Django

class-based views

План

- в чем собственно проблема?
- class-based views
- TemplateView, ListView, ItemView
- forms and class-based views
- mixins

От частного к общему

Если внимательно посмотреть на views для вывода айтемы объекта, или вывода списка объектов и даже редактирования и добавления новых объектов которые мы насоздавали в процессе выполнения ДЗ. То станет очевидно что все они мало чем отличаются от аппликейшина к аппликейшину. Отличается только модель из которой мы делаем выборку (ну может еще и шаблон).

От частного к общему

```
def students_item(request, student_id):
    student = get_object_or_404(Student, id=student_id)
    return render(request, 'students/item.html',
                  {'student': student})
```

```
def courses_item(request, course_id):
    course = get_object_or_404(Course, id=course_id)
    return render(request, 'courses/item.html',
                  {'course': course})
```

От частного к общему

```
def view_item(request, obj_id, obj_class):
    class_name = obj_class.__name__.lower()
    obj = get_object_or_404(obj_class, id=obj_id)
    return render(request,
                  '%s/item.html' % class_name,
                  {class_name: obj})

def courses_item(request, course_id):
    return view_item(request, course_id, Course)

def students_item(request, student_id):
    return view_item(request, student_id, Student)
```

От частного к общему

Т.е. мы один раз создали общий метод с логикой и далее уже просто его вызываем для частных случаев.

Аналогично можно поступить и со view для вывода списка айтемов.

От частного к общему

```
def student_add_edit(request, student_id=None):
    student = None
    if student_id is not None:
        student = get_object_or_404(Student, id=student_id)

    if request.method == 'POST':
        form = StudentModelForm(request.POST,
                                instance=student)
        if form.is_valid():
            student = form.save()
            return redirect('student-edit', student.id)
    else:
        form = StudentModelForm(instance=student)
    return render(request, 'students/edit.html', {'form': form})
```

class-based views

Наивно было бы думать, что в django про это не подумали и не позаботились :)

Только для сбора и конфигурации параметров используются классы!

```
class CView(object):
    def as_view(request, ...):
        ...
        return HttpResponse(...)
```

Дока наше все!

https://docs.djangoproject.com/en/1.7/topics/class-based-views/

TemplateView

```
from django.views.generic import TemplateView

class HomeView(TemplateView):
    template_name = "index.html"

urlpatterns = patterns('',
    (r'^$', HomeView.as_view()),

    (r'^$', TemplateView.as_view(
                        template_name="index.html")),
)
```

DetailView

```
from django.views.generic.detail import DetailView
from students.models import Student

class StudentView(DetailView):
    template_name = “student/item.html"
    model = Student
```

student/item.html

```
<h1>{{ object.name }}</h1>
```

context_object_name

ListView

```
from django.views.generic.list import ListView
from students.models import Student

class StudentListView(ListView):
    #template_name = "students/student_list.html"
    model = Student
```

student/student_list.html

```
{% for student in object_list %}
    <li>{{ student.name }}</li>
{% endfor %}
```

ListView customisation

```
from django.views.generic.list import ListView
from students.models import Student

class StudentView(ListView):
    template_name = “students/student_list.html”
    model = Student
    queryset = Student.objects.filter(package='vip')

    def get_queryset(self):
        filter_package = self.request.GET.get(‘package’, ‘vip’)
        return Student.objects.filter(package= filter_package)
```

Дока наше все!

https://docs.djangoproject.com/en/1.7/topics/class-based-views/generic-editing/

FormView

```
from django.views.generic.edit import FormView
from students.models import Student

class StudentEditForm(FormView):
    template_name = “students/edit.html”
    form_class = StudentForm
    success_url = ‘/students/list/’

    def form_valid(self, form):
        # process form-data here
        form.cleaned_data ...
        return super(StudentEditForm, self).form_valid(form)
```

Model FormView

```
from django.views.generic.edit import (CreateView,
                                       UpdateView, DeleteView)
from students.models import Student


class StudentCreate(CreateView):
    model = Student
    template_name = 'students/student_form.html'

class StudentUpdate(UpdateView):
    model = Student

class StudentDelete(DeleteView):
    model = Student
    template_name = 'students/student_confirm_delete.html'
```

Model FormView Custom form

```
from django.views.generic.edit import CreateView,
from students.models import Student

class StudentForm(forms.ModelForm)
    model = Student
    exclude = (‘note’, ‘kurator_group’)

class StudentCreate(CreateView):
    model = Student
    form_class = StudentForm
    template_name = ‘students/student_form.html’
```

Class View Custom context

```
from django.views.generic.detail import DetailView
from students.models import Student

class StudentView(DetailView):
    model = Student

    def get_context_data(self, **kwargs):
        context = super(StudentView,
self).get_context_data(**kwargs)
        context['courses'] = Course.objects.all()
        return context
```

Class View get post methods

```
from django.views.generic import View
from django.http import HttpResponse

class SomeView(View):
    def get(self, **kwargs):
        # view logic here!
        return HttpResponse("Hello world!")

    def post(self, **kwargs):
        # view logic here!
        return HttpResponse("Post!")
```

Class View dispatch method

```
from django.views.generic import View
from django.http import HttpResponse

class SomeView(View):

    def dispatch(self, **kwargs):
        # view logic here!
        return super(SomeView, self).dispatch(*args, **kwargs)
```

Дока наше все!

https://docs.djangoproject.com/en/1.7/topics/class-based-views/mixins/

Mixins

```
class CoursesMixin(object):
    def get_context_data(self, **kwargs):
        context = super(CoursesMixin,
self).get_context_data(**kwargs)
        context['courses'] = Courses.objects.all()
        return context

    def dispatch(self, **kwargs):
        # extra view logic here
        return super(CoursesMixin, self).dispatch(*args,
**kwargs)

class SomeView(CoursesMixin, TemplateView):
    template_name = “index.html”
```

Mixins

```
from django.views.generic import View
from django.http import JsonResponse

class JSONResponseMixin(object):
    def render_to_response(self, context,
                           **response_kwargs):
        return JsonResponse(context,
                            **response_kwargs)

class JSONView(JSONResponseMixin, View):
    pass
```