Outline

• Session Framework

• User Authentication & Authorization in Django
Session Framework

- **Session Framework** lets you store and retrieve arbitrary data on a per-site-visitor basis.

- Stores data on server-side and abstracts sending and receiving cookies.

- Cookies contain a **Session ID**, not the data itself (There is one exception).
Session Framework

• Enabled by default in projects created by `django-admin.py`

```python
settings.py

MIDDLEWARE_CLASSES = (...,
    'django.contrib.sessions.middleware.SessionMiddleware',
    ...
)

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

We'll talk about middlewares later.
Session Life-Cycle

- Is there a cookie?
- Is the session expired?
Session Engines

- Database-backed sessions
- Cached sessions
- File-based sessions
- Cookie-based sessions

```
settings.py
SESSION_ENGINE = 'django.contrib.sessions.backends.file'
```
“But... what is a cache?”

Students panicked and asked the teacher.
Using Sessions in Views

- When **SessionMiddleware** is activated, each HttpRequest object will have a **session** attribute.

- It’s a dictionary-like object.

```python
def my_hilarious_view(request):
    ...
    fav_color = request.session['fav_color']
    ...

def my_awesome_view(request):
    ...
    request.session['fav_color'] = 'blue'
    ...
```
Using Sessions in Views

- Showtime!
- Let’s do something with our blog example 😜
User Authentication & Authorization

Thank you, Django.
User Authentication & Authorization

- **Authentication**
  Verifies that a user is who they claim to be.

- **Authorization**
  Determines what an authenticated user is allowed to do.

- We used the term “authentication” to refer both of these tasks.
The Authentication System

- The authentication system consists of:
  - Users
  - Permissions
  - Groups
  - Password hashing system
  - Forms and view tools for logging in users, etc.
  - Pluggable backend system
Installation

```python
settings.py

MIDDLEWARE_CLASSES = (...,
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.contrib.sessions.middleware.AuthenticationMiddleware',
    ...
)

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

```bash
python3 manage.py syncdb
```
The Authentication System

• There is Django’s default implementation.

• More than sufficient for most common project needs.

• Django supports extensive extension and customization of authentication system.
User Objects

```python
>>> from django.contrib.auth.models import User
>>> user = User.objects.create_user('john', 'lennon@thebeatles.com', 'johnpassword')

>>> from django.contrib.auth.models import User
>>> u = User.objects.get(username__exact='john')
>>> u.set_password('new password')
>>> u.save()

from django.contrib.auth import authenticate
user = authenticate(username='john', password='secret')
if user is not None:
    # the password verified for the user
    if user.is_active:
        print("User is valid, active and authenticated")
    else:
        print("The password is valid, but the account has been disabled!")
else:
    # the authentication system was unable to verify the username and password
    print("The username and password were incorrect.")
```
Login/Logout

- Showtime!
- Let’s add a login page to our blog!

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

def my_login_view(request):
    username = request.POST['username']
    password = request.POST['password']
    user = authenticate(username=username, password=password)
    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.
```
Limiting Access to Logged-In Users

• There is a simple, raw way:

```python
def my_view(request):
    if not request.user.is_authenticated():
        return redirect('/login/?next=%s' % request.path)
```

• There is a convenient way:

```python
from django.contrib.auth.decorators import login_required
@login_required
def my_view(request):
    ...
```

• What happens when user is not logged in?
Limiting Access to Logged-In Users

- Showtime!
- Let’s restrict users!
Accessing User in Templates

- `{{ user }}` would do it.
- Showtime.
Built-in Views

- Django provides several views that you can use for handling login, logout, and password management.

  ```python
  (r'^accounts/login/$', 'django.contrib.auth.views.login',
  {'template_name': 'myapp/login.html'}),
  ```

- Views for **logout**, **password_change**, etc.
Built-in Forms

- A bunch of default forms located in `django.contrib.auth.forms`.

- `AuthenticationForm`, `PasswordChangeForm`, `PasswordResetForm`, `UserCreationForm`, `SetPasswordForm`...

- Showtime.
Users in Admin

- Let’s see there.
Any Questions?
References

- https://docs.djangoproject.com/en/1.6/topics/http/sessions/
- https://docs.djangoproject.com/en/1.6/topics/auth/
- https://docs.djangoproject.com/en/1.6/topics/auth/default/
- https://docs.djangoproject.com/en/1.6/ref/contrib/auth/
- http://www.python.org