Table of Content

Contents

Table of Content 1

Django ver 1

Database/Model 2

URL 3

VIEW/Exception 3

Write URl 4

Form/HTML 4

POST 4

Less code is better 4

TestCase 5

Style 5

Admin 6

Custom admin 6

-Django ver **: 1.11**

**Add app**:

python manage.py startapp polls

Urls polls: polls/urls.py

**from** **django.conf.urls** **import** url

**from** **.** **import** views

urlpatterns = [

url(r'^$', views.index, name='index'),

]

**Urls polls:** mysite/urls.py

**from** **django.conf.urls** **import** include, url

**from** **django.contrib** **import** admin

urlpatterns = [

url(r'^polls/', include('polls.urls')),

url(r'^admin/', admin.site.urls),

]

-Database/Model

**Models:** polls/models.py

**from** **django.db** **import** models

**class** **Question**(models.Model):

question\_text = models.CharField(max\_length=200)

pub\_date = models.DateTimeField('date published')

**~~def~~** ~~was\_published\_recently(self):~~

**~~return~~** ~~self.pub\_date >= timezone.now() - datetime.timedelta(days=1)~~

**class** **Choice**(models.Model):

question = models.ForeignKey(Question, on\_delete=models.CASCADE)

choice\_text = models.CharField(max\_length=200)

votes = models.IntegerField(default=0)

Setting: mysite/settings.py

INSTALLED\_APPS = [

'polls.apps.PollsConfig',

…

]

Console:

python manage.py makemigrations polls

python manage.py migrate

Code API:

python manage.py shell

**import** **django**

-URL

polls/urls.py

app\_name = 'polls'

urlpatterns = [

url(r'^$', views.index, name='index'),

url(r'^(?P<question\_id>[0-9]+)/$', views.detail, name='detail'),

url(r'^(?P<question\_id>[0-9]+)/results/$', views.results, name='results'),

url(r'^(?P<question\_id>[0-9]+)/vote/$', views.vote, name='vote'),

]

-VIEW/Exception

{% block content %}

{% endblock %}

{% **extends** "base\_generic.html" %}

**def** index(request):

latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

context = {'latest\_question\_list': latest\_question\_list}

**return** render(request, 'polls/index.html', context)

**def** detail(request, question\_id):

**try**:

question = Question.objects.get(pk=question\_id)

**except** Question.DoesNotExist:

**raise** Http404("Question does not exist")

**return** render(request, 'polls/detail.html', {'question': question})

question = get\_object\_or\_404(Question, pk=question\_id)

-Write URl

{% **url** 'detail' question.id %}

{% **url** 'polls:detail' question.id %} -> point with namespace

-Form/HTML

{%

Write code here

%}

{{ show data here }}

{% **csrf\_token** %}

https://docs.djangoproject.com/en/1.11/topics/forms/modelforms/

https://tutorial.djangogirls.org/en/django\_forms/

-POST

request.POST['name of input']

-Less code is better

app\_name = 'polls'

urlpatterns = [

url(r'^$', views.IndexView.as\_view(), name='index'),

url(r'^(?P<pk>[0-9]+)/$', views.DetailView.as\_view(), name='detail'),

url(r'^(?P<pk>[0-9]+)/results/$', views.ResultsView.as\_view(), name='results'),

url(r'^(?P<question\_id>[0-9]+)/vote/$', views.vote, name='vote'),

]

polls/views.py

**class** **IndexView**(generic.ListView):

template\_name = 'polls/index.html'

context\_object\_name = 'latest\_question\_list'

**def** get\_queryset(self):

**return** Question.objects.order\_by('-pub\_date')[:5]

**class** **DetailView**(generic.DetailView):

model = Question

template\_name = 'polls/detail.html'

**class** **ResultsView**(generic.DetailView):

model = Question

template\_name = 'polls/results.html'

**def** vote(request, question\_id):

... *# same as above, no changes needed.*

-TestCase

**class** **QuestionModelTests**(TestCase):

**def** test\_was\_published\_recently\_with\_future\_question(self):

time = timezone.now() + datetime.timedelta(days=30)

future\_question = Question(pub\_date=time)

self.assertIs(future\_question.was\_published\_recently(), **False**)

-Style

**static** : folder in polls project

polls/static/polls/css/style.css

polls/templates/polls/index.html

{% **load** static %}

<**link** rel="stylesheet" type="text/css" href="{% **static** 'polls/css/style.css' %}" />

Add background:

**background**: **white** url("images/background.gif") **no-repeat** **right** **bottom**;

-Admin

python manage.py createsuperuser

make poll app modifile in admin: polls/admin.py:

**from** **django.contrib** **import** admin

**from** **.models** **import** Question

admin.site.register(Question)

-Custom admin

polls/admin.py

**class** **QuestionAdmin**(admin.ModelAdmin):

fields = ['pub\_date', 'question\_text']

admin.site.register(Question, QuestionAdmin)

**class** **QuestionAdmin**(admin.ModelAdmin):

fieldsets = [

(**None**, {'fields': ['question\_text']}),

('Date information', {'fields': ['pub\_date']}),

]

admin.site.register(Question, QuestionAdmin)

Add related Object

polls/admin.py

**from** **django.contrib** **import** admin

**from** **.models** **import** Choice, Question

*# ...*

admin.site.register(Choice)

Or:

**class** **ChoiceInline**(admin.StackedInline):

model = Choice

extra = 3

**class** **QuestionAdmin**(admin.ModelAdmin):

fieldsets = [

(**None**, {'fields': ['question\_text']}),

('Date information', {'fields': ['pub\_date'], 'classes': ['collapse']}),

]

inlines = [ChoiceInline]

admin.site.register(Question, QuestionAdmin)

**class** **ChoiceInline**(admin.TabularInline):

Custom List:

polls/admin.py

**class** **QuestionAdmin**(admin.ModelAdmin):

~~list\_display = ('question\_text', 'pub\_date')~~

list\_display = ('question\_text', 'pub\_date', 'was\_published\_recently')

Filter:

list\_filter = ['pub\_date']

Search:

search\_fields = ['question\_text']