ECS 89

Announcements

- Final Django code due tomorrow night
  - Form enter user data, put it into User table - this code is now **available** on the project Web page
  - Python program to put data from steps.csv into Pedometer table – discussed last time
  - Form to display combined User and Pedometer data
- Prof. Amenta extra office hour Mon 2-3
- Jesse's regular office hour Mon 4-5

Next steps

- We want to add in the pedometer data
- This is coming from steps.csv
- We can write a regular Python program to load objects into Django databases
- Put the program in /var/www/yourname/mysite

Load Django, settings, our classes

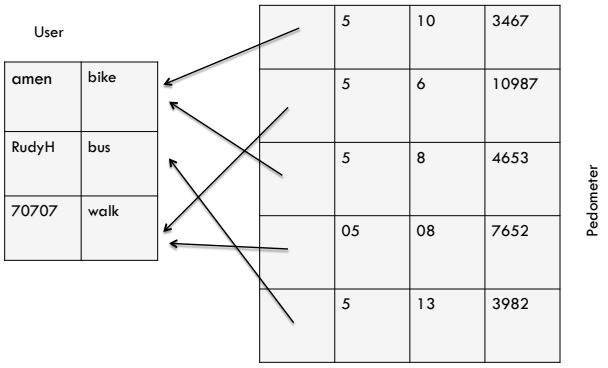
```
from django.core.management import setup_environ
from mysite import settings
setup_environ(settings)
from steps.models import User, Pedometer

# We can now write a normal python program that accesses
# our Django database.  For example:
#u = User(uid="gump", transport="bike")
#u.save()
```

Class for pedometer data

```
class Pedometer(models.Model):
    user = models.ForeignKey(User)
    steps = models.PositiveIntegerField()
    month = models.PositiveIntegerField()
    day = models.PositiveIntegerField()
```

- The ForeignKey function indicates that this attribute is a relation to a row of the User table
- If the User is not in the user table, add them in, with a default transport mode of "walk"

Our database setup



Query by user

Pedometer System User Lookup

Enter a User ID [ ] Submit

Reponse

Pedometer System User Lookup

User amen travels by bike.

Daily steps:
3046 steps on 5/5
9176 steps on 5/7
3612 steps on 5/8
14388 steps on 5/9
3154 steps on 5/10
7327 steps on 5/11
5045 steps on 5/12

Ask about another user

Response

- Includes data from both tables
- First, get the User data
- Use the get function; for instance, say you have put the uid from the HTTP GET request into variable queryID:

```
u = User.objects.get(uid=queryID)
t = u.transport   # get the mode of transportation
```

Database get function

- Returns an object containing a data row
- Raises an exception if there is no row that matches the condition, or if there is more than one.
- So it has to sit in a try-except construction!

- This is the obvious approach for the User data, but how about the Pedometer data?
- Let's review our options.

The all function

```
ps = Pedometer.objects.all()
```

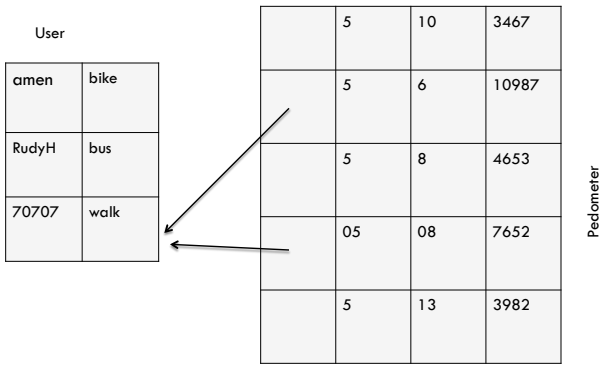
- Gets data in all rows of pedometer table.
- ps will contain a QuerySet (basically a list) of pedometer objects, one for each row.
- We could read through the list and find all whose user object was equal to u (the one we got out of the user table).
- Pros/cons?

The filter function

ps = Pedometer.objects.filter(user__uid=queryID)

- Gets only the Pedometer records where the uid of the pointed-to user matches the queryID (see pix next slide)
- Digs through whole Pedometer table
- Might have faster implementation
- Pros/cons?

So if queryID was 70707…



The pedometer_set function

ps = u.pedometer_set.all()

- Goes through the user object, traverses the arrows backwards
- Should be very efficient
- This is such a common operation there is extra support for it.