KAZAKH BRITISH TECHNICAL UNIVERSITY

BACKEND FRAMEWORK (DJANGO)

Lesson 3

Users/Anonymous

Django’s Session Framework

Enabling Sessions

- `INSTALLED_APPS`
    - `django.contrib.sessions`
- `MIDDLEWARE_CLASSES`
    - `django.contrib.sessions.middleware.SessionMiddleware`

Using Sessions in Views

```
# 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:
    …
```

Authentication / Authorization

Authentication

- Who is the user?
- Is the user really who he/she represents himself to be?

Authorization

- Is user X authorized to access resource R?
- Is user X authorized to perform operation P?
- Is user X authorized to perform operation P on resource R?

Enabling Authentication Support

- INSTALLED_APPS
  - django.contrib.auth

- MIDDLEWARE_CLASSES
  - django.contrib.auth.middleware.AuthenticationMiddleware

`request.user`

Enabling Authentication Support

```
request.user.is_authenticated
```

User

fields

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

User

Method

- `is_authenticated()`
- `is_anonymous()`
- `get_full_name()`
- `set_password(password)`
- `check_password(password)`
- `get_all_permissions()`
- `has_perm(perm)`

Logging In and Out

```
>>> from django.contrib import auth
>>> user = auth.authenticate(username='john',
password='secret')
>>> if user is not None:
        print "Correct!"
    else:
        print "Oops, that's wrong!"
```

Logging In and Out

```
>>> from django.contrib import auth
>>> user = auth.authenticate(username='john',
password='secret')
>>> if user is not None:
        print "Correct!"
    else:
        print "Oops, that's wrong!"
```

`login()`

Logging In and Out

```
from django.contrib import auth

def login(request):
   username = request.POST[‘username']
   password = request.POST['password']
   user = auth.authenticate(username=username, password=password)
   if user is not None and user.is_active:
      auth.login(request, user)
      return HttpResponseRedirect("/account/loggedin/")
   else:
      return HttpResponseRedirect("/account/invalid/")
```

Logging In and Out

```
from django.contrib import auth

def logout(request):
   user = auth.logout(request)
   return HttpResponseRedirect("/account/loggedout/")
```

Limiting Access to Logged-in Users

```
from django.contrib import auth

def my_view(request):
   if not request.user.is_authenticted:
      return HttpResponseRedirect(‘/login/‘)
   …
```

User Model

```
>>> from django.contrib.auth.models import User
>>> user = User.objects.create_user(username='user1',
... email=‘user1@gmail.com',
... password=‘asdasdasd’)
>>> user.is_staff = True
>>> user.save()
```

Changing Passwords

```
>>> user = User.objects.get(username='user1')
>>> user.set_password(‘qweqweqwe')
>>> user.save()
```

Changing Passwords

```
>>> user = User.objects.get(username='user1')
>>> user.set_password(‘qweqweqwe')
>>> user.save()
```

Don’t set the password attribute directly

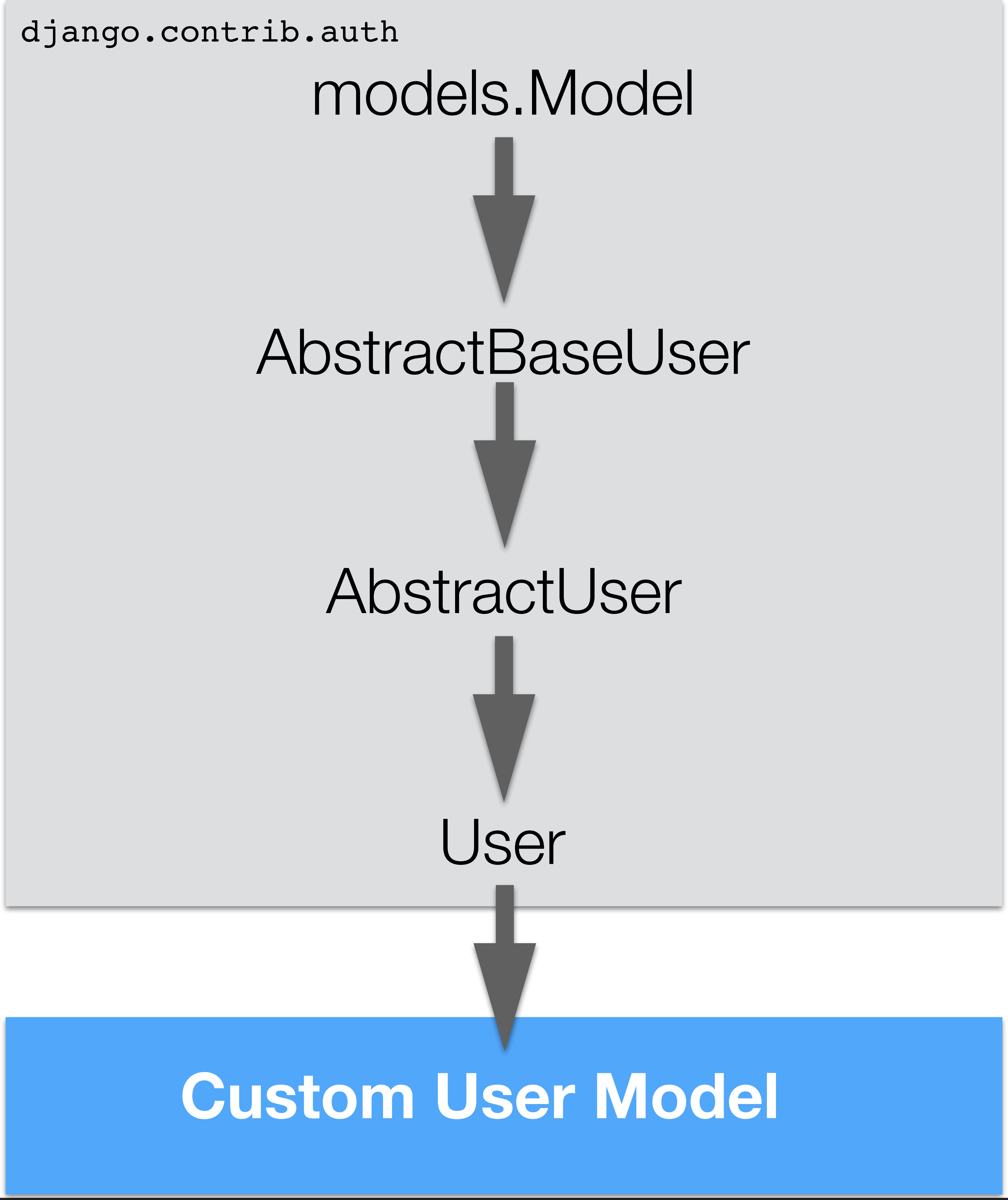
Django User

`django.contrib.auth`

Django User

1. Extending model with **proxy model**
2. Extend with a one-to-one field to **User (Profile)**
3. Substitute by subclassing from **AbstractUser**
4. Substitute by subclassing from **AbstractBaseUser**

New app for User logic


django.contrib.auth
models.Model
AbstractBaseUser
AbstractUser
User
Custom User Model

User extending

1. Extending from **AbstractUser**
2. Extending from **AbstractBaseUser and PermissionsMixin**
3. configure **`settings.py`** AUTH_USER_MODEL
4. Extend **UserManager** if necessary
5. Register in admin extending from

   `django.contrib.auth.admin.UserAdmin`

Questions?