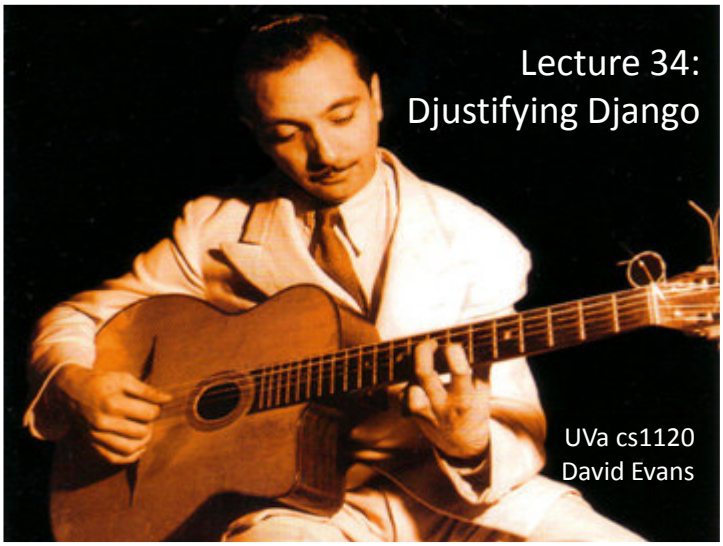


Menu

- Django Innards
- Sign up for design reviews in class today

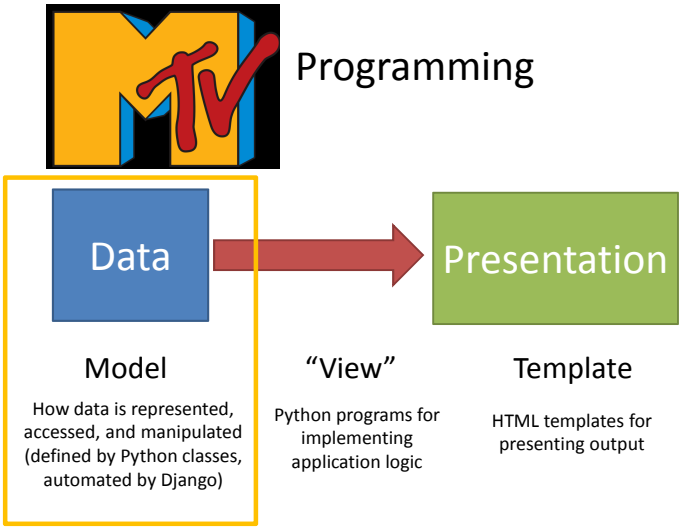
Exam 2 Review Session: tonight, 6-7:30 in Olsson 001

Wednesday, 5-6:30pm, Olsson 228E: Fireside chat with Wes Weimer and Kim Hazelwood

Projects and Teams

- Project teams are posted on website
- Everyone should be happy with their team – if you're not happy, let me know **today**!
- At the end of the project you will rate each of your teammates on:
  - Effort
  - Contribution
  - "Works well with others"
- All team members are responsible for the success of the project

Programming



Web Application Data

Client (Browser)

Server

GET /overheardit HTTP/1.0

<html>
<head>
…

Contents of file

HTTP protocol is *stateless*: each request is independent

For data to persist across requests, needs to be either:
stored on server (next: database)
or embedded in request (later: cookies, form fields)

Django Data Models

overheardit/stories/models.py:

```
class Story(models.Model):
    body = models.CharField(max_length=500)
    upvotes = models.IntegerField(default=0)
    orig_poster = models.CharField(max_length=15)
    post_date = models.DateTimeField('date published',
                                     auto_now=True)
```

class variables: each instance gets its own copy (like an instance variable) that is assigned to real value or corresponding type

```
] python manage.py shell
Python 2.6.1 (r261:67517, Dec  4 2008, 16:51:00) ...
>>> from overheardit.stories.models import Story
>>> Story.objects.all()
[<Story: Brr!  Its cold here.>, <Story: It is intuitively obvious that (lambda (n) ((lambda (f) (f f
n)) (lambda (f k) (if (= k 1) 1 (* k (f f (- k 1)))))))) is a familiar function.>]
>>> import datetime
>>> s = Story(body="I am the greatest!", upvotes=132, orig_poster="aph",
              post_date=datetime.datetime.now())
>>> s.id
>>> s.save()
>>> s.id
3
>>> s.body
'I am the greatest!'
>>> s.body="I am the bestest!"
>>> s.save()
>>> s
<Story: I am the bestest!>
>>> Story.objects.all()
[<Story: Brr!  Its cold here.>, <Story: It is intuitively obvious that (lambda (n) ((lambda (f) (f f
n)) (lambda (f k) (if (= k 1) 1 (* k (f f (- k 1)))))))) is a familiar function.>, <Story: I am the
bestest!>]
```

s is a instance of the Story class
s.save() stores it in the database

SQL

- Structured Query Language
  - Developed by IBM (San Jose lab) in 1970s
- Standard language for interacting with databases
  - Database functions from PS5 were "inspired" by SQL

```
(table-select bids 'item-name (lambda (pitem) (string=? pitem "CLAS")))
```

↓

```
SELECT * FROM bids WHERE itemname = "CLAS"
```

Creating Tables

```
(define bids
  (make-new-table (list 'bidder-name 'item-name 'amount))
```

↓

```
CREATE TABLE bids (
  "id" integer NOT NULL PRIMARY KEY,
  "bidder_name" NOT NULL varchar (50)
  "item_name" NOT NULL varchar (50)
  "amount" integer
);
```

Another language to learn?

No! Django (probably) automatically generates all the SQL you should need from Python code.

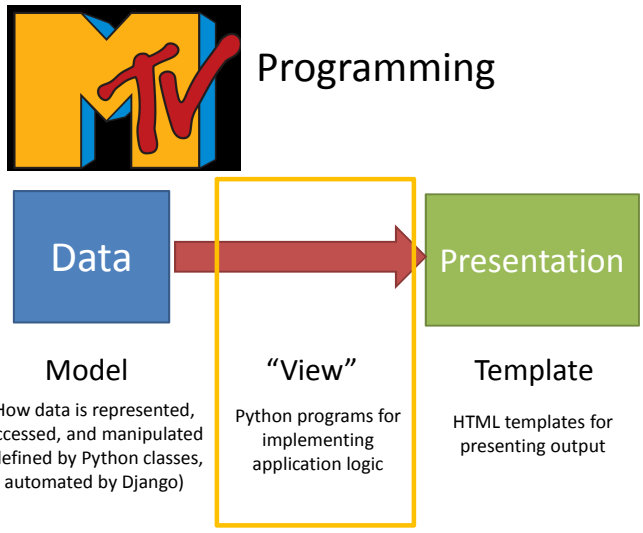
Django and Databases

overheardit/stories/models.py:

```
class Story(models.Model):
    body = models.CharField(max_length=500)
    upvotes = models.IntegerField(default=0)
    orig_poster = models.CharField(max_length=15)
    post_date = models.DateTimeField('date published', auto_now=True)
```

```
> python manage.py sqlall stories
BEGIN;
CREATE TABLE "stories_story" (
    "id" integer NOT NULL PRIMARY KEY,
    "body" varchar(500) NOT NULL,
    "upvotes" integer NOT NULL,
    "orig_poster" varchar(15) NOT NULL,
    "post_date" datetime NOT NULL
)
;
COMMIT;
```

MTV Programming



Model

How data is represented, accessed, and manipulated (defined by Python classes, automated by Django)

"View"

Python programs for implementing application logic

Template

HTML templates for presenting output

View

```
class Story(models.Model):
    body = models.CharField(max_length=500)
    upvotes = models.IntegerField(default=0)
    orig_poster = models.CharField(max_length=15)
    post_date = models.DateTimeField('date published', auto_now=True)
```

overheardit/stories/views.py:

```
def index(request):
    latest_story_list = Story.objects.all().order_by('-upvotes')[:20]
    return render_to_response('stories/index.html',
                              {'latest_story_list' : latest_story_list,
                               'user' : request.user})})
```

```
SELECT * FROM stories_story
  ORDER BY upvotes DESC
  LIMIT 20
```

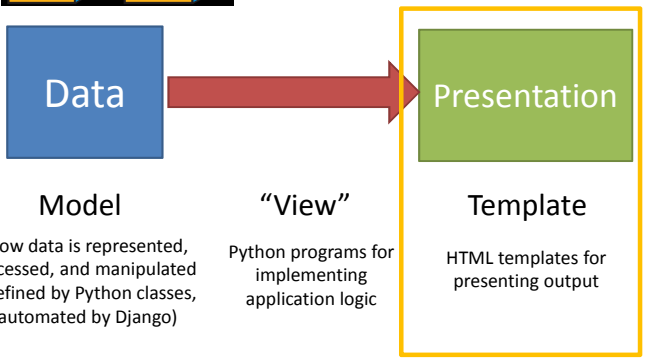
→ convert result (table) to a **QuerySet** object

More Complex Sorting

```
def index(request):
    latest_story_list = list(Story.objects.all()[:20])
    latest_story_list.sort(lambda s1, s2: cmp(s1.upvotes – s1.downvotes,
                                              s2.upvotes – s2.downvotes))
    return render_to_response('stories/index.html',
                              {'latest_story_list' : latest_story_list,
                               'user' : request.user})
```

convert the QuerySet into a Python list of Story objects

Programming



Model	"View"	Template
How data is represented, accessed, and manipulated (defined by Python classes, automated by Django)	Python programs for implementing application logic	HTML templates for presenting output

Connecting Data and Presentation

```
def index(request):
    latest_story_list = Story.objects.all().order_by('-upvotes')[:20]
    return render_to_response('stories/index.html',
                              {'latest_story_list' : latest_story_list,
                               'user' : request.user})})
```

render_to_response is a Django procedure that takes two inputs:
- pathname to an HTML template
- dictionary that defines variables for use in the template

It processes the HTML template to produce the HTML output file, which will be sent back as the response.

Template: templates/stories/index.html

```
{% include "header.html" %}
<body id="backdrop">
<a href="/posts/newpost/">Post New Quotation</a>
{% if latest_story_list %}
  {% for story in latest_story_list %}
    <div id="space"></div>
    <div id="story">
      Posted by <em>{{story.orig_poster}}</em>
          on {{story.post_date|date:"F j, Y"}}
      [up votes: {{story.upvotes}}]
      <blockquote>{{story.body}}</blockquote>
      <div id="voting">
        <a href="/posts/{{story.id}}/upvote/" method="post">Vote Up</a> |
        <a href="/posts/{{story.id}}/addcomment/" method="post">Add Comment</a> |
        {% load comments %}
        {% get_comment_count for story as comment_count %}
        <a href="/posts/{{story.id}}/" method="post">View Comments ({{comment_count}})</a>
      </div>
    </div>
  {% endfor %}
{% else %}
    <center><p>No stories are available.</p></center>
{% endif %}
{% include "footer.html" %}
```

Blue: in {% ... %}
  Commands interpreted by Django's template processor
Green: in {{ ... }}
  Variables from dictionary passed as second parameter
Red: HTML
  <div ...> ...

Okay, so how do I change the colors?

templates/stories/index.html:

```
{% include "header.html" %}
...
```

templates/stories/header.html:

```
{% include "style.html" %}
<div id="space"></div>
<div id="header">
Overheardit at UVa
</div>
...
```

templates/stories/style.html:

```
#backdrop {
  padding:0;
  margin-left: 100px;
  margin-right: 100px;
  margin-top: 10px;
  background-color:#ffffff;
  font: calibri, helvetica, arial, sans-serif;
}

#header {
  height:60px;
  background-color: #FFFF00;
  margin:auto;
  font: 300% calib
  text-align: cente
  color:#ffffff;
}
```

Defines the style of the ...

24-bit RGB colors like in PS1!
#RRGGBB
Hexadecimal: 0-F
FF = 255

Wednesday's Class

Steve Huffman (UVa CS 2005)
Co-founder (with Alexis Ohanian) of reddit

Before Wednesday's class: visit and try reddit.com

CHARGE

- Changing background images, colors and fonts is fun, but not the most important thing for your Project!
- Focus on doing computationally interesting things with a clean, simple interface

Exam 2 Review Session: tonight, 6-7:30 in Olsson 001

Wednesday, 5-6:30pm, Olsson 228E: Fireside chat with Wes Weimer and Kim Hazelwood