Лекция 16 марта 2016

Основы веб-разработки (первый семестр)

https://s.mail.ru/dhV5uzBoEDMN/stackoverflow.zip
https://s.mail.ru/352jXFCqPXAJ/stackoverflow%202.zip
https://s.mail.ru/53BYSLLehiZw/web3.pdf

ДЗ и описания будут позже, после обновления поста сделаю рассылку

UPD.

Итак, ДЗ. Ваша задача - создать внутри вашего django-проекта несколько приложений, описать в них простейшие модели для основных вещей, которые будут в вашем проекте, добавить их в админку django и реализовать простенькие странички для показа объектов этих моделей.

Давайте я вкратце напомню, что в проекте должен быть реализован следующий минимум -
1. Наличие авторизации
2. Наличие создаваемого пользователем контента
3. Комментарии
4. Пользовательские действия «в одно нажатие» (лайк, дизлайк, вроде того)
5. Проверка прав

Поясняю по каждому пункту:
1. Наличие авторизации - имеется ввиду возможность пользователя залогиниться и разлогиниться на проекте, реализованная вами. Логин в админку django - это не про то (но пока что можете использовать его для того единственного пользователя, которого мы создали). Позже мы реализуем собственные страницы логина и выхода с сайта.
2. Наличие создаваемого пользователем контента - например, пост в блогодвижке. Или фотогалерея. Или заданный на stackoverflow вопрос. Всё то, что пользователь по идее может запостить на сайт самостоятельно.
3. Комментарии - здесь я подразумеваю не обязательно именно каменты, я имею ввиду любую модель данных, которая тоже создается пользователем, и бессмысленная сама по себе, без того объекта, к которому она "привязана". Тот же комментарий, например, обязательно привязан к посту, к которому его написали. Или вопрос на stackoverflow привязан к ответу. Если же вы, например, пишете сайт-аукцион, то к аукциону там будет привязаны ставки. Надеюсь, эта мысль ясна.
4. Пользовательские действия в "в одно нажатие" - проще всего это объяснить с помощью лайков. Пользователь лайкнул пост, в посте должен увеличиться счетчик лайков на один, плюс мы должны запомнить, что именно этот юзер лайкнул именно этот пост. То есть, собственно контента (какого-то текста) у лайка нет. Важен сам факт его наличия. Так же, как и в предыдущем пункте - если вы

Прямой эфир

Марина Даньшина 16 минут назад
Расписание занятий → Расписание пересдач 0

Екатерина Рогушкова 8 часов назад
Получение значков достижений на портале (ачивки) 2

Ольга Августан Вчера в 18:41
Опросы → Получение значков достижений на портале (ачивки) 2

Ольга Августан Вчера в 13:19
Общие вопросы → Завершение семестра: последние итоговые встречи 0

Ольга Августан Вчера в 13:12
Завершение семестра: итоговые встречи по дисциплинам 10

Марина Даньшина 25 Мая 2016, 14:01
Разработка на Java (первый семестр) → Забыли зонтик 0

Ksenia Sternina 25 Мая 2016, 13:38
Домашнее задание - Юзабилити-тестирование 4

Летяго Владимир 25 Мая 2016, 01:40
Разработка приложений на Android - 1 (второй семестр) → Контрольный рубеж 0

Летяго Владимир 24 Мая 2016, 22:45
Домашнее задание №3 10

Вячеслав Ишутин 24 Мая 2016, 22:31
Разработка на C++ (открытый курс) → Итоговая встреча по дисциплине 0

Вячеслав Ишутин 24 Мая 2016, 22:22
Разработка на C++ (открытый курс) → Итоговое занятие. Презентация курсовых проектов. 0

Александр Агафонов 24 Мая 2016, 19:17
Мастер-класс Ильи Стыценко онлайн «Использование OAuth 2 в приложениях на Django» 1

Марина Даньшина 23 Мая 2016, 20:59
Мероприятия → Мастер-класс Ильи Стыценко онлайн «Использование OAuth 2 в приложениях на Django» 1

Ольга Августан 23 Мая 2016, 17:56
Разработка приложений на iOS - 1 (второй

Связь с разработчиками

вместо лайков придумаете нечто другое, похожее по функционалу - это ок.
5. Проверка прав - имеется ввиду тот факт, что если пользователь создал пост - то и редактировать или удалить его может только этот пользователь. Ну и в отношении всего остального также.

Если у вас сомнения в том, как построить модель для своего проекта, подходит ли для проекта выбранная вами тематика - пишите здесь или в личку, помогу. Собственно, многие из вас так и делают, получают ответ и им становится проще :)

Теперь про ДЗ более подробно - вам нужно реализовать модели и странички для того, что я указал в пп. 2 и 3 (создаваемый юзером контент, основной и то, что я назвал "комментариями". Под страничками я, естественно, имею ввиду не просто html-страницы, а возможность зайти на сайт и увидеть тот или иной объект или группу объектов.

Пример того, что мы делали на лекции - в аттаченных файлах выше, плюс вот вам небольшая памятка:

0. Работа с проектом.
Все действия с manage.py стоит выполнять из папки src (там же собственно manage.py и лежит). При этом нужно находиться в питоньем виртуальном окружении (source env/bin/activate). Просто убеждайтесь что в приглашении командной строки у вас есть в начале обозначение (env).

1. Приложения в django.

Всё просто. Приложение создается командой ./manage.py startapp имя-приложения, после чего имя-приложения нужно добавить в переменную INSTALLED_APPS в настройках (settings.py).
Внутри проекта создастся одноименная папка, в ней мы создаем еще две - templates и static (для шаблонов приложения и для статических файлов). Плюс файл для роутинга urls.py
Создайте себе как приложение например core (./manage.py startapp core), чтобы сложить туда шаблоны и статику, которые нужны для всего проекта (статики у вас пока наверное не будет, а вот базовый шаблон в ходе работы с ДЗ появится).
Модели описываем в файле models.py, контроллеры - views.py, урлы - в urls.py.
Как правильно делить проект на приложения - как минимум создайте отдельное приложение для той модели данных, которая будет у вас основной (у кого-то блог, у кого-то вопрос, у кого-то аукцион итп). И для второй модельки, которая привязана к первой. А большего на данном этапе вам и не нужно, так что у вас получается три приложения (вместе с core). Если кто-то ну очень уверен что эти две модели данных логически неотделимы друг от друга - ну пусть будет не три приложения, а два, вполне ок.

семестр) → Завершение семестра 0

Ольга Августан 23 Мая 2016, 17:16
Итоговая встреча по дисциплине 2

Ольга Августан 23 Мая 2016, 17:07
Основы веб-разработки (первый семестр) → Итоговая встреча по дисциплине 0

Ольга Августан 23 Мая 2016, 17:06
Основы мобильной разработки (первый семестр) → Итоговая встреча по дисциплине 2

Ольга Августан 23 Мая 2016, 17:05
Проектирование интерфейсов мобильных приложений (второй семестр) → Итоговая встреча по дисциплине 0

Ольга Августан 23 Мая 2016, 17:02
Проектирование СУБД (второй семестр) → Итоговая встреча по дисциплине 0

Ольга Августан 23 Мая 2016, 16:59
Общие вопросы → Завершение семестра: итоговые встречи по дисциплинам 10

Весь эфир

Блоги

Основы веб-разработки (первый семестр) 26,04
Разработка на C++ (открытый курс) 16,97
Основы мобильной разработки (первый семестр) 14,70
Разработка на Java (первый семестр) 13,58
Общие вопросы 5,71
Разработка приложений на Android - 1 (второй семестр) 5,66
Стажировка 4,52
Проектирование интерфейсов мобильных приложений (второй семестр) 3,42
Разработка приложений на iOS - 1 (второй семестр) 2,29
Проектирование СУБД (второй семестр) 2,26

2. Описываем модели данных.

Это делается в файлах models.py, подход такой - создаем класс-наследник от models.Model, в качестве аттрибутов описываем ему необходимые поля.
Про различные типы полей можно почитать например тут https://docs.djangoproject.com/en/1.9/ref/models/fields/ (официальная дока) или тут http://djbook.ru/rel1.8/ref/models/fields.html (её перевод)
Вкратце, наиболее употребимы следующие поля (названия у них вполне доносят смысл):
models.CharField - строк, обязателен аргумент max_length - максимальная длина строки
models.TextField - для больших текстов (например, название поста - CharField, текст поста - TextField)
models.IntegerField - для целых чисел
models.FloatField - для дробных чисел
models.DateField - для дат (date), необязательный аргумент auto_now_add - "всегда автоматически ставить текущую дату при создании"
models.DateTimeField - для дат вместе со временем (datetime), аргумент auto_now_add аналогично DateTime
models.ForeignKey - id какой-либо другой модели, обязательный параметр - имя модели

Про models.ForeignKey - данное поле - это возможность привязать одну модель к другой. Об этом буду более подробно рассказывать на следующих лекциях, сейчас же - просто пример:

(в приложении blogs)

```
class Post(models.Model):
title = models.CharField(max_length=255)
text = models.TextField()
```

(в приложении comments)

```
class Comment(models.Model):
text = models.TextField()
post = models.ForeignKey('blogs.Blog')
```

Вот так можно создать простейшие модели блога и комментария к нему. Благодаря полю ForeignKey мы указываем принадлежность одного объекта модели к другому. То есть, допустим у нас переменная p - объект класса пост в блоге, а переменная c - объект класса комментарий, тогда мы сможем сделать подобное:
c.post.title (заголовок поста, к которому создан камент c)
p.comment_set.all() (list из всех комментариев к посту p)

Подробней об этом я расскажу вживую, сейчас же можно просто пользоваться

этой частью функционала

Если для модели нужно хранить дату ее создания, выручает created_at = models.DateTimeField(auto_now_add=True)
Лучше хранить дату создания вообще для всего :)

3. Создаем нужные структуры в базе данных.
Здесь особо не о чем рассказывать - вызываем последовательно ./manage.py makemigrations и ./manage.py migrate - первая команда заставит django записать сделанные вами изменения моделей в виде миграций (это просто скрипты на питоне в папке migrations), вторая команда - применит к базе данных те миграции, которые еще не применялись.
Если потом добавим поля к модели - нужно будет снова запустить последовательно первую и вторую команду, чтобы эти поля добавились в структуру данных в базе.

4. Выводим наши модели данных в админку.
Здесь всё очень просто - приложение django.contrib.admin уже включено в новом проекте по умолчанию, и прекрасно работает. Просто находим в приложении файл admin.py, импортируем в нем все модели нашего приложения и подключаем модель к админку функцией admin.site.register

Например:

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

admin.site.register(Post)
```

С этого момента в админке мы сможем управлять всеми объектами, созданными по этой модели.
Кстати, в саму админку можно попасть по ссылке localhost/admin/, предварительно создав первого (и пока единственного) пользователя сайта командой ./manage.py createsuperuser

5. Создаем базовый шаблон представлений (основной html-шаблон сайта, грубо говоря).
Про шаблоны в django можно почитать к примеру тут
http://djbook.ru/rel1.8/ref/templates/language.html
Пусть это будет файлик base.html в папке templates приложения core. Пусть в нем будет как минимум два блока - title и content (первый для заголовка текущей просматриваемой страницы, второй собственно для содержимого. Блоки вставляются тегом {% block имя-блока %}{% endblock %}

Все остальные шаблоны будут наследоваться от этого, то есть будут начинаться

со строки {% extends "base.html" %}, а блоки title и content будут в них переопределяться (то есть тот же block-endblock, просто с контентом посередине, типа {% block title %}Главная страница сайта{% endblock %})

6. Создаем список роутов (шаблонов ссылок) в каждом приложении и подключаем его в основной urls.py
Почитать можно тут
http://djbook.ru/rel1.8/topics/http/urls.html#including-other-urlconfs

Пока что контроллеров у нас нет, просто создаем каркас для будущих урлов проекта. Можете ориентироваться на файлик application/urls.py и news/urls.py в моем примере (том, что я писал в течение лекции). Почитать про это можно тут

Например:

(blogs/urls.py)

```
from django.conf.urls import url

urlpatterns = [
]
```

(applications/urls.py)

```
...
urlpatterns = [
url(r'^admin/', admin.site.urls),
url(r'^blogs/', include('blogs.urls', namespace="blogs")),
]
```

7. Последовательно создаем нужные контроллеры и шаблоны, контроллеры прописываем в urls.py приложений
Почитать можно тут http://djbook.ru/rel1.8/ref/class-based-views/index.html
Предполагаю что вам нужно как минимум иметь, для вашей основной модели:
- Страничку со всеми объектами модели (список постов)
- Страничку с отдельным объектом модели (отдельный пост + список комментариев)

Используем django class-based views (например, DetailView из примера - это и есть class-based-view)
Фактически, вам понадобится только DetailView и ListView

Как их использовать, должно быть видно из примера, сейчас же подскажу кратко:

Для отображения списка объектов используем ListView, его параметры:
model - имя модели, список объектов которой хотим показывать (предварительно

нужно ее импортировать)
template_name - имя шаблона (т.н. представления), с помощью которого хотим показать список

В шаблоне у нас будет список объектов в переменной {{ object_list }}, показать каждый можно в цикле (тег {% for %})

Пример:

(blogs/views.py)

```
class PostList(ListView):
template_name = "blogs/post_list.html"
model = Post
```

(blogs/templates/blogs/post_list.html) - кто первый ответит в каментах, зачем мы создаем еще одну папку blogs - получит +1 балл :)

{% extends "base.html" %} (файл base.html будет последовательно искаться во всех приложениях, пока не будет найден в приложении core)

```
{% block title %}Список постов{% endblock %}

{% block content %}
{% for post in object_list %}
<div>
<div>{{ post.title }}</div>
<div>{{ post.text }}</div>
</div>
{% endfor %}
{% endblock %}
```

(blogs/urls.py)

```
from .views import *

urlpatterns = [
url(r'^posts/$', PostList.as_view(), name="post_list"),
]
```

Таким образом, зайдя на урл вида localhost/blogs/posts/ , вы должны увидеть список постов (можно добавить несколько через админку и так и проверять)

Страница одного объекта делается через DetailView, она есть в примере
Добавлю только, что методы, предоставляемые нам полем ForeignKey можно использовать и в шаблонах - то есть {% for comment in object.comment_set.all %}... {% endfor %} вполне будет работать, если в переменной object в шаблоне у нас

лежит объект класса Post

7. Небольшой справочник по HTML и тегах шаблонизатора django

Для тех, кто не силен в верстке - мы посвятим ей отдельную лекцию, пока же не заморачиваемся на красоты, так что можно вообще использовать всего два тега, по идее их хватит
какой-то текст - текст будет показан на отдельной строке
текст текст будет являться ссылкой, которая будет вести на адрес, описанный в href

Теги и фильтры django-шаблонов можно найти здесь http://djbook.ru/rel1.8/ref/templates/builtins.html , по идее вам оттуда сейчас нужны: extends, block, for, url да и должно хватить.

Требования и оценка:
5 баллов - правильное описание моделей, включая админку
5 баллов - работающая страница со списком объектов "основной" модели (каждый пункт списка содержит ссылку на собственную страничку объекта)
5 баллов - работающая страница с детальным описанием модельки и списком связанных с ней

Всего, как видите, 15 баллов.

Пример:
Заходим на ссылку /blogs/posts/, видим там список постов, каждый пост - дата публикации и заголовок, заголовок является ссылкой и ведет на /blogs/posts/(id поста)/, то есть кликаем на пост и видим страничку с текстом и списком комментариев.

Имейте ввиду, приведенные мной выше примеры - в целом не для бездумного копипаста в свой проект ;)
Дело в том, что я их писал вот прямо сейчас прямо вот тут и даже не проверял, нет ли в них ошибок.

Также имейте ввиду, что оценивать я буду не только наличие чего-либо работающего в проекте, но и правильность исполнения (например, непротиворечивое описание моделей, грамотно подключенные urls.py...). Если прочитали весь пост и слушали лекцию - по идее, у вас должно остаться понимание того, как правильно, а как нет :)

Если такого понимания не сложилось, и вы сомневаетесь, верно ли вы сделали

"вот тут вот" - обязательно спрашивайте, я отвечу.

Если какой-то из вопросов кажется вам не достаточно освещенным для полного выполнения ДЗ - обязательно сообщите, я дополню пост.

Если не уверены в том, что какой моделью должно стать в выбранной вами тематике - тоже можете спросить.

Если остались еще какие-либо вопросы - можете писать в личку, или публиковать их в виде комментариев.

Удачной подготовки к семинару :)