is able to take the UTC time from the web browser, originally sent by the server, and converts it to local time for the user.

Previous Next → Add a Profile Editor