Accelerating Information Technology

http://aiti.mit.edu

Ghana Summer 2012
Lecture DJ07– Users and Auth/Auth

MISTI

Auth/Auth

- Authentication := are you who you say you are?
  - Check username and password
- Authorization := ok, you are who you say are, but what are you allowed to do?
  - Check assigned permissions to a user

Examples in this lecture are adapted from

- The Django docs, Authorization: https://docs.djangoproject.com/en/dev/topics/auth/
- The Django Book, Ch 14: http://djangobook.com/en/2.0/chapter14/

Django's auth/auth

- Users: People registered with your site
- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task
- Groups: A generic way of applying labels and permissions to more than one user
- Messages: A simple way to queue and display system messages to users

from http://djangobook.com/en/2.0/chapter14/

Persistent state

- How do you store information while browsing from page to page?
  - Which user are you?
  - What's in your shopping cart?
- Cookies: dictionary-like information sent with HTTP requests.

HTTP request to Google

```
GET / HTTP/1.1
Host: google.com
...
```

HTTP response from Google

```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: PREF=ID=5b14f22bdaf1e81c:TM=1167000671:LM=1167000671;
        expires=Sun, 17-Jan-2038 19:14:07 GMT;
        path=/; domain=.google.com
Server: GWS/2.1
…
```

HTTP request to Google

```
GET / HTTP/1.1
Host: google.com
Cookie: PREF=ID=5b14f22bdaf1e81c:TM=1167000671:LM=1167000671
...
```

Sessions

- Cookies can be messy and tedious to manipulate
- Cookies used unwisely are insecure
- The Django session API abstract the use of cookies and is secure

Enabling session support

(default with `django-admin startproject …`)

```
MIDDLEWARE_CLASSES = (
   ...

   'django.contrib.sessions.middleware.SessionMi
   ddleware‘,
   ...
)

INSTALLED_APPS = (
   ...
   'django.contrib.sessions',
   ...
)
```

Using sessions

```
def view_function(request):
    # Set a session value:
    request.session["fav_color"] = "blue"

    # Get a session value -- this could be called in a
    different view,
    # or many requests later (or both):
    fav_color = request.session["fav_color"]

    # Clear an item from the session:
    del request.session["fav_color"]

    # Check if the session has a given key:
    if "fav_color" in request.session:
        ...
```

class model.User
in django.contrib.auth.model

Fields

- username
- first_name
- last_name
- email
- password
- is_staff
- is_active
- is_superuser
- last_login
- date_joined

Methods

- is_authenticated()
- is_anonymous()
- get_full_name()
- set_password(passwd)
- check_password(passwd)
- get_group_permissions()
- get_all_permissions()
- has_perm(perm)
- has_perms(perm_list)
- has_module_perms(app_label)
- get_and_delete_messages()
- email_user(subj, msg)

Enabling auth/auth

(default with django-admin startproject …)

```
MIDDLEWARE_CLASSES = (
   ...

   'django.contrib.sessions.middleware.SessionMiddlew
   are‘,
   ...
)

INSTALLED_APPS = (
   ...
    'django.contrib.sessions',
   'django.contrib.auth’,
   'django.contrib.contrib.contenttypes ',
   ...
)
```

authenticate(username, password)

```
from django.contrib.auth import authenticate

user = authenticate(username='john', password='secret')
if user is not None:
    if user.is_active:
        print “Correct username and password!"
    else:
        print "Your account has been disabled!"
else:
    print “Incorrect username and password."
```

login(request, user)

```
from django.contrib.auth import authenticate, login

def my_view(request):
    uname = request.POST['username']
    pass = request.POST['password']
    user = authenticate(username=uname, password=pass)
    if user is not None:
        if user.is_active:
            login(request, user)
            # Redirect to a success page.
        else:
            # Return a 'disabled account' error message
    else:
        # Return an 'invalid login' error message.
```

logout(request)

```
from django.contrib.auth import logout

def logout_view(request):
    logout(request)
    # Redirect to a success page.
```

Enforcing logging in

```
from django.contrib.auth.decorators import
  login_required

@login_required
def my_view(request):
    ...
```

- If the user isn't logged in, redirect to settings.LOGIN_URL, passing the current absolute path in the query string. Example: /accounts/login/?next=/polls/3/.
- If the user is logged in, execute the view normally. The view code is free to assume the user is logged in.

Restricting access to users

```
# Simple way
def vote(request):
    if request.user.is_authenticated() and
       request.user.has_perm('polls.can_vote')):
        # vote here
    else:
        return HttpResponse("You can't vote in this poll.")

# Using the @user_passes_test decorator
def user_can_vote(user):
    return user.is_authenticated() and user.has_perm("polls.can_vote")

@user_passes_test(user_can_vote, login_url="/login/")
def vote(request):
    # Code here can assume a logged-in user with the correct
    # permission.
    ...
```

Referencing the user in a template

```
{% if user.is_authenticated %}
    <p>Welcome, {{ user.username }}. Thanks for
  logging in.</p>
{% else %}
    <p>Welcome, new user. Please log in.</p>
{% endif %}
```

And more…

- Permissions
- Groups

Read the Django docs!