Django

Getting Python in the web

Ahmed Moawad

Django Template Language

It's a language that you will love

Variables

```
{{ variable_name }}
```

library/index.html

```
<html>
  <head> ... </head>
  <body>
        <p> Hello, {{ name }} </p>
  </body>
</html>
```

```
Hello, Ahmed
```

library/views.py

```
from django.shortcuts import render
def index(request):
    return render(request, 'library/index.html', {'name': 'Ahmed'})
```

Tags

```
{% tag_name %}
```

library/index.html

```
<body>
  <ul>
  {% for m in movies %}
    <li>{{ m }}</li>
  {% endfor %}
  </ul>
</body>
```

- Forrest Gump
- The Dark Knight
- Inception

library/views.py

```
from django.shortcuts import render
def index(request):
    names = ['Forrest Gump', 'The Dark Knight', 'Inception']
    return render(request, 'persons/index.html',{'names': names})
```

for statement

`{% for .. in ..%} {% empty %} {% endfor %}`

library/index.html

```
<ul>
{% for m in movies %}
  <li>{{ m }}</li>
{% empty %}
  <li> No Movies in the List</li>
{% endfor %}
</ul>
```

- No Movies in the List

library/views.py

```
from django.shortcuts import render
def index(request):
    names = []
    return render(request, 'persons/index.html',{'names': names})
```

if statement

{% if %} {% elif %} {% else %} {% endif %}

library/index.html

```
{% if age <= 15 %}
  <p> Child </p>
{% elif age > 15 and age < 40 %}
  <p> Young Man </p>
{% else %}
  <p> Old Man </p>
{% endif %}
```

Child

library/views.py

```
from django.shortcuts import render
def index(request):
    age = 14
    return render(request, 'persons/index.html',{'age': age})
```

Template Inheritance

{% block %} {% endblock %} .... {% extends %}

base.html

```
<html>
  <head>
    <title>{% block title %}Default title{% end %}</title>
  </head>
  <body>
        {% block student %} Student Name {% end %}
  </body>
</html>
```

student.html

```
{% extends "base.html" %}
{% block title %}Student Page{% end %}
{% block student %} Ahmed{% end %}
```

Filters

```
{{ value | filter : options }}
```

library/index.html

```
<html>
  <body>
    <h2>{{ name | title }}</h2>
    <p>{{ body | truncatewords:5 }}</p>
  </body>
</html>
```

Home Page

This is my home page …

library/views.py

```
def index(request):
    name = "home page"
    body = "This is my home page. Please tell me your opinion."
    return render(request,'index.html',{'name': name, 'body': body})
```

Some useful Filters

Filter	value	Example	Output
add	value =3	{{ value | add : 3 }}	6
first	value = [1,2,4]	{{ value | first }}	1
join	value =['a','b','c']	{{ value | join: ':' }}	a:b:c
linebreaks	value ="Hi\nOS"	{{ value | linebreaks }}	Hi OS
pluralize	v = 4	{{ v }} value{{ v | pluralize}}	4 values

Comments

```
{# write your comment #}
```

url

```
<a href={{ url 'posts' }} >Posts</a>
```

Forms

Intro

Django Forms is a method to make a dynamic HTML Forms with its validation

library/forms.py

```
from django import forms


class AddBookForm(forms.Form):
    title = forms.CharField(label='Title', max_length=100)
    author = forms.CharField(label='Author', max_length=100)
    is_free = forms.BooleanField(label='Is Free')
    date_published = forms.DateField(label="Date Published")
```

How to use

library/views.py

```
from .forms import AddBookForm
from django.shortcuts import render
def index(request):
    form = AddBookForm()
    return render(request, 'library/index.html',{'form': form})
```

library/index.html

```
<h2> Add New Book </h2>
<form method="POST"> {% csrf_token %}
   {{ form.as_p }}
</form>
```

Title

Author

Is Free

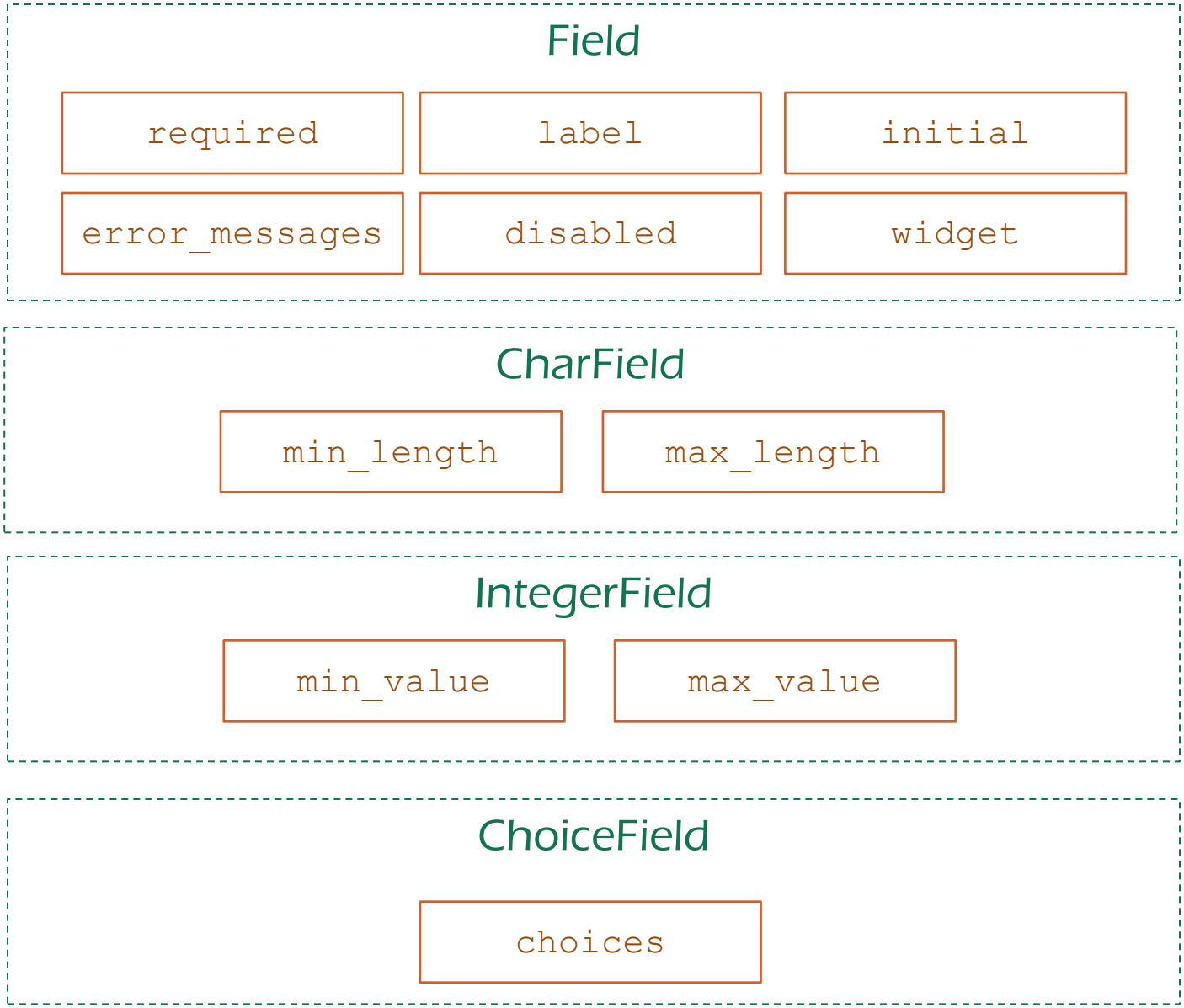
Form Class

`Form.is_bound()`	Check if the Form has populated by data or not
`Form.is_valid()`	Check if the Form valid or not
`Form.errors`	The errors list for all fields
`Form.fields`	The field list

Field – Widget Map

Field	Widget
CharField	TextInput
EmailField	EmailInput
IntegerField	NumberInput
BooleanField	CheckboxInput
ChoiceField	Select
DateField	DateInput

Fields Options



Example

library/views.py

```
def index_view(request):
        if request.method=='POST':
                form = AddBookForm(request.POST)
                if form.is_valid():
                        # Add New Book to database
                        return HttpResponse("Mission Complete")
        else:
                form = AddBookForm()
        return render(request,"library/add_book.html",{"form":form})
```

More on Views

Know more about Django Views

View Decorator

View Decorator is a method to make the view run under specific conditions

```
from django.views.decorators.http import require_http_methods


@require_http_methods(["GET"])
#or
@require_GET()
def my_view(request):
    # Get Method Actions
    pass
```

Class-based Views

```
# views.py
from django.http import HttpResponse
from django.views import View

class MyView(View):
    #method to be called if request Method is GET
    def get(self, request):
        return HttpResponse('GET result')
    #Method to be called if request Method is GET
    def post(self, request):
        return HttpResponse('Post result')
# urls.py
...
urlpatterns = [ url(r'^about/$', MyView.as_view()) ]
```

More Shortcuts

```
get_object_or_404(Model, *args, **kargs)
```

```
get_list_or_404(Manager, *args, **kargs)
```

```
get_object_or_404(QuerySet, *args, **kargs)
```

Get Object (List of objects) or redirect to http404 Page

```
from django.shortcuts import get_object_or_404, get_list_or_404
def my_view(request):
    book = get_object_or_404(Book,pk=2)
    return HttpResponse("Book Title is "+book.title)
    #or
    books = get_list_or_404(Book, author='Paulo Kauli')
    return render(request, 'library/book.html', books = books)
```

Generic Views

Intro

Generic Views is class-based Views that has a special purpose like listing & editing.

```
from django.http import HttpResponse
from django.views.generic import ListView, DetailView
from .models import Book
class BookList(ListView):
        model = Book

class BookView(DetailView):
        model = Book
```

```
urlpatterns = [
        url(r'^$', BookList.as_view()),
        url(r'^/book/(?P<pk>[0-9]+)$', BookView.as_view()),
]
```

Views - template Map

if app_name is library and model = Book

View	Template Name	Context
ListView	library /book_list.html	object_list
DetailView	library /book_detail.html	object
CreateView	library /book_form.html	form
UpdateView	library /book_update_form.html	form
DeleteView	library / book_confirm_delete.html	

More on URLs

Know more about Django URLs

URL Namespacing

namespace:url_name

library/urls.py

```
...

app_name = 'library' #namespace

urlpatterns = [ url(r'^$', views.index, name='index')]
```

library:index

reverse

```
reverse(view, args=None, kwargs=None)
```

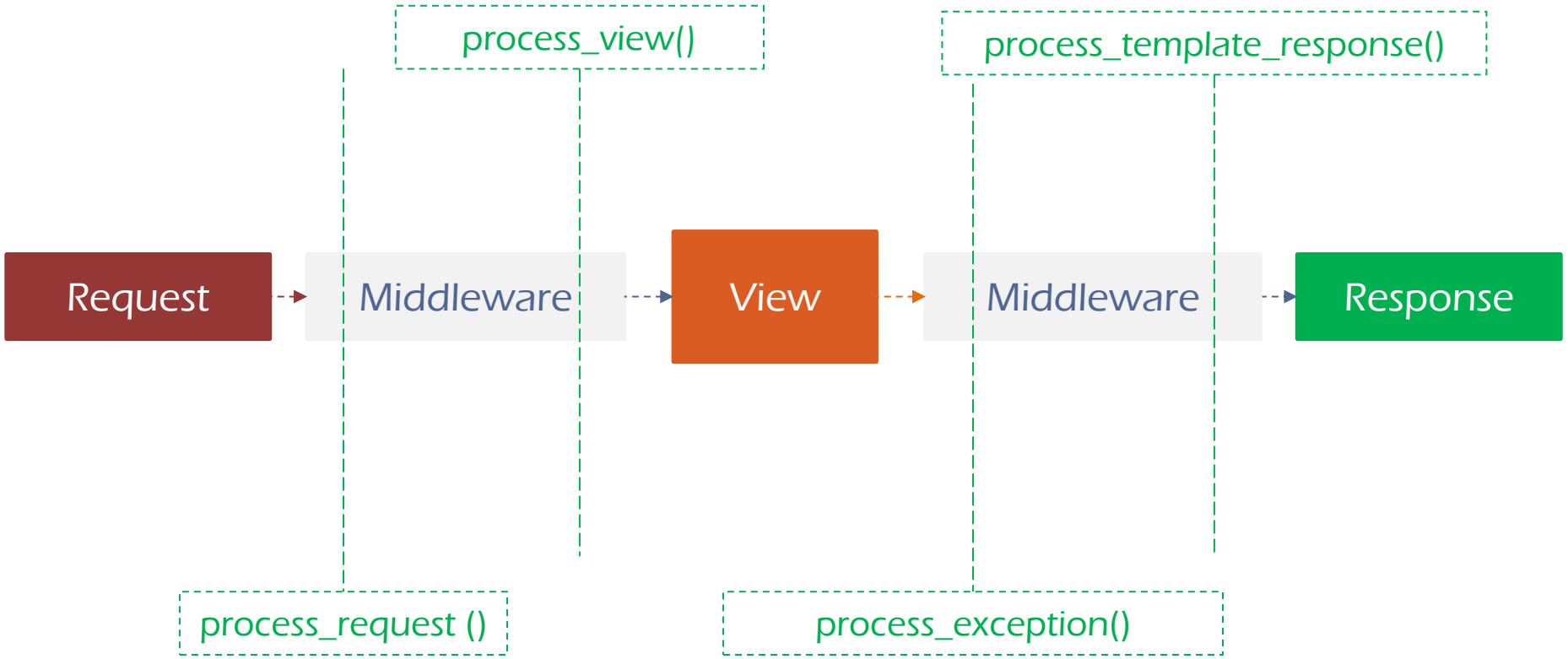
or

```
reverse(patternName, args=None, kwargs=None)
```

```
# urls.py

app_name = 'library' #namespace

urlpatterns = [

url(r'^post/([0-9]+)/comment/([0-9]+)$', view.posts, name='posts')]

# view.py
def index(request):
        url = reverse('posts', args=[2,3])
#or     url = reverse('library:posts', args=[2,3])
        return HttpResponseRedirect(url)    # '/posts/2/comment/3'
```

Middleware

Intro


process_view()
process_template_response()
Request
Middleware
View
Middleware
Response
process_request ()
process_exception()

Create Custom Middleware

library/middleware/dummymiddleware.py

```
try:
        from django.utils.deprecation import MiddlewareMixin
except ImportError:
        MiddlewareMixin = object
class DummyMiddleware(MiddlewareMixin):
        def process_request(self,req):
                req.dept = "OS"
                return None
```

mysite/settings.py

```
MIDDLEWARE = [
        # Other Middleware Classes,
        "library.middleware.dummymiddleware.DummyMiddleware"
]
```

Useful Built-in Middleware

`ExceptioMiddleware`	Handles 400 and 500 Status Exceptions
`SessionMiddleware`	Create a sessions for the clients to make the relationship more clearified.
`CommonMiddleware`	Handles some common cases like appending Slash or prepending www to the url to match the reserved one.
`CSRFViewMiddleware`	To protect you're app from CSRF with generating CSRF Tokens.
`AuthenticationMiddleware`	It's a framework has built for user authentication operations.
`MessageMiddleware`	It's a pretty. It used for send contextual messages to the client based on the action states.

Sessions

Intro

The session framework lets you store and retrieve arbitrary data on a per-site-visitor basis. It stores data on the server side and abstracts the sending and receiving of cookies.

settings.py

```
INSTALLED_APPS = [
        # Other Apps,
        "django.contrib.sessions"
]

MIDDLEWARE = [
         # Other Middleware Classes,
        "django.contrib.sessions.middleware.SessionMiddleware"
]
```

How to use

request.session

views.py

```
def login(request,user_id):
      #Setting Session item
      request.session['member_id'] = user_id
      return HttpResponse("You are logged in")


def view_profile(req):
     #Getting Session item
     if 'member_id' in req.session and req.session['member_id']:
          return HttpResponse("View Profile")
     else:
          return HttpResponse("Please login to view ")
```

methods

```
request.session.get(key,default=None)
```

```
request.session.pop(key,default=__not_given)
```

```
request.session.clear()
```

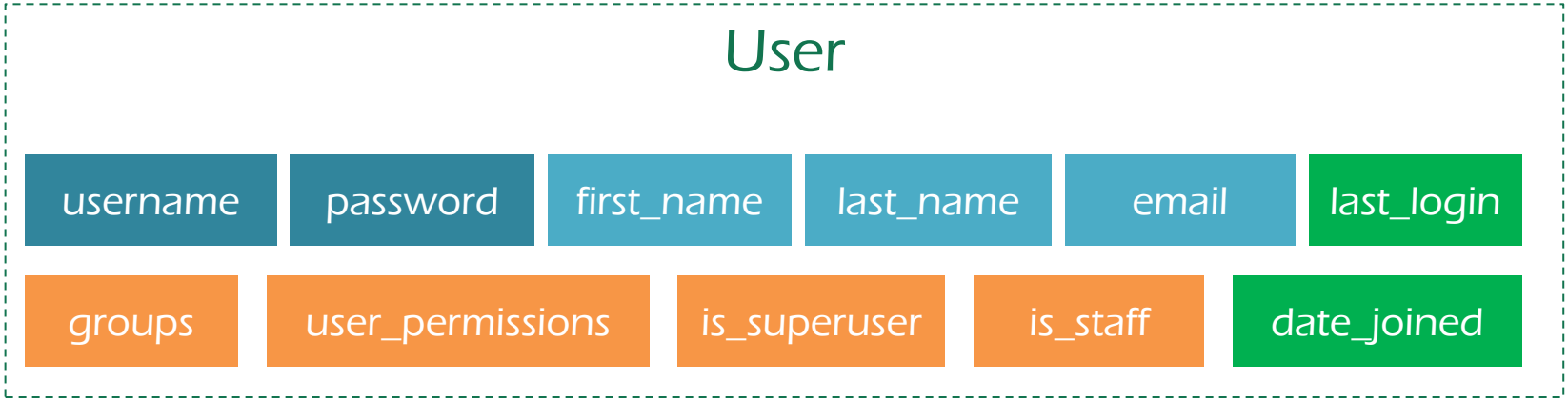
```
request.session.set_expiry(value)
```

settings

`SESSION_COOKIE_AGE`	Set the Session Cookie Age
`SESSION_EXPIRE_AT_BROWSER_CLOSE`	If True, Session Cookie Age will be deleted after browser closed
`SESSION_COOKIE_NAME`	Set the Session Cookie Name

Authentication

User




```
from django.contrib.auth.models import User
user = User.objects.create_user('Ahmed','am@gmail.com','120829')
```

User Authentication Example

```
from django.contrib.auth import authenticate, login, logout
def login_view(request):
    uname = request.POST['username']
    pword = request.POST['password']
    user = authenticate(username=uname, password=pword)
    if user is not None:
        login(request,user)
    else:
        # Do actions for that the login process failed


def any_view(request):
    if request.user.is_authenticated():
        # Do actions for Logged in Users
    else:
        # Do actions for Guests
```

Thank You