

Validating Forms



Built-in Validators



Form Level Validation



Custom Validator



Field Level Validation

Validating Forms - Form Level Validation

► In forms.py

```
from django import forms
from django.core.exceptions import ValidationError

class ContactForm(forms.Form):

    subject = forms.CharField(max_length=100, required=True)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)

    def myclean(self):
        cleaned_data = super(ContactForm, self).clean()

        email = cleaned_data.get('sender')
        dom = 'gmail.com'

        if dom not in email:
            print('here')
            raise ValidationError('incorrect domain')
```

Validating Forms - Form Level Validation

► In views.py

```
from django.shortcuts import render
from django.http import HttpResponseRedirect

from .forms import ContactForm

# Create your views here.
def page(request):

    if request.method == 'POST':

        form = ContactForm(request.POST)
        if form.is_valid():
            form.myclean()
            return HttpResponseRedirect('thanks')

    else:

        form = ContactForm()
        context = {'form': form}

        return render(request, 'home.html', context)
```

Validating Forms - Form Level Validation

► In home.html

```
<form method='POST'>
{% csrf_token %}
{{ form }}

<input type="submit" value="submit">
</form>
```

Validating Forms - Field Level Validation

- In forms.py

forms.py

```
from django import forms
from django.core.exceptions import ValidationError

class ContactForm(forms.Form):

    subject = forms.CharField(max_length=100, required=True)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)

    def field_validation(self):
        email = self.cleaned_data.get('sender')
        dom = 'gmail.com'
        if dom not in email:
            raise ValidationError('field validation failed')
```

Validating Forms - Field Level Validation

► In forms.py

views.py

```
from django.shortcuts import render
from django.http import HttpResponse

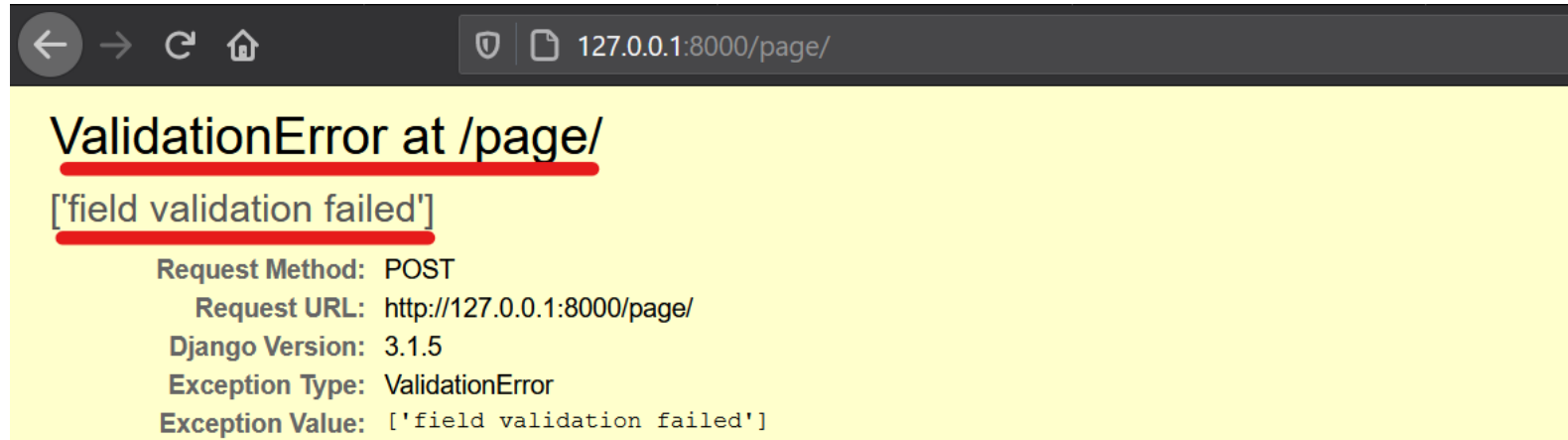
from .forms import ContactForm

def page(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            form.field_validation()
            return HttpResponse('thanks')

    else:
        form = ContactForm()
        context = {'form': form}
        return render(request, 'home.html', context)
```

Validating Forms - Field Level Validation

- ▶ If sender has any mail domain other than gmail.com



A screenshot of a web browser window showing a Django error page. The browser's address bar displays the URL `127.0.0.1:8000/page/`. The error message, displayed on a yellow background, is `ValidationError at /page/` with the value `['field validation failed']`. Below the error message, the following details are listed: Request Method: POST, Request URL: `http://127.0.0.1:8000/page/`, Django Version: 3.1.5, Exception Type: `ValidationError`, and Exception Value: `['field validation failed']`.

```
ValidationError at /page/
['field validation failed']
Request Method: POST
Request URL: http://127.0.0.1:8000/page/
Django Version: 3.1.5
Exception Type: ValidationError
Exception Value: ['field validation failed']
```

Forms - Built-in fields

Built-in Fields

- ▶ BooleanField
- ▶ CharField
- ▶ ChoiceField
- ▶ TypeChoiceField
- ▶ DateField
- ▶ DateTimeField
- ▶ DecimalField
- ▶ DurationField
- ▶ EmailField
- ▶ FileField
- ▶ FilePathField
- ▶ FloatField
- ▶ ImageField
- ▶ IntegerField
- ▶ JSONField
- ▶ URLField

More:

<https://docs.djangoproject.com/en/3.1/ref/forms/fields/#built-in-field-classes>

Forms - Built-in fields

forms.py

```
from django import forms

class myForm(forms.Form):
    x1 = forms.BooleanField(required=False)
    x2 = forms.CharField(min_length = 2, max_length = 10,
required=False)

    x = (('s', 'small'),
        ('m', 'medium'))
    x3 = forms.ChoiceField(choices=x)

    x = (('1', '10'),
        ('1', '20'))
    x4 = forms.TypedChoiceField(choices=x,coerce=int)
```

Forms - Built-in fields

forms.py

Contd..

```
x5 = forms.DateField(required=False)
x6 = forms.DateTimeField(required=False)
x7 = forms.DecimalField(required=False)
x8 = forms.DurationField(required=False)
x9 = forms.EmailField(required=False)
x10 = forms.FileField(required=False)
x11 =
forms.FilePathField(allow_folders=True, path=r'C:\Users\harulp663
\Desktop\practice\myproj', required=False)
x12 = forms.FloatField(required=False)
x14 = forms.IntegerField(required=False)
x15 = forms.JSONField(required=False)
x16 = forms.URLField(required=False)
```

Forms - Built-in fields

Home.html

```
<form method='POST' enctype="multipart/form-data">  
{% csrf_token %}  
{{ form }}  
  
<input type="submit" value="submit">  
</form>
```

Forms - Built-in fields

Views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from .forms import myForm

def page(request):
    if request.method == 'POST':
        form = myForm(request.POST,request.FILES)
        if form.is_valid():
            values = form.clean()
            print(values)
            return HttpResponse('check ur command prompt')
        else:
            return HttpResponse('not valid form')

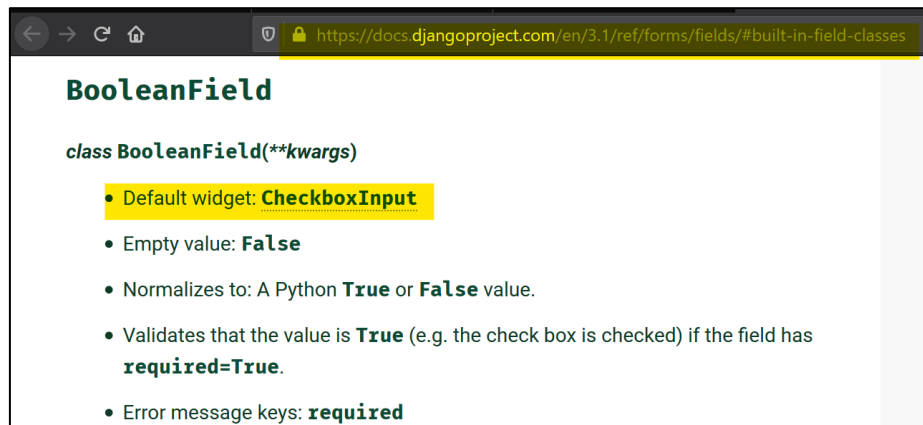
    else:
        form = myForm()
        context = {'form': form.as_ul}
        return render(request, 'home.html', context)
```

Forms - Built-in widgets

Built-in widgets

Highlighted some widgets in the diagram

- ▶ Whenever a built-in field is used, there is a default widget used



- X1: ☐
- X2:
- X3:
- X4:
- X5:
- X6:
- X7:
- X8:
- X9:
- X10: No file selected.
- X11:
- X12:
- X14:

Forms - Built-in widgets

- ▶ **Widgets handling input of text**

- ▶ TextInput
- ▶ NumberInput
- ▶ EmailInput
- ▶ URLInput
- ▶ PasswordInput
- ▶ HiddenInput
- ▶ DateInput
- ▶ DateTimeInput
- ▶ TimeInput
- ▶ TextArea

- ▶ **Selector and checkbox widgets**

- ▶ CheckBoxInput
- ▶ Select
- ▶ NullBooleanSelect
- ▶ SelectMultiple
- ▶ RadioSelect
- ▶ CheckboxSelectMultiple
- ▶ **File upload widgets**
 - ▶ File Input
 - ▶ ClearableFileInput

More:

<https://docs.djangoproject.com/en/3.1/ref/forms/widgets/#built-in-widgets>

Forms - Built-in widgets

forms.py

```
from django import forms

class myForm(forms.Form):
    x5 = forms.DateField( required=False,
                          widget=forms.SelectDateWidget)
```

127.0.0.1:8000/page/

- X1: ☐
- X2:
- X3:
- X4:
- X5:
- X6:
- X7:
- X8:
- X9:
- X10:
- X11:
- X12:
- X13:
- X14:

null

Model form

- ▶ For database-driven app
- ▶ Using models to create a form

models.py

```
from django.db import models

# Create your models here.
class member(models.Model):
    first_name= models.CharField(max_length=30)
    last_name= models.CharField(max_length=30)
```


Model form

forms.py

```
from django.forms import ModelForm
from .models import member

class myModelForm(ModelForm):
    class Meta:
        model = member
        fields = ["first_name", "last_name"]
```

Model form

views.py

```
from django.shortcuts import render
from django.http import HttpResponseRedirect

from .forms import myModelForm
from .models import member

def page(request):
    if request.method == 'POST':
        form = myModelForm(request.POST)
        if form.is_valid():
            form.save()
            values = form.clean()
            print(values)
            return HttpResponseRedirect('check ur command prompt')
        else:
            return HttpResponseRedirect('not valid form')

    else:
        form = myModelForm()
        context = {'form': form}
        return render(request, 'home.html', context)
```

Model form

- ▶ You can save the form data to the DB

views.py

```
form = myModelForm(request.POST)
if form.is_valid():
    form.save()
```

- ▶ Suppose update an existing record in db?

views.py

```
x = member.objects.get(first_name="BEULAH S")
form = myModelForm(request.POST, instance=x)
if form.is_valid():
    form.save()
```

Form sets

- ▶ Suppose you need the same form to be displayed n times - use formset

views.py

```
from django.forms import formset_factory

myFormSet = formset_factory(myModelForm, extra=2)
context = {'form': myFormSet}
return render(request, 'home.html', context)
```

- ▶ Output

First name: Last name: First name: Last name:

File upload

- ▶ In forms you need `FileField`
- ▶ You need to bind the file
 - ▶ With post method add `request.files`
 - ▶ In template html file add `enctype` in form tag
- ▶ Write a function which reads that file and write to someother location

File upload

forms.py

```
from django import forms

class myForm(forms.Form):
    x = forms.FileField()
```

Index.html

```
<form method='POST' enctype="multipart/form-data">
{% csrf_token %}
{{ form }}

<input type="submit" value="submit">
</form>
```

File upload

forms.py

```
from django.shortcuts import render
from .forms import myForm

def handle_file(f):
    with open('myfile.pdf', 'wb+') as fp:
        for chunk in f.chunks():
            fp.write(chunk)

def page(request):
    if request.method == 'POST':
        form = myForm(request.POST, request.FILES)
        handle_file(request.FILES['x'])
    else:
        form = myForm()
        return render(request, 'home.html', {'form': form})
```

Output:

- ▶ A new file would have been created at your project directory

Models Layer

-

Accessing related objects

Relation

- ▶ One-to-many
- ▶ Many-to-many

How ?

- ▶ ForeignKey for one-to-many
- ▶ ManyToManyField for many-to-many

Models Layer

-

Accessing related objects

one-to-many

Models.py

```
from django.db import models

# Create your models here.
class Mother(models.Model):
    name = models.CharField(max_length=10, null=True)
    num_children = models.IntegerField(null=True)

    def __str__(self):
        return self.name

class Child(models.Model):
    mother = models.ForeignKey(Mother, on_delete=models.CASCADE,
                               null=True)

    name = models.CharField(max_length=10, null=True)
    age = models.IntegerField(null=True)
    def __str__(self):
        return self.name
```

Models Layer

-

Accessing related objects

one-to-many

Python manage.py shell

```
>>> from myapp.models import Mother
>>> from myapp.models import Child
>>>
>>>
>>> m = Mother(name="Angelina", num_children=6)
>>> m.save()
>>>
>>> c1 = Child(name="Maddox", age=10, mother=m)
>>> c1.save()
>>> c2 = Child(name="Zahara", age=12, mother=m)
>>> c2.save()
```

```
sqlite> select * from myapp_mother;
3|Angelina|6
sqlite>
sqlite>
sqlite> select * from myapp_child;
4|2|10|Krishnan
5|2|12|SriDevi
6|3|10|Maddox
7|3|12|Zahara
```

Models Layer

-

Accessing
related
objects

one-to-many

- ▶ Accessing mother through child

Python manage.py shell

```
>>> x = Child.objects.get(id=6)
>>> x
<Child: Maddox>
>>> x.mother
<Mother: Angelina>
>>> x.mother.name
'Angelina'
>>> x.name
'Maddox'
```

Models Layer

-

Accessing
related
objects

one-to-many

- ▶ Accessing child through mother

Python manage.py shell

```
>>> x = Mother.objects.get(id=3)
>>> x.child_set.all()
<QuerySet [<Child: Maddox>, <Child: Zahara>]>
```

Models Layer

-

Accessing
related
objects

one-to-many

- ▶ creating a relation without save() explicitly called

Python manage.py shell

```
>>> c3 = x.child_set.create(name="Shiloh", age=13)
>>> x.child_set.all()
<QuerySet [<Child: Maddox>, <Child: Shiloh>]>
```

Models Layer

-

Accessing related objects

one-to-many

- ▶ Removing a relation

Python manage.py shell

```
>>> x = Mother.objects.get(id=3)
>>> x.child_set.all()
<QuerySet [<Child: Maddox>, <Child: Zahara>]>
>>> x.child_set.remove(c2)
>>> x.child_set.all()
<QuerySet [<Child: Maddox>]>
```

Models Layer

- ▶ adding a relation

-

Accessing related objects

one-to-many

Python manage.py shell

```
>>> c2 = Child.objects.get(id=7)
>>> c2
<Child: Zahara>
>>> x
<Mother: Angelina>
>>> x.child_set.all()
<QuerySet [<Child: Maddox>, <Child: Shiloh>]>
>>> x.child_set.add(c2)
>>> x.child_set.all()
<QuerySet [<Child: Maddox>, <Child: Zahara>, <Child: Shiloh>]>
```

Models Layer

-

Accessing related objects

many-to-many

Models.py

```
# many-to-many
```

```
class Subject(models.Model):  
    name = models.CharField(max_length=10)  
    def __str__(self):  
        return self.name  
  
class Student(models.Model):  
    name = models.CharField(max_length=10)  
    subject = models.ManyToManyField(Subject)  
    def __str__(self):  
        return self.name
```


Models Layer

-

Accessing
related
objects

many-to-many

Models.py

```
>>> from myapp.models import Subject, Student
>>>
>>> s1 = Subject(name="Tamil")
>>> s1.save()
>>>
>>> s2 = Subject(name="ECA")
>>> s2.save()
>>>
>>> x = Student(name="Jessy")
>>> x.save()
>>> x.subject.set(s)
>>>
>>> x = Student(name="Peter")
>>> x.save()
>>> x.subject.set(s)
```

Authentication

Django built-in authentication

- ▶ Authentication - Checking if the user is valid
- ▶ Authorization - what the authenticated user is allowed to do
- ▶ APPS
 - ▶ `django.contrib.auth`
 - ▶ `django.contrib.contenttypes`
- ▶ MIDDLEWARE
 - ▶ `SessionMiddleware`
 - ▶ `AuthenticationMiddleware`

Authentication

Django built-in authentication

USER

- ▶ username
- ▶ password
- ▶ email
- ▶ first_name
- ▶ last_name

Authentication

Django built-in authentication

using *authenticate()*

views.py

```
from django.contrib.auth import authenticate

# Create your views here.
def page(request):

    # NOTE: passwords shouldn't be passed as plain text !
    # usually, we will get this value through a form
    # that it would be passed as a variable value

    user = authenticate(username='admin', password='admin')
    return HttpResponse(user)

    # also username admin and password admin
    # is a weak account as this is a default value
    # in most of the systems and people can easily guess it !!
```

Authentication

Django built-in authentication

- ▶ using *request.user.is_authenticated* to check if the user is logged-in
- ▶ Returns True if the user is logged in
- ▶ Returns False if not

Authentication

Django built-in authentication

Login

views.py

```
from django.contrib.auth import authenticate
from django.contrib.auth import Login

# Create your views here.
def page(request):

    user = authenticate(username='admin', password='admin')
    if user:
        Login(request, user)
```

Authentication

Django built-in authentication

logout

views.py

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

# Create your views here.
def page(request):

    user = authenticate(username='admin', password='admin')
    if user:
        login(request, user)
        # ur logics etc, finally logout ?
        Logout(request)
```

Authentication

Password management

- ▶ In settings.py - AUTH_PASSWORD_VALIDATORS
 - ▶ UserAttributeSimilarityValidator
 - Checks whether there is similarity between username and password
 - ▶ MinimumLengthValidator
 - Minimum chars to be in a password
 - ▶ CommonPasswordValidator
 - Not a commonly used password, example: admin
 - ▶ NumericPasswordValidator
 - Checks whether the password contains only numbers

Authentication

Password management

Validate password

views.py

```
from django.shortcuts import render
from django.http import HttpResponseRedirect

from django.contrib.auth.password_validation import
validate_password

def page(request):
    mypassword = 'admin'
    return HttpResponseRedirect(validate_password(mypassword))
```

ValidationError at /page/

['This password is too short. It must contain at least 8 characters.', 'This password is too common.']

Authentication

customizing authentication

Try out -

<https://docs.djangoproject.com/en/3.1/topics/auth/customizing/#a-full-example>

Logging

- ▶ Using python inbuilt module logging

views.py

```
from django.shortcuts import render
from django.http import HttpResponse

import logging

# Create your views here.
def page(request):
    logging.basicConfig(level=logging.DEBUG)
    logging.info('page() is invoked')
    logging.error('some error occurred')

    return HttpResponse('check cmd')
```

Static file management

css, js, img

- ▶ Add `django.contrib.staticfiles` to `INSTALLED_APPS` in `settings.py`
- ▶ In `settings.py`

`settings.py`

```
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, 'static')  
]
```

`index.html`

```
{% load static %}  
...  
{% static '/css/main.css' %}
```

pagination

views.py

```
from django.shortcuts import render
from django.http import HttpResponse

from django.contrib.auth.models import User
from django.core.paginator import Paginator, EmptyPage

# Create your views here.
def foo(requests):
    users = User.objects.filter(groups__name='member')
    users = Paginator(users, 5)

    page_num = requests.GET.get('page', 1)
    try:
        users_in_page = users.page(page_num)
    except EmptyPage:
        users_in_page = users.page(1)

    context = {'users': users_in_page}

    return render(requests, 'indexx.html', context)
```

pagination

index.html

```
<table>
<thead>
  <tr>
    <th>Username</th>
    <th>Date Joined</th>
  </tr>
</thead>
<tbody>
  {% for user in users %}
    <tr>
      <td> {{ user.username }} </td>
      <td> {{ user.date_joined }} </td>
    </tr>
  {% endfor %}
</tbody>
</table>

{% if users.has_previous %}
  <a href="{% url 'trainer' %}?page={{ users.previous_page_number }}"
  class="ms-auto"> Prev </a>
{% endif %}

{% if users.has_next %}
  <a href="{% url 'trainer' %}?page={{ users.next_page_number }}"
  class="ms-auto"> Next </a>
{% endif %}
```

caching

- ▶ To cut the overhead of making queries to database

Views.py

```
from django.views.decorators.cache import cache_page

# cache page takes argument of num of seconds the cache needs to be saved

@cache_page(30)
def foo(requests):
    users = User.objects.filter(groups__name='member')
```

Site maps

- ▶ A way to tell search engines about your websites links and posts
- ▶ How?
 - Using a xml file
 - That says the urls with some meta data for the engines to understand the content
- ▶ How in Django?

In settings.py, add below to `INSTALLED_APPS`

settings.py

```
INSTALLED_APPS = [  
    'django.contrib.sites',  
    'django.contrib.sitemaps',  
]  
  
SITE_ID = 1
```


Site maps

models.py

```
from django.db import models
from django.urls import reverse

# Create your models here.
class blogs(models.Model):
    title = models.CharField(max_length=10)

    def get_absolute_url(self):
        return reverse("my_blogs", args=[str(self.id)])
```

urls.py

```
from django.contrib.sitemaps.views import sitemap
from common.sitemaps import mysitemap
from common.views import myview

sitemaps = {'blogs': mysitemap}

urlpatterns = [
    path('my_blogs/<int:id>', myview, name='my_blogs'),
    path('sitemap.xml', sitemap, {'sitemaps': sitemaps})
]
```

Site maps

views.py

```
from .models import blogs
from django.shortcuts import get_object_or_404

def myview(requests, post_id = id):
    item = get_object_or_404(blogs, id=post_id)

    return render(requests, 'sample.html', {'items': item})
```

Sample.html

```
{{ item }}
```

Admin.py

```
from django.contrib import admin
from .models import blogs

# Register your models here.
admin.site.register(blogs)
```

Site maps

Sitemaps.py

```
from django.contrib.sitemaps import Sitemap

from .models import blogs

class mysitemap(Sitemap):
    def items(self):
        return blogs.objects.all()
```



Message Framework

- ▶ For notification messages

In settings.py we have already have these

- ▶ `INSTALLED_APPS` -- `django.contrib.messages`
- ▶ `MIDDLEWARE` --
`django.contrib.sessions.middleware.SessionMiddleware,`
`django.contrib.messages.middleware.MessageMiddleware`
- ▶ `TEMPLATES` --
`django.contrib.messages.context_processors.messages`

You need view function, urls , and a html template

Message Framework

views.py

```
from django.contrib import messages

def my_message(requests):
    messages.add_message(requests, messages.INFO, 'hiiii')
    messages.success(requests, 'success message')
    messages.warning(requests, 'warning message')
    messages.error(requests, 'error occurred')
    context = {}
    return render(requests, 'my_message.html', context)
```

views.py

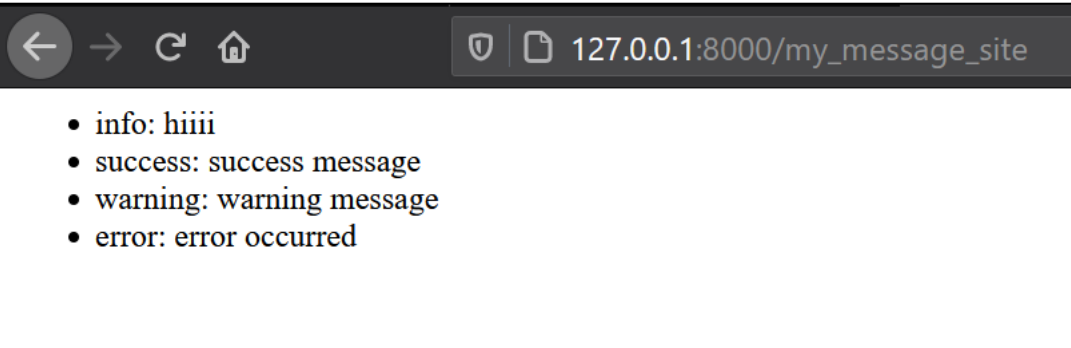
```
from trainer.views import my_message

urlpatterns = [
    path('my_message_site', my_message)
]
```

Message Framework

My_message.html

```
{% if messages %}
<ul>
  {% for message in messages %}
    <li>{{message.tags}}: {{ message }} </li>
  {% endfor %}
</ul>
{% endif %}
```

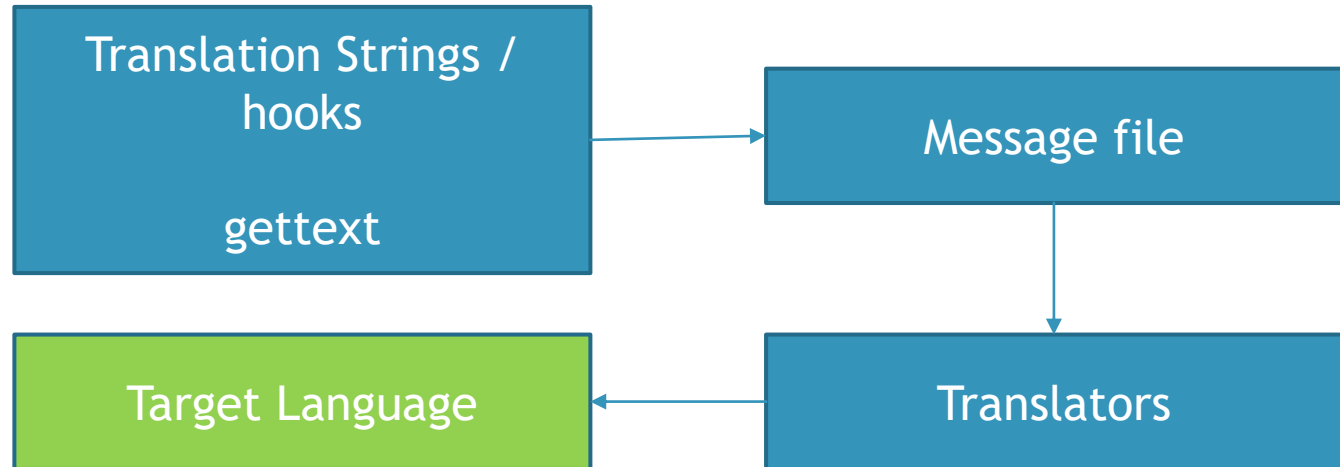


Internationalization and localization

- ▶ Making websites having option to switch languages
- ▶ Internalization
 - ▶ The code part to translate
 - ▶ Done by developers
- ▶ Localization
 - ▶ Translation
 - ▶ Done by Translators

Internationalization and localization

► Translation strings



Internationalization and localization

Translation Strings /
hooks
gettext

► Installing get text

<https://docs.djangoproject.com/en/3.1/topics/i18n/translation/#gettext-on-windows>

download from - <https://mlocati.github.io/articles/gettext-iconv-windows.html>

Internationalization and localization

Message file

How to generate message file

Views.py

```
from django.shortcuts import render
from django.utils.translation import gettext as _
from django.utils.translation import activate

# Create your views here.

def mytranslator(request):
    language_code = request.GET.get('Langcode', 'en-us')
    activate(language_code)

    context = {'message': _('hello')}
    return render(request, 'translation_ex.html', context)
```

Internationalization and localization

Message file

Translation_ex.html

```
{{ message }}  
  
<br/><br/><br/>  
  
<a href="{% url 'translation_ex' %}?langcode=en-us"> English </a>  
<a href="{% url 'translation_ex' %}?langcode=ta"> Tamil </a>
```

- ▶ In project directory create a directory **locale**
- ▶ In project directory
django-admin makemessages -l ta

This will create a .po file in locale dir

Translators

- ▶ Now translate the messages
- ▶ In locale dir, in that file translate
- ▶ Added வணக்கம் in line 22

Internationalization and localization

```
1 # SOME DESCRIPTIVE TITLE.
2 # Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
3 # This file is distributed under the same license as the PACKAGE package.
4 # FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
5 #
6 #, fuzzy
7 msgid ""
8 msgstr ""
9 "Project-Id-Version: PACKAGE VERSION\n"
10 "Report-Msgid-Bugs-To: \n"
11 "POT-Creation-Date: 2021-03-23 21:19+0530\n"
12 "PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
13 "Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
14 "Language-Team: LANGUAGE <LL@li.org>\n"
15 "Language: \n"
16 "MIME-Version: 1.0\n"
17 "Content-Type: text/plain; charset=UTF-8\n"
18 "Content-Transfer-Encoding: 8bit\n"
19 "Plural-Forms: nplurals=2; plural=(n != 1);\n"
20 #: .\translation_ex\views.py:8
21 msgid "hello"
22 msgstr "வணக்கம்"
23
```

Internationalization and localization

Translators

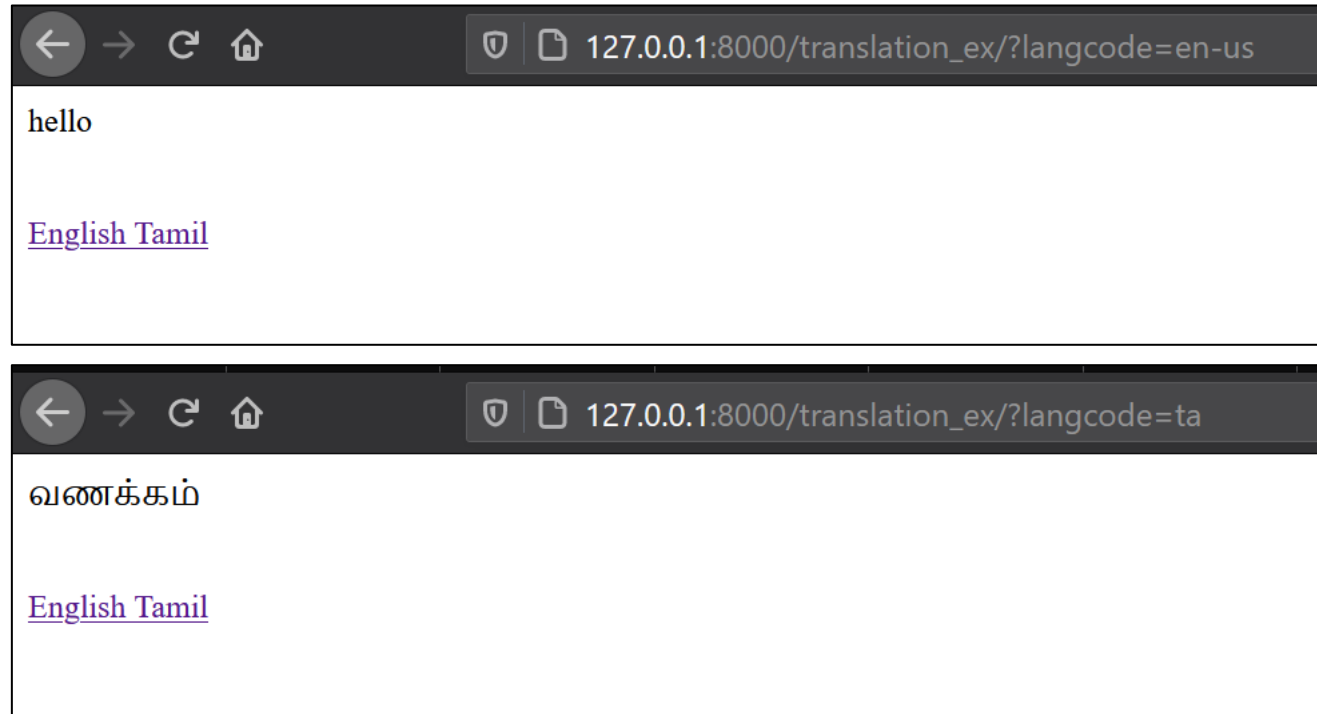
- ▶ In project directory,
django-admin compilemessages

This will create .mo file in locale dir

- ▶ Language codes can be found at
https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

Internationalization and localization

- ▶ Makemigrations, migrate, runserver



- ▶ In settings.py, by default we have UTC timezone

TIMEZONE='UTC'

- ▶ In admin page → users → we have date joined


- ▶ This is in UTC TZ

Timezone


Important dates

Last login:

Date:

Today | 


Time:

Now | 


Note: You are 5.5 hours ahead of server time.

Date joined:

Date:

Today | 

Time:

Now | 

Note: You are 5.5 hours ahead of server time.

Delete

Save and add another

Save and continue editing

SAVE

TimeZone

- ▶ In settings.py, Change timezone to what you expect


https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

- ▶ TIMEZONE='Asia/Kolkata'
- ▶ Now in same admin → user page we have date joined in India time


Important dates

Last login:

Date:


Today 

Time:


Now 

Date joined:

Date:

Today 

Time:

Now 

Delete

Save and add another

Save and continue editing

SAVE

- ▶ In settings.py, Add below configuration

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'  
  
EMAIL_HOST_USER = 'xxx@gmail.com'  
EMAIL_HOST = 'smtp.gmail.com'  
EMAIL_PORT = 587  
EMAIL_USE_TLS = True  
EMAIL_HOST_PASSWORD = "xxx"
```

Sending Emails

- ▶ Make changes in ur sending account so that it would allow apps to send email

Sending Emails

► In view function

views.py

```
from django.core.mail import send_mail

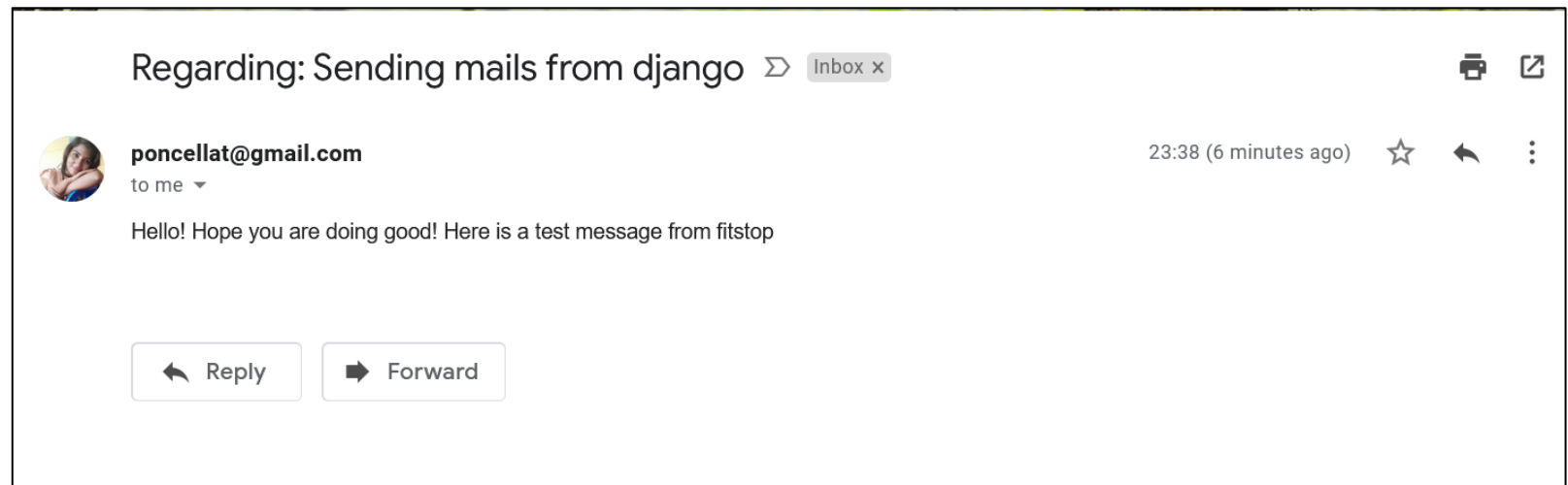
from django.http import HttpResponse

# Create your views here.
def mailsender(request):
    send_mail(
        'Regarding: Sending mails from django',
        'Hello! Hope you are doing good! Here is a test message from fitstop',
        'xxx@gmail.com',
        ['xxx@gmail.com'],
    )

    return HttpResponse('Check ur mail')
```

Sending Emails

- ▶ Runserver, visit the page which will call this view function
- ▶ Check the account to which mail was sent



Serialization

- ▶ Converting to a format like xml, json, yaml
- ▶ How ?
- ▶ After retrieving table data, use Django serializer

views.py

```
from django.shortcuts import render
from django.contrib.auth.models import User

from django.core import serializers

# Create your views here.
def myserializer(request):


    users = User.objects.filter(groups__name='member')
    data = serializers.serialize("yaml", users)

    context = {'data': data}

    return render(request, 'myserializer.html', context)
```

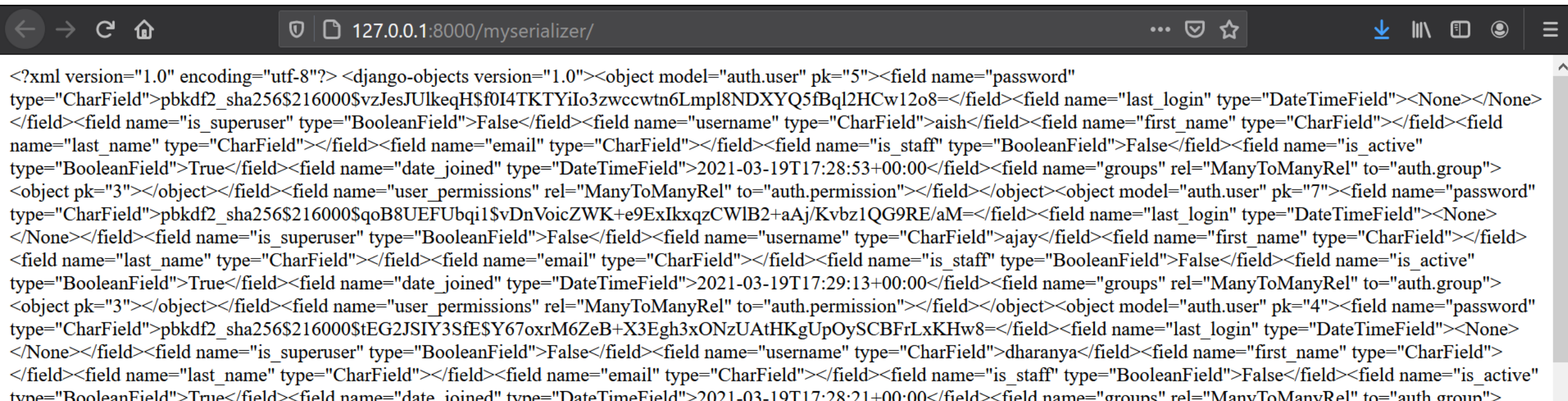
Serialization

► Without any serialization



A screenshot of a web browser displaying a REST API response. The address bar shows the URL `127.0.0.1:8000/myserializer/`. The response body is a raw Django QuerySet representation: `<QuerySet [<User: aish>, <User: ajay>, <User: dharanya>, <User: gayathri>, <User: hephzi>, <User: karthik>, <User: ramya>, <User: sam>, <User: sandhya>, <User: sankar>, <User: sharmila>, <User: shob>, <User: ussain>, <User: wonderwoman>]>`

► Xml serialization



A screenshot of a web browser displaying a REST API response with XML serialization. The address bar shows the URL `127.0.0.1:8000/myserializer/`. The response body is an XML document: `<?xml version="1.0" encoding="utf-8"?> <django-objects version="1.0"><object model="auth.user" pk="5"><field name="password" type="CharField">pbkdf2_sha256$216000$VzJesJUlkeqH$f0I4TKTYiIo3zwcwtn6Lmpl8NDXYQ5fBql2HCw12o8=</field><field name="last_login" type="DateTimeField"><None></None></field><field name="is_superuser" type="BooleanField">False</field><field name="username" type="CharField">aish</field><field name="first_name" type="CharField"></field><field name="last_name" type="CharField"></field><field name="email" type="CharField"></field><field name="is_staff" type="BooleanField">False</field><field name="is_active" type="BooleanField">True</field><field name="date_joined" type="DateTimeField">2021-03-19T17:28:53+00:00</field><field name="groups" rel="ManyToManyRel" to="auth.group"><object pk="3"></object></field><field name="user_permissions" rel="ManyToManyRel" to="auth.permission"></field></object><object model="auth.user" pk="7"><field name="password" type="CharField">pbkdf2_sha256$216000$qoB8UEFUbqi1$VdnVoicZWk+e9ExIkxqzCWlB2+aAj/Kvbz1QG9RE/aM=</field><field name="last_login" type="DateTimeField"><None></None></field><field name="is_superuser" type="BooleanField">False</field><field name="username" type="CharField">ajay</field><field name="first_name" type="CharField"></field><field name="last_name" type="CharField"></field><field name="email" type="CharField"></field><field name="is_staff" type="BooleanField">False</field><field name="is_active" type="BooleanField">True</field><field name="date_joined" type="DateTimeField">2021-03-19T17:29:13+00:00</field><field name="groups" rel="ManyToManyRel" to="auth.group"><object pk="3"></object></field><field name="user_permissions" rel="ManyToManyRel" to="auth.permission"></field></object><object model="auth.user" pk="4"><field name="password" type="CharField">pbkdf2_sha256$216000$tEG2JSIY3SfE$Y67oxrM6ZeB+X3Egh3xONzUAthKkgUpOySCBfrLxKHw8=</field><field name="last_login" type="DateTimeField"><None></None></field><field name="is_superuser" type="BooleanField">False</field><field name="username" type="CharField">dharanya</field><field name="first_name" type="CharField"></field><field name="last_name" type="CharField"></field><field name="email" type="CharField"></field><field name="is_staff" type="BooleanField">False</field><field name="is_active" type="BooleanField">True</field><field name="date_joined" type="DateTimeField">2021-03-19T17:28:21+00:00</field><field name="groups" rel="ManyToManyRel" to="auth.group">`

Serialization

► Json format

```
[{"model": "auth.user", "pk": 5, "fields": {"password": "pbkdf2_sha256$216000$svzJesJUlkeqH$f0I4TKTYiIo3zwccwtn6Lmpl8NDXYQ5fBql2HCw12o8=", "last_login": null, "is_superuser": false, "username": "aish", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:28:53Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 7, "fields": {"password": "pbkdf2_sha256$216000$qoB8UEFUbqi1$VdNVoicZWK+e9ExIkxqzCWIB2+aAj/Kvbz1QG9RE/aM=", "last_login": null, "is_superuser": false, "username": "ajay", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:29:13Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 4, "fields": {"password": "pbkdf2_sha256$216000$tEG2JSIY3SfE$Y67oxrM6ZeB+X3Egh3xONzUAthKGuUpOySCBFrLxKHw8=", "last_login": null, "is_superuser": false, "username": "dharanya", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:28:21Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 17, "fields": {"password": "pbkdf2_sha256$216000$aO03uenzqXC1$Sc0mNVB0Jw+81YPEf9Cd+QYLb70cLKwtHOLrCR14pQuA=", "last_login": null, "is_superuser": false, "username": "gayathri", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:32:21Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 9, "fields": {"password": "pbkdf2_sha256$216000$e9ahn578OeXz$5nmalXx7RGET+vrIkGjA6g59D8uKuInTxQvIQXlg14Q=", "last_login": null, "is_superuser": false, "username": "hephzi", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:29:50Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 15, "fields": {"password": "pbkdf2_sha256$216000$otX2L0p38aTd$sknmukCF4mAbw0uq0TQbovbLI+UjwglomUg20RqKcpKM=", "last_login": null, "is_superuser": false, "username": "karthik", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:31:30Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 16, "fields": {"password": "pbkdf2_sha256$216000$5X8w8JHLcHpr$M1LEw90IxIiqVh/ooGUPQGvjXKmnobAZyKpvhGGT4M=", "last_login": null, "is_superuser": false, "username": "ramya", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:32:02Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 6, "fields": {"password": "pbkdf2_sha256$216000$9JPbvqVjY0MNSQHHXNhjt9L2qMtyJ2fd9Jym1PF3p8ZkYVb+npRUhdFo=", "last_login": null, "is_superuser": false, "username": "sam", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:29:03Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 8, "fields": {"password": "pbkdf2_sha256$216000$rw5kNYQZNe3n$Da+h2V/FvjAKEeIAHHUMcpnWWMteVXEvVLJlAkn2ti4=", "last_login": null, "is_superuser": false, "username": "sandhya", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:29:34Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 12, "fields": {"password": "pbkdf2_sha256$216000$hNNGmoQGWKY3$ljchMrzDwLN290yKZMj8rm4SyMGVn6yTTIPXsTi5bLs=", "last_login": null, "is_superuser": false, "username": "sankar", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:30:36Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 14, "fields": {"password": "pbkdf2_sha256$216000$X6vAzcCx7dvJ$w+p9jomUNdnDUmT/vH+Mlmb5mhedpgfpCpXF9onA1k=", "last_login": null, "is_superuser": false, "username": "sharmila", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:31:07Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 13, "fields": {"password": "pbkdf2_sha256$216000$TczjPgSvnS0b$8kX109s0sgj5rssfC4bq+x+aQNQJ1jQcxj2tUAEMij0=", "last_login": null, "is_superuser": false, "username": "shob", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:30:47Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 11, "fields": {"password": "pbkdf2_sha256$216000$gerAazUuUpPu$OcT6kpOMwdB9rpDUszSZjtQMFLb3ew1BiOgRSZy+zk=", "last_login": null, "is_superuser": false, "username": "ussain", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:30:22Z", "groups": [3], "user_permissions": []}}, {"model": "auth.user", "pk": 10, "fields": {"password": "pbkdf2_sha256$216000$FXHQXutNtXiY$0lJu3M1ZOuimeaELZ5aCQb0SCDMzaFkzp2p8Erwtllg=", "last_login": null, "is_superuser": false, "username": "wonderwoman", "first_name": "", "last_name": "", "email": "", "is_staff": false, "is_active": true, "date_joined": "2021-03-19T17:30:11Z", "groups": [3], "user_permissions": []}}]
```


Serialization



```
- model: auth.user pk: 5 fields: password: pbkdf2_sha256$216000$VzJesJUlkeqH$F0I4TKTYiIo3zwccwtN6Lmpl8NDXYQ5fBql2HCw12o8= last_login: null is_superuser: false username: aish
first_name: " last_name: " email: " is_staff: false is_active: true date_joined: 2021-03-19 17:28:53+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 7 fields: password:
pbkdf2_sha256$216000$QoB8UEFUbqi1$VdNVoicZWK+e9ExIkxqzCWIB2+aAj/Kvbz1QG9RE/aM= last_login: null is_superuser: false username: ajay first_name: " last_name: " email: " is_staff:
false is_active: true date_joined: 2021-03-19 17:29:13+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 4 fields: password:
pbkdf2_sha256$216000$StEG2JSIY3SfE$Y67oxrM6ZeB+X3Egh3xONzUATHKgUpOySCBfrLxKHw8= last_login: null is_superuser: false username: dharanya first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:28:21+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 17 fields: password:
pbkdf2_sha256$216000$aO03uenzqXC1$C0mNVB0Jw+81YPeF9Cd+QYLB70cLKwtH0lrCR14pQuA= last_login: null is_superuser: false username: gayathri first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:32:21+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 9 fields: password:
pbkdf2_sha256$216000$e9ahn578OeXz$5nmALXx7RGET+vrIkGjA6g59D8uKuInTxQvIQXlg14Q= last_login: null is_superuser: false username: hephzi first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:29:50+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 15 fields: password:
pbkdf2_sha256$216000$otX2L0p38aTd$sknmukCF4mAbw0uq0TQbovbLI+UjwglomUg20RqKcpKM= last_login: null is_superuser: false username: karthik first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:31:30+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 16 fields: password:
pbkdf2_sha256$216000$5X8w8JHLcHpr$M1LEw90IxIiqVh/ooGUPQGvjXKmnobAZyKpvhGGT4M= last_login: null is_superuser: false username: ramya first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:32:02+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 6 fields: password:
pbkdf2_sha256$216000$9JPbvqVjY0MN$QHhXNht9L2qMtyJ2fd9Jym1PF3p8ZkYVb+npRUhdFo= last_login: null is_superuser: false username: sam first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:29:03+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 8 fields: password:
pbkdf2_sha256$216000$rw5kNYQZNe3n$Da+h2V/FvjAKEeIAHHUMcpnWWMteVXEVLJlAkn2ti4= last_login: null is_superuser: false username: sandhya first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:29:34+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 12 fields: password:
pbkdf2_sha256$216000$shNNGmoQGWKY3$ljchMrzDwLN290yKZMj8rm4SyMGVn6yTTIPXsTi5bLs= last_login: null is_superuser: false username: sankar first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:30:36+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 14 fields: password:
pbkdf2_sha256$216000$X6vAzcCx7dvJ$w+p9jomUNdnDUmT/vH+Mlmb5mhedpgfpuCpXF9onA1k= last_login: null is_superuser: false username: sharmila first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:31:07+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 13 fields: password:
pbkdf2_sha256$216000$TczjPgSvnS0b$8kX109s0sgj5rssfC4bq+x+aQNQJ1jQcxj2tUAEMij0= last_login: null is_superuser: false username: shob first_name: " last_name: " email: " is_staff: false
is_active: true date_joined: 2021-03-19 17:30:47+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 11 fields: password:
pbkdf2_sha256$216000$gerAazUuUpPu$OcT6kpOMwdB9rpDUszSZjtQMufLb3ew1BiOgRSZy+zk= last_login: null is_superuser: false username: ussain first_name: " last_name: " email: "
is_staff: false is_active: true date_joined: 2021-03-19 17:30:22+00:00 groups: - 3 user_permissions: [] - model: auth.user pk: 10 fields: password:
pbkdf2_sha256$216000$FXHQXutNtXiY$0lJu3M1ZOuimeaELZ5aCQb0SCDMzaFkzp2p8Erwtllg= last_login: null is_superuser: false username: wonderwoman first_name: " last_name: " email:
" is_staff: false is_active: true date_joined: 2021-03-19 17:30:11+00:00 groups: - 3 user_permissions: []
```

Signals

- ▶ Say you have two models, Student and Subject
- ▶ Every student has a set of subjects
- ▶ Say if a subject is added, It might need to set to one of the interested student
- ▶ How ?
- ▶ Using Signals
- ▶ If there are any action on one model, that needs to signalled to another model use signals

Signals

- ▶ Say you have two models, Student and Subject
- ▶ Every student has a set of subjects
- ▶ Say if a subject is added, It might need to set to one of the interested student
- ▶ How ?
- ▶ Using Signals
- ▶ If there are any action on one model, that needs to signalled to another model use signals

Signals

signals.py

```
from django.db.models.signals import post_save
from django.dispatch import receiver

from .models import Student
from .models import Subject

@receiver(post_save, sender=Subject)
def update_student(sender, instance, created, **kwargs):
    print('here')
    if not created:
        x = Student(id=4)

        tmp = sender.objects.filter(name=instance)[:1]
        y = sender(id=tmp[0].id)

        x.subject.add(y.id)
        print('Updated student', y.id, y.name)

#post_save.connect(update_student, sender=Subject)
```

Signals

models.py

```
from django.db import models

class Subject(models.Model):
    name = models.CharField(max_length=10)

    def __str__(self):
        return self.name

class Student(models.Model):
    name = models.CharField(max_length=10)
    subject = models.ManyToManyField(Subject, null=True)

    def __str__(self):
        return self.name
```

Signals

- ▶ In Settings.py use 'mysignals.apps.MysignalsConfig' in INSTALLED_APPS

apps.py

```
from django.apps import AppConfig

class MysignalsConfig(AppConfig):
    name = 'mysignals'

    def ready(self):
        import mysignals.signals
```

admin.py

```
from django.contrib import admin

from .models import Subject
from .models import Student

# Register your models here.
admin.site.register(Subject)
admin.site.register(Student)
```

Signals

► Before creating subject

Change student

HISTORY

Name:

barry

Subject:

maths

sci

geography

phy

IT

cs

Mech

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Delete

Save and add another

Save and continue editing

SAVE

► After creating subject

Change student

HISTORY

Name:

barry

Subject:

maths

sci

geography

phy

IT

cs

Mech

embedded

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Delete

Save and add another

Save and continue editing

SAVE

- ▶ To store and retrieve data on site
- ▶ Say you have a user logged in, if the user tries to refresh the same login page
- ▶ It should not show up the page where user credentials are to be entered
- ▶ Instead redirect him to his home page

Sessions

- ▶ Django comes with sessions app installed
`'django.contrib.sessions'`
- ▶ Also, with session middleware
`django.contrib.sessions.middleware.SessionMiddleware`

Sessions

- ▶ In the get method, if the user has a session, then redirect him to his home page

views.py

```
class Login(View):  
    def get(self, requests):  
        if 'username' in requests.session:  
            groups = requests.session['group']  
            return redirect(groups+'/')  
        return render(requests, 'index.html')
```

- ▶ In post method, when the user logs in set the necessary session data

Sessions

- ▶ In post method, when the user logs in set the necessary session data

views.py

```
class Login(View):
    def post(self, requests):
        username = requests.POST.get('username')
        password = requests.POST.get('password')
        user = authenticate(username=username, password=password)
        if user:
            user = User.objects.get(username=username)
            user_groups = user.groups.all()

            if
user.groups.filter(name=self.management_grp_name).exists():

            requests.session['username'] = username
            requests.session['group'] = self.management_grp_name
            return redirect('management/')

... ..
```


Syndicating Feeds RSS

- ▶ What is it ?
Random rss from internet - <https://py-bucket.blogspot.com/feeds/posts/default>
- ▶ How in Django ?

models.py

```
from django.db import models

# Create your models here.
class Blog(models.Model):
    title = models.CharField(max_length=10)
    description = models.CharField(max_length=10)
    link = models.CharField(max_length=10, null=True)
```

Syndicating Feeds RSS

feeds.py

```
from django.contrib.syndication.views import Feed

from .models import Blog

class MyLatestPostFeeds(Feed):
    title = 'blog'
    description = 'hola! hope you are doing good'
    link = 'example.com/'

    def items(self):
        return Blog.objects.order_by('-id')[:5]

    def item_title(self, item):
        return item.title

    def item_description(self, item):
        return item.description

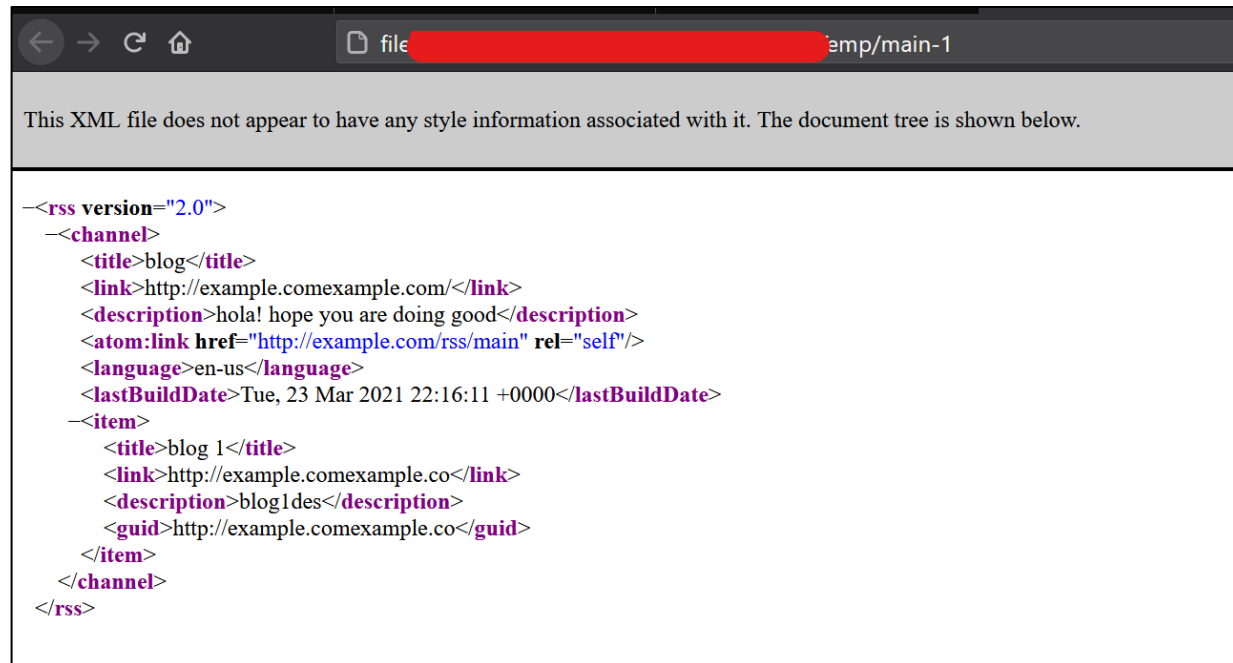
    def item_link(self, item):
        return item.link
```

Syndicating Feeds RSS

feeds.py

```
from myfeed.feeds import MyLatestPostFeeds

urlpatterns = [
    #...
    path('rss/main', MyLatestPostFeeds())
    # ..
]
```



The screenshot shows a web browser window with a dark theme. The address bar displays a file path: `file:///.../temp/main-1`. Below the address bar, a message states: "This XML file does not appear to have any style information associated with it. The document tree is shown below." The XML document tree is displayed in a light-colored area, showing the structure of an RSS 2.0 feed. The root element is `<rss version="2.0">`, followed by a `<channel>` element. Inside the channel, there is a `<title>` element with the value "blog", a `<link>` element with the value "http://example.comexample.com/", a `<description>` element with the value "hola! hope you are doing good", an `<atom:link href="http://example.com/rss/main" rel="self"/>` element, a `<language>` element with the value "en-us", and a `<lastBuildDate>` element with the value "Tue, 23 Mar 2021 22:16:11 +0000". Following the channel element is an `<item>` element. Inside the item, there is a `<title>` element with the value "blog 1", a `<link>` element with the value "http://example.comexample.co/", a `<description>` element with the value "blog1des", a `<guid>` element with the value "http://example.comexample.co/", and a `</item>` closing tag. The feed ends with a `</channel>` closing tag and a `</rss>` closing tag.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
  <channel>
    <title>blog</title>
    <link>http://example.comexample.com/</link>
    <description>hola! hope you are doing good</description>
    <atom:link href="http://example.com/rss/main" rel="self"/>
    <language>en-us</language>
    <lastBuildDate>Tue, 23 Mar 2021 22:16:11 +0000</lastBuildDate>
  </channel>
  <item>
    <title>blog 1</title>
    <link>http://example.comexample.co/</link>
    <description>blog1des</description>
    <guid>http://example.comexample.co/</guid>
  </item>
</rss>
```