arne brodowski

Tracking user-activity with Django

First I would like to describe what this entry is not about. It is not about tracking users in the Google Analytics way. This piece of code will not allow you to track every action a user makes and so on. It fullfills an other need. What I wanted for a website was the posibility to messure, if a user is active - in terms of: visiting the site often - or not. Some of you may know this feature from the Xing (http://www.xing.com/) website, where each userprofile page shows an indicator between 0 and 100% indicating if it's an active user.

My first thought was I could use the last_login field of the User model, but this only gets updated, when the user actually logs in - not when the session remains active for a couple of days or weeks.

So my aproach works as follows. Every User already has an related UserProfile model, I added an last_active DateField to the UserProfile model an wrote a small middleware to update this field. The field is only updated once per day, if the user is browsing the site, to keep database writes at a minimum and because shorter intervals don't lead to a more precise activity indiciation. My definition is: if the user was on the site on a given day, then his activity is updated to 100%. The activity then decreases for every day the user is not visiting the site.

The field added to the UserProfile looks like this:

```
last_active = models.DateField(null=True, blank=True)
```

The middleware looks like this:

```
import datetime
from project.profile.models import UserProfile

class ActivityMiddleware(object):
    def process_request(self, request):
        if request.user.is_authenticated():
            today = datetime.date.today()
            profile = request.user.get_profile()
            if profile.last_active is None:
                #workaround for profiles not having last_active set
                try:
                    profile.last_active = today
                    profile.save()
                except:
                    pass

            if profile.last_active is not None \
                 and profile.last_active < today:
                # update last_active
                profile.last_active = today
                profile.save()
```

The middleware works by intercepting each request and checking if the user is authenticated. If the user is not authenticated, it does nothing. If the user is authenticated the middleware checks if the last_active field has a value older than today (or no value at all). If that's the case, last_active gets updated to today's date and no further updates will occur on this day, while the user browses the site.

Now only one final step is missing. We need a way to query a user for it's activity value. I solved this by adding a property to the UserProfile model which is responsible for calculating an activity percentage between 0 and 100 out of today's date and the last_active date from the model.

The code for the activity property looks like this:

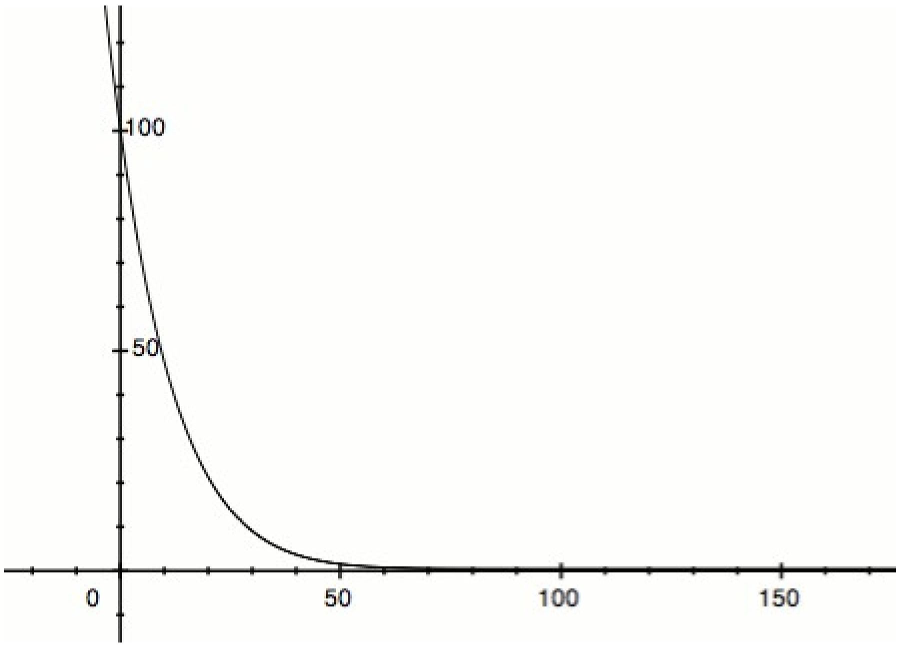
```
def _activity(self):
       '''returns a value between 0 and 100
       indicating the useractivity'''
       today = datetime.date.today()
       if self.last_active is None:
           return 0

       if self.last_active >= today:
           #just in case
           return 100

       else:
           diff = today - self.last_active
           if diff.days < 87:
               activity_ = int(((diff.days/87.)-1.585)**10)
           else:
               activity_ = 0
           return activity_

activity = property(_activity)
```

The only tricky part here was finding the formular for calculating the activity:

```
activity_ = int(((diff.days/87.)-1.585)**10)
```

If the user is inactive for 87 or more days I return 0 instead of the calculated activity, because the graph of the function is sort-of hyperbolic and will return values greater 0 at very high values of x (e.g. diff.days).

It would be possible to just use a linear mapping between 'days since last visit' and 'activity in percent', which yould mean, that a user beeing absent for 99 days would get an activiy of 1%, a user being absent for 50 days would get an activity of 50% and a user being absent for 1 day would get an activity of 99% - you get the point. The formular I'm using looks like this:

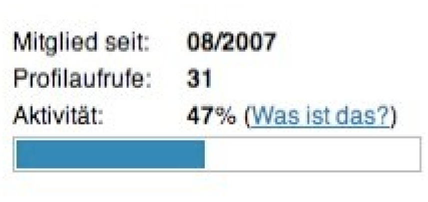


This must not represent the ways you would like to map days to activity, but it fits pretty good for my purpose.

To display the activity on the user's profile page the `widthratio` template`` tag comes really handy. The coresponding part of my template looks like this (profile is the UserProfile object of the user who is displayed on the page):

```
<div>Activity: {{ profile.activity }}%</div>
<div style="line-height:14px;font-size:12px;width:200px;
    border:1px solid #aaa;padding:1px;">
    <div style="background-color:#07a;
               width:{% widthratio profile.activity 100 200 %}px;">
    &nbsp;</div>
</div>
```

In the context of an application I'm currently working on it looks like this (text in german):



Mitglied seit: 08/2007
Profilaufrufe: 31
Aktivität: 47% (Was ist das?)

This solution serves my purpose pretty well, what do you think about it, are there any better ways? Or any way to improve this solution? Any comments are highly appreciated!

Veröffentlicht von Arne Brodowski am 16. Jan. 2008, 17:44 in django, middleware, python, tracking, user-activity.

Kommentare

Damn, if I had any registered Users on my website, I would love to implement this... even if I don't really recognize what "int(((diff.days/87.)-1.585)**10)" means. :-)

Geschrieben von Martin (http://www.mahner.org/) 1 Tag, 19 Stunden nach Veröffentlichung des Blog-Eintrags am 18. Jan. 2008, 12:47. Antworten (#comment-form)

Doesn't your formula mean that if the user was absent for 2 years and then he is using the system for 2 days, his activity will be 100%?

Also, I wouldn't repeat the request.user.get_profile() although it returns cached info, because Python is interpreted language and each traveling through object trees adds executing time.
Better assign
profile = request.user.get_profile()
and then use it as necessary.

Geschrieben von Aidas Bendoraitis [aka Archatas] (http://aaiddennium.com/blog/) 4 Tage, 21 Stunden nach Veröffentlichung des Blog-Eintrags am 21. Jan. 2008, 14:45. Antworten (#comment-form)

Aidas: thats right. Everytime the user visits the page - no matter how long he's been absent - his activity gets reset to 100%. It depends on what you want to express with the "activity", for my usecase this fits good, activity in my case is more or less an indicator of how likely it is to (virtually) meet the user in the social network.

Calling get_profile() only once is a good point, I will change the code accordingly.

Geschrieben von Arne (http://www.arnebrodowski.de/blog/) 4 Tage, 21 Stunden nach Veröffentlichung des Blog-Eintrags am 21. Jan. 2008, 15:20. Antworten (#comment-form)

Hey Arne,

I just saw this entry, pretty good work. I'm not sure, maybe I'll use this for my new project.

By the way, I wouldn't write a function called "_activity" when you want to declare it as property. Either you use "_get_activity" (this enables you to create another function "_set_activity") or you use:

```
@property
def activity(self):
    pass
```

Usually I use "_get_activity", as it is very common and enables you to set the ``short_description``-property for this function and call it in admin's ``list_display``.

Anyway, keep the great work!

Cheers from Cologne
Martin

Geschrieben von Martin Geber (http://www.martin-geber.com/) 3 Monate, 1 Woche nach Veröffentlichung des Blog-Eintrags am 23. April 2008, 16:54. Antworten (#comment-form)

Hi Martin,

thanks for your feedback.

I didn't used _get_activity because there will never be a _set_activiy method, because the activity-value is a computed value. But you are right, using _get_activity as the name would be more obvious.

I didn't used the @property syntax, because if I remeber it right, the @-decorators where added in Python 2.4 and Django tries to stay Python 2.3 compatible if possible, but that's more an excuse than a real reason.

Geschrieben von Arne (http://www.arnebrodowski.de/blog/) 3 Monate, 1 Woche nach Veröffentlichung des Blog-Eintrags am 23. April 2008, 17:48. Antworten (#comment-form)

The user object already has a last_login field, why bother to make another on in the profile?

Geschrieben von Philipp 2 Jahre, 4 Monate nach Veröffentlichung des Blog-Eintrags am 1. Juni 2010, 13:23. Antworten (#comment-form)

Hey Philipp,
SESSION_COOKIE_AGE is 2 weeks by default in Django.
So if you will log into site and will visit it every day your last_login date will be not changed.
It will be changed only when you log out and log in again.

Geschrieben von Viacheslav Svyrydiuk (http://korman.cv.ua/) 4 Jahre nach Veröffentlichung des Blog-Eintrags am 27. Jan. 2012, 19:48. Antworten (#comment-form)

Suche