Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

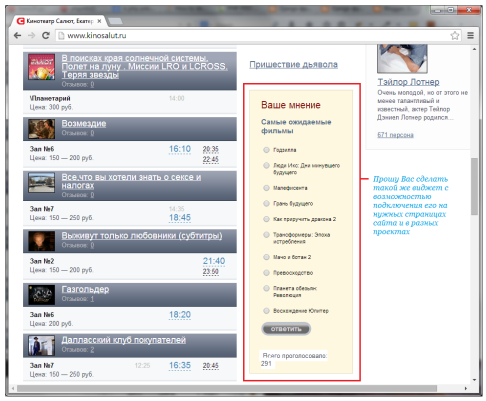
суббота, 25 февраля 2012 г.

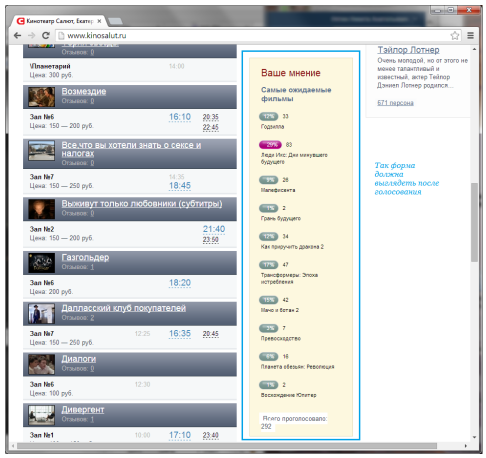
Опрос (ч.1)

Вас попросили создать веб-приложение для проведения опросов. Вы, как человек новатор, решили сделать его на веб-фреймворке Django.

Задание

Заказчик выслал вам скриншот того, что он хочет получить.





Требования:
Frontend:
1. На странице отображать панель проводимого в данный период голосования.
2. Проголосовав, пользователь должен увидеть число голосов по каждому ответу.

Backend:

1. Создание опроса через админку.
2. У каждого опроса должна быть дата публикации.
3. Каждый опрос может иметь N вариантов ответа.
4. Число проголосовавших по каждому ответу можно задавать вручную.

Этап 1. Подготовка

1.1. Нам нужно разобраться с тем, как мы будем хранить данные. В одной таблице? В двух? Какова структура таблиц? Нарисуйте возможную схему таблиц базы данных на листе бумаги. Вот что у нас

Архив блога
► 2014 (13)
► 2013 (38)
▼ 2012 (6)
▼ Февраль (6)
Опрос (ч.4)
Опрос (ч.3.2)
Опрос (ч.3.1)
Опрос (ч.2)
Опрос (ч.1)
Пролог

Обо мне

yesnik
Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.
Просмотреть профиль

получилось:

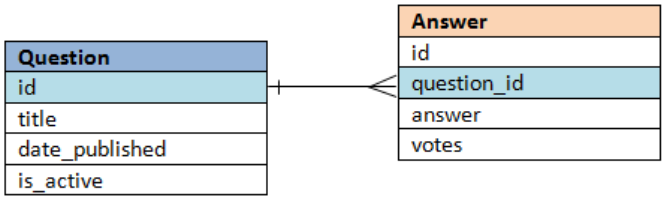


Схема БД

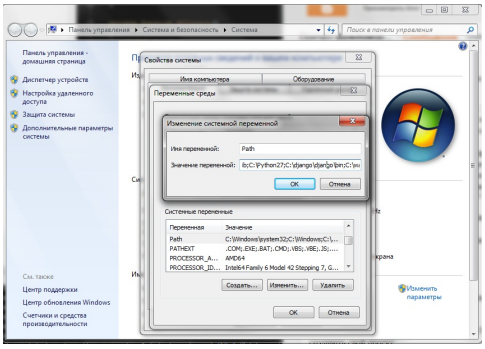
Полученный рисунок показывает идею, согласно которой данные будут храниться в базе данных.

1.2. Мы можем создать таблицы в БД вручную, но зачем? Ведь у нас есть Django! Его особенность состоит в том, что таблицы он создает сам на базе кода python. На следующем этапе мы напишем код моделей.

1.3. Очень важно **настроить переменные среды вашей операционной системы**. В Windows 7 переменная среды path настраивается через Панель управления - Система и безопасность - Система - Дополнительные параметры системы - Переменные среды.
Далее выбираем переменную Path и добавляем

```
1 C:\python27;C:\django\django\bin
```

Это в том случае, если у вас Python установлен по адресу *C:\Python27* и Django установлен в папке *C:\django*



На этом этап подготовки заканчивается, и мы приступаем к написанию кода.

Этап 2. Написание кода моделей

Создадим проект *firstsite*, а в нем – приложение *polls*. Полный перечень операций в командной строке приведен на *рисунке.*

2.1. Откройте cmd.exe (Командная строка для Windows). Перейдите в директорию, где будете создавать свой проект:

cd *C:\projects*

2.2. Создайте проект *firstsite* в этой директории:

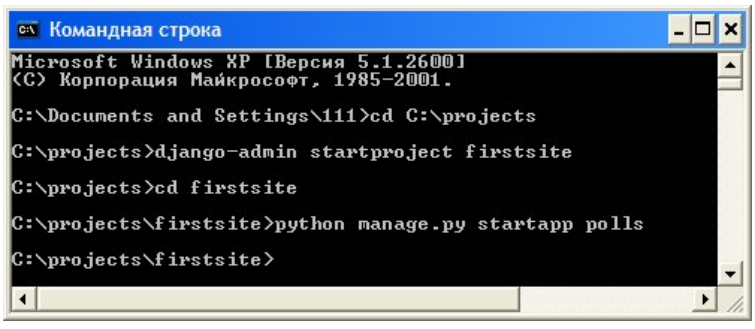
django-admin.py startproject *firstsite*

2.3. Перейдите в директорию созданного проекта:

cd *firstsite*

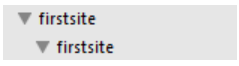
2.4. Создайте там приложение *polls*:

python manage.py **startapp** polls



Создание приложения polls в командной строке

Если все было сделано верно, то вы увидите в папке firstsite файлы вашего проекта:



Структура файлов проекта firstsite

Теперь приступим к созданию моделей в файле *polls/models.py*:

```
# -*- coding:utf-8 -*-

import datetime
from django.db import models


class Question(models.Model):
    """Вопрос"""
    title = models.CharField(max_length=200, verbose_name = "Вопрос")
    date_published = models.DateTimeField(verbose_name = "Дата публикации",
        default = datetime.datetime.now())
    is_active = models.BooleanField(verbose_name = "Опубликован")

    def __unicode__(self):
        return self.title

    class Meta:
        verbose_name = 'Вопрос'
        verbose_name_plural = 'Вопросы'


class Answer(models.Model):
    """Вариант ответа на вопрос"""
    question_id = models.ForeignKey(Question)
    answer = models.CharField(max_length=200, verbose_name = "Ответ")
    votes = models.IntegerField(verbose_name = "Голосов", default = 0)

    def __unicode__(self):
        return self.answer

    class Meta:
        verbose_name = 'Ответ'
        verbose_name_plural = 'Ответы'
```

Примечание: По умолчанию текст файлов Django имеет кодировку ANSI. Чтобы избежать проблем с русскими символами, рекомендуется сменить кодировку используемых файлов на UTF-8. В Notepad++ это можно сделать командой "Кодировки" - "Преобразовать в UTF-8 (без BOM)".

Классы названы в единственном числе, т.к. они характеризуют сущность, на основе которой будут создаваться объекты: *Question* – вопрос, *Answer* – ответ. Это общепринятый стандарт, которому нужно следовать.

verbose_name = "Вопрос" – так будет называться поле question в админке.
max_length=200 – предельно допустимое число символов в поле.
default – позволяет задавать значение поля по умолчанию

Один вопрос может иметь много вариантов ответа, следовательно перед нами связь "один-ко-многим". Эта связь в Django реализуется выражением:
question_id = models.ForeignKey(Question)
Эта строка означает, что в создаваемой таблице поле *question_id* будет внешним ключом таблицы *Question*.

Если вы используете версию Pytnon 2.x, то для *каждого класса* ***обязательно*** использовать функцию __unicode__(self), которая позволяет вернуть объект в формате unicode. Лучше будет, если эта функция будет возвращать один-два элемента класса, например в классе Answer:

```
def __unicode__(self):
    return self.answer
```

Здесь функция возвращает только формулировку ответа, а не число голосов и т.д.

Также обратите внимание на класс *Meta*, включенный в каждый класс модели. Из названия понятно, что этот класс содержит мета-данные о классе (описательного характера). В нашем случае это русское название модели в административном интерфейсе - для единственного и множественного числа.

Код моделей написан, можно генерировать на его основе таблицы. Однако сначала произведем настройку нашего проекта *firstsite*.

Продолжение - Опрос (ч.2)

Автор: yesnik на 05:11 0 Comments

Ярлыки: admin

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

суббота, 25 февраля 2012 г.

Опрос (ч.2)

В части 1 мы создали файл с моделями нашего приложения *polls/models.py*. В этой части мы настроим наш проект и создадим таблицы в базе данных.

Этап 3. Настройка проекта

3.1. По умолчанию в файле настроек проекта *firstsite/firstsite/settings.py* приведены настройки подключения к базе данных sqlite. Оставим их.
3.2. Подключим приложение *polls* к проекту:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Приложение "Опросы"
    'polls',
)
# ...

# Указываем русский язык для админки
LANGUAGE_CODE = 'ru-RU'
```

Важно: Файл settings.py, как и другие файлы проекта рекомендуется сохранять в кодировке utf-8 (без BOM) и предварять каждый файл явным указанием кодировки:

```
# -*- coding:utf-8 -*-
```

В противном случае при использовании в файле русских букв Django покажет ошибку:

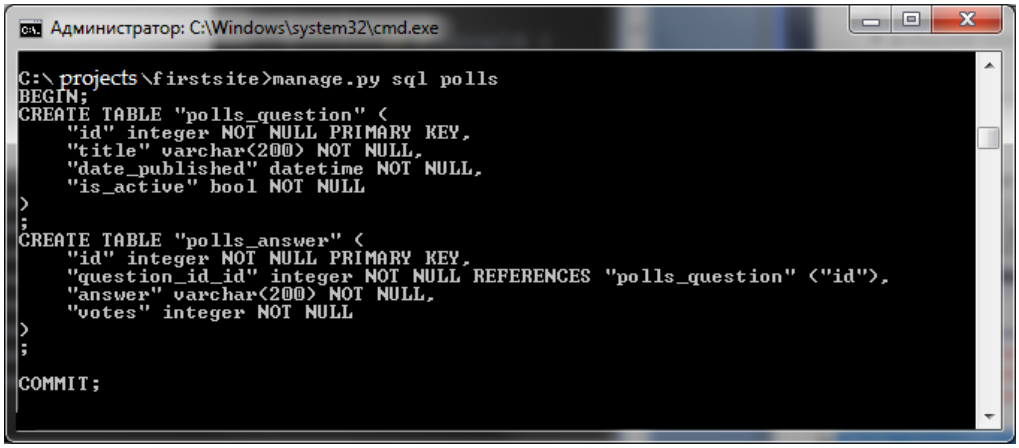
```
File "C:\sites\firstsite\firstsite\settings.py", line 40
SyntaxError: Non-ASCII character '\xd0' in file C:\sites\firstsite\firstsite\settings.py on line
40, but no encoding declared; see http://www.python.org/peps/pep-0263.html for details
```

Этап 4. Генерация таблиц в БД

4.1. Проверим правильность настройки. Откройте командную строку (cmd.exe), перейдите в папку проекта *firstsite*. Выполните команду:

```
manage.py sql polls
```

При правильной настройке вы получите список команд, которые Django выполнит при создании таблиц в БД для приложения *polls*:



Список команд при создании БД

Архив блога
► 2014 (13)
► 2013 (38)
▼ 2012 (6)
▼ Февраль (6)
Опрос (ч.4)
Опрос (ч.3.2)
Опрос (ч.3.1)
Опрос (ч.2)
Опрос (ч.1)
Пролог

Обо мне

yesnik
Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.
Просмотреть профиль

Обратите внимание:

- названия таблиц генерируются так: имяПриложения_имяМодели
- id каждой таблицы генерируется автоматически
- по умолчанию поля NOT NULL
- к названию поля внешнего ключа добавляется _id
- внешний ключ создается выражением NULL REFERENCES "polls_question" ("id")

4.2. Проверим модели на наличие ошибок. Выполните в командной строке:

```
manage.py validate
```

4.3. Сгенерируем таблицы в БД:

```
manage.py syncdb
```

При генерации таблиц вам предложат создать *superuser*. Согласитесь, набрав *yes*. Введите логин и пароль – их вы будете использовать для входа в админку.

Продолжение: Опрос (ч.3.1)

Автор: yesnik на 05:25 0 Comments

Ярлыки: admin

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

суббота, 25 февраля 2012 г.

Опрос (ч.3.1)

В части 2 мы сгенерировали таблицы для нашего приложения в базе данных. В этой части мы настроим админку для использования нашего приложения "Опросы".

Этап 5. Настройка админки

1. Зарегистрируем модель *Question* в админке. Для этого отредактируем файл *polls/admin.py*:

```
1 # -*- coding:utf-8 -*-
2 from django.contrib import admin
3 from .models import Question
4
5 admin.site.register(Question)
```

Здесь мы импортируем только модель *Question*. Модель Answer импортировать не нужно, т.к. она связана с Question через внешний ключ.

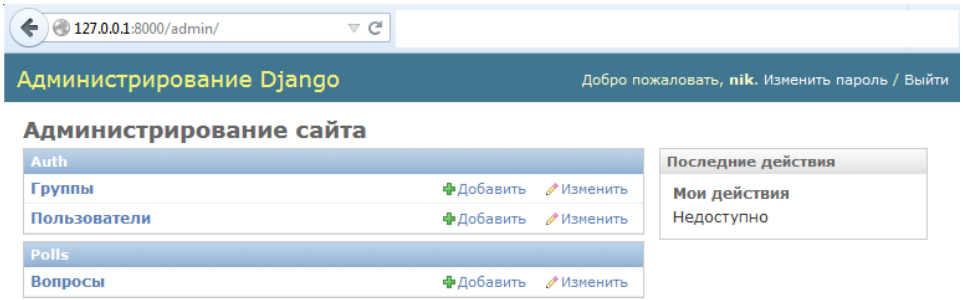
Внимание: при импорте модели *Question* мы не указываем имя приложения *polls*:
`from .models import Question`
Это позволяет нам не привязываться к названию приложения. Мы просто импортируем модель *Question* из файла *models.py*, который лежит в той же папке, что и файл *admin.py*.

2. Запустим сервер разработки, выполнив в консоли *cmd.exe* команду:

```
python manage.py runserver
```

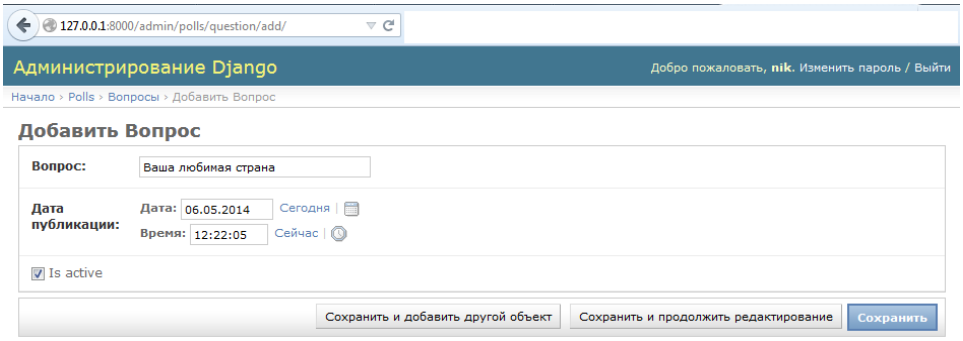
По умолчанию, сервер будет запущен по адресу: *http://127.0.0.1:8000/*

3. Проверьте работу админки: http://127.0.0.1:8000/admin/
У вас получится следующее:



Внешний вид аминки

Нажмите на пункт *Вопросы*, на открывшейся странице в правом верхнем углу нажмите *Добавить вопрос* и создайте вопрос для голосования, нажав *Сохранить*:



Создание вопроса для голосования

Архив блога
► 2014 (13)
► 2013 (38)
▼ 2012 (6)
▼ Февраль (6)
Опрос (ч.4)
Опрос (ч.3.2)
Опрос (ч.3.1)
Опрос (ч.2)
Опрос (ч.1)
Пролог

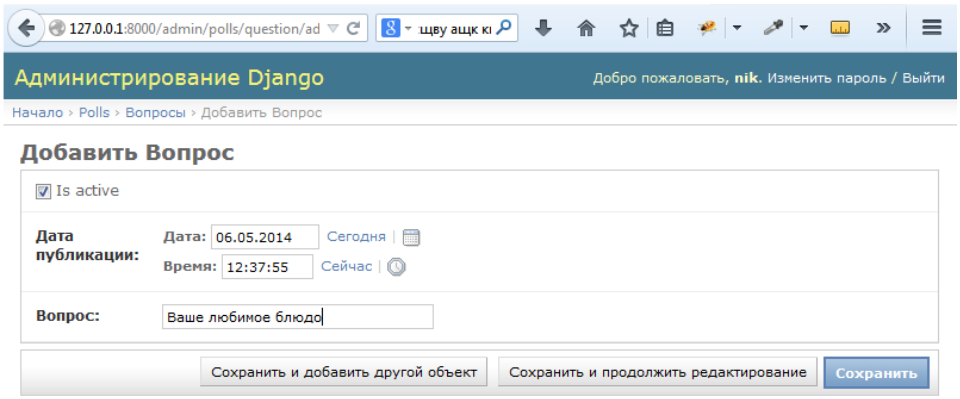
Обо мне

yesnik
Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.
Просмотреть профиль

4. Поменяем порядок полей в админке Django. Для этого скорректируем файл *polls/admin.py*:

```
# -*- coding:utf-8 -*-
from django.contrib import admin
from .models import Question

class QuestionAdmin(admin.ModelAdmin):
    fields = ['is_active', 'date_published', 'title']

admin.site.register(Question, QuestionAdmin)
```

Смысл состоит в создании класса с настройками, который передается в качестве второго параметра в функцию admin.site.register().
Поля поменялись местами:

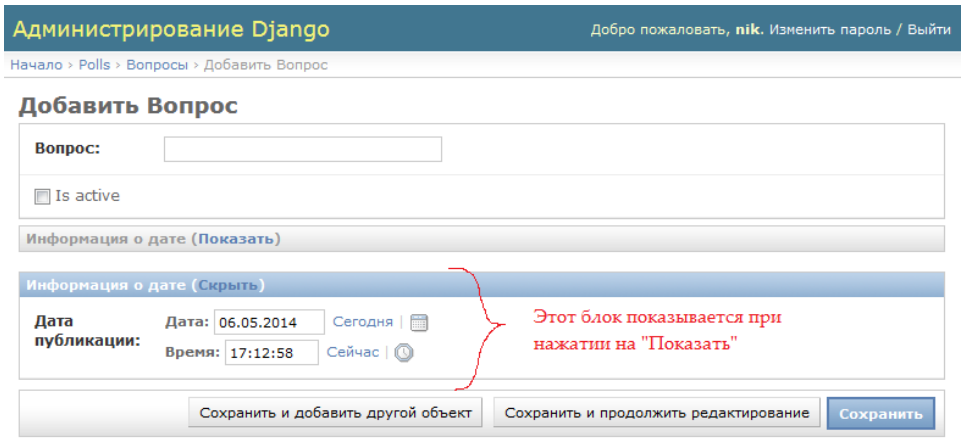


Поля поменялись местами

5. Добавим **филдсеты** (fieldsets) – группы полей. Для групп полей, которые редко используются, можно предусмотреть, чтобы они по умолчанию были в свернутом состоянии - при помощи 'classes': ['collapse']. Отредактируем файл *polls/admin.py*:

```
# -*- coding:utf-8 -*-
from django.contrib import admin
from .models import Question


class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None,
            {'fields': ['title', 'is_active']}
        ),
        ('Информация о дате',
            {'fields': ['date_published'],
            'classes': ['collapse']}
        ),
    ]

admin.site.register(Question, QuestionAdmin)
```

Указание в качестве свойства класса collapse позволяет отображать филдсет свернутым:



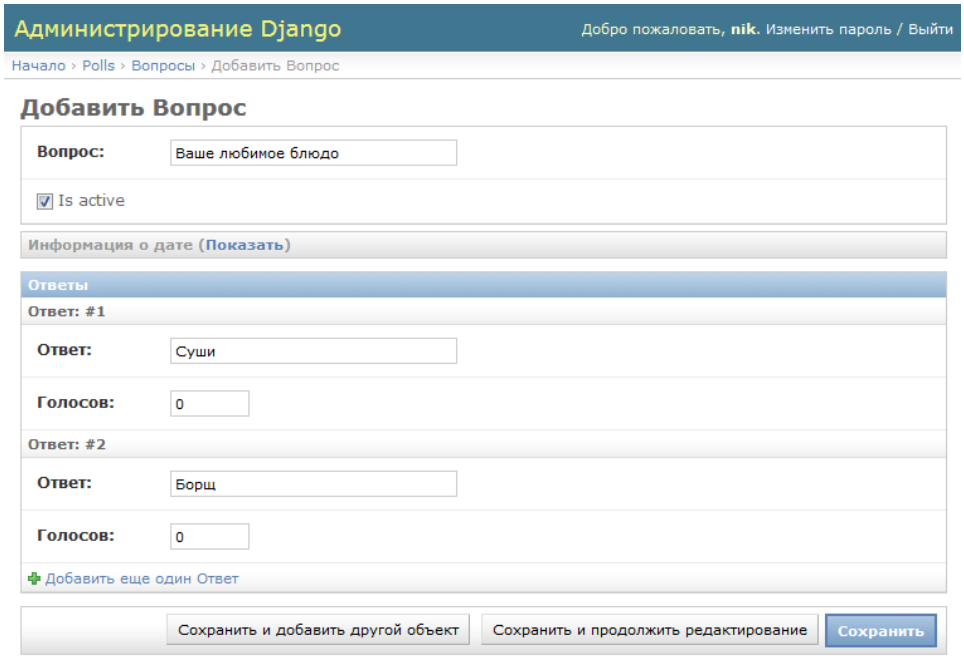
Филдсет свернут

6. Будет удобно, если варианты ответа можно будет добавить на той же странице, что и вопрос опроса. Благо, что фреймворк Django легко позволяет это реализовать. Отредактируем файл *polls/admin.py*:

```
# -*- coding:utf-8 -*-
from django.contrib import admin
from .models import Question, Answer


class AnswerInline(admin.StackedInline):
    model = Answer
```

```
    extra = 2


class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None,
            {'fields': ['title', 'is_active']}
        ),
        ('Информация о дате',
            {'fields': ['date_published'],
            'classes': ['collapse']}
        ),
    ]
    inlines = [AnswerInline]


admin.site.register(Question, QuestionAdmin)
```

Теперь мы можем редактировать ответы на странице создания вопроса для голосования. В классе настроек *AnswerInline* (имя может быть любым) указана используемая модель *Answer* и число ответов по умолчанию *extra* = 2.

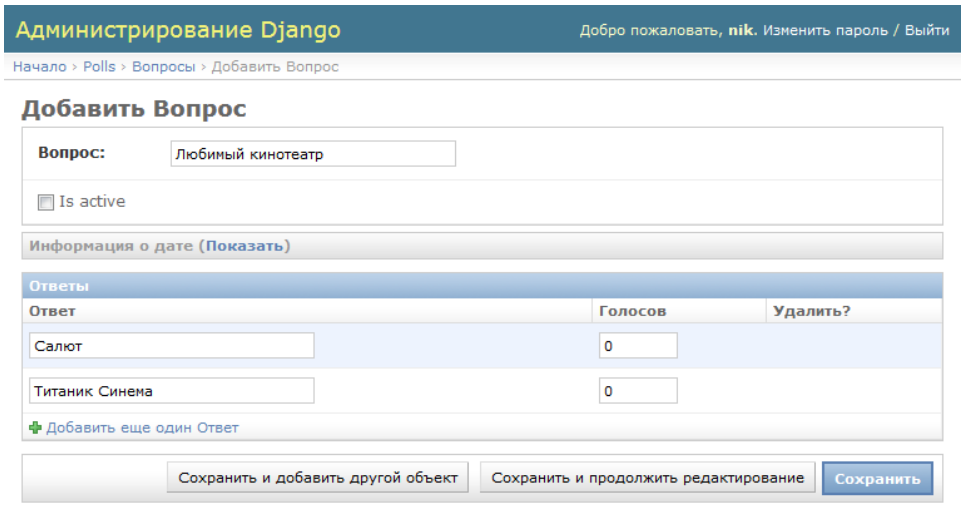


Ответы на странице создания вопроса

Однако удобнее, если ответы и голоса располагаются в одну линию. Для этого нужно поправить файл *polls/admin.py*:

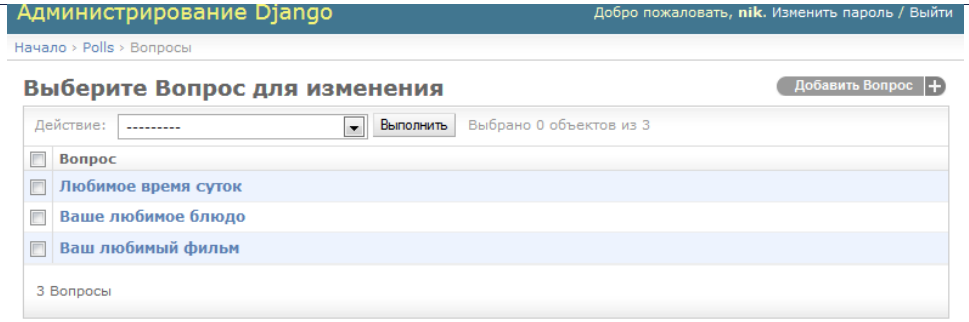
```
class AnswerInline(admin.TabularInline):
    #...
```

Как видно, мы заменили admin.StackedInline на admin.TabularInline. Вот результат:



Расположение ответов TabularInline

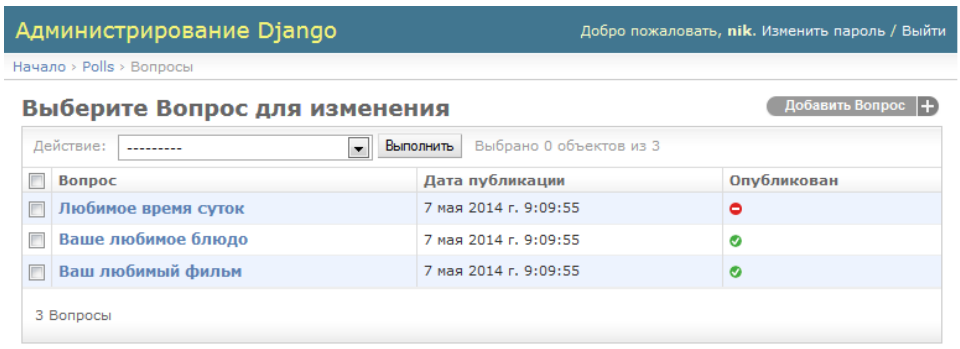
7. По умолчанию Django отображает для каждой модели то, что возвращает функция *__unicode__(self)*, расположенная в каждом классе модели. Так, например, в модели Question эта функция возвращает *self.title*. Поэтому по умолчанию отображается только поле *title (в модели Question мы присвоили полю название "Вопрос")*:



Поле Вопрос

Пусть еще отображается дата публикации опроса и его активность (неактивный опрос на сайте отображать не будем). Для этого нужно поправить файл *polls/admin.py*:

```
class QuestionAdmin(admin.ModelAdmin):
    # ...
    list_display = ('title', 'date_published', 'is_active')
```



Поля Вопрос, Дата публикации, Опубликован

Продолжение - Опрос (ч.3.2)

Автор: yesnik на 05:44 0 Comments

Ярлыки: admin

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Технологии Blogger.

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

суббота, 25 февраля 2012 г.

Опрос (ч.3.2)

В части 3.1 мы исследовали возможности админки Django. В этой части мы продолжим наши исследования.

1. Опросов может быть много, поэтому мы хотим отслеживать популярные из них. В админке Django нужно видеть, в каких опросах проголосовало более 100 человек. Таким образом, в админке наряду с полями *Вопрос*, *Дата публикации*, *Опубликован* должно появиться поле *Популярный*.
Обратите внимание, что количество голосов, при котором пост становится популярным, может измениться в будущем. Следовательно, этот параметр разумно вынести в настройки. Поскольку эта настройка специфична для приложения *polls*, то разместим ее в нем. Создадим файл *polls/settings.py*:

```
# -*- coding: utf-8 -*-

# Кол-во голосов, при котором пост в админке становится Популярным
POLLS_POPULAR_VOTES_LIMIT = 100
```

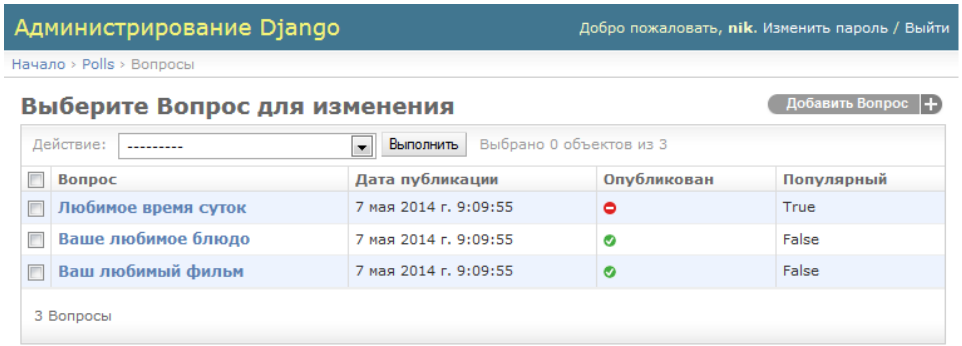
2. Используем эти настройки в файле моделей нашего приложения *polls/models.py*:

```
# ...
# Импортируем настройки приложения polls
import settings

class Question(models.Model):
    # ...
    # Этот метод позволяет выявить Популярный опрос для показа в админке
    def is_popular(self):
        answers = Answer.objects.filter(question_id=self.id)
        votes_total = sum([answer.votes for answer in answers])
        return votes_total > settings.POLLS_POPULAR_VOTES_LIMIT
    # Описание столбца в админке
    is_popular.short_description = 'Популярный'
```

3. Чтобы колонка *Популярный* появилась в админке, нужно отредактировать файл *polls/admin.py*:

```
class QuestionAdmin(admin.ModelAdmin):
    # ...
    # Добавим название метода модели is_popular()
    list_display = ('title', 'date_published', 'is_active', 'is_popular')
```

В результате в админке появится новая колонка *Популярный*:



4. Согласитесь, названия True, False не очень презентабельны. Давайте их заменим на изображения, подобные тем, что находится в колонке Опубликован. Для этого метод is_popular() должен возвращать html изображения. При этом картинок у нас будет две, путь к ним мы укажем в настройках приложения - *polls/settings.py*:

```
# ...

IMG_TRUE_PATH = '/static/admin/img/icon-yes.gif'
IMG_FALSE_PATH = '/static/admin/img/icon-no.gif'
```

Архив блога
► 2014 (13)
► 2013 (38)
▼ 2012 (6)
▼ Февраль (6)
Опрос (ч.4)
Опрос (ч.3.2)
Опрос (ч.3.1)
Опрос (ч.2)
Опрос (ч.1)
Пролог

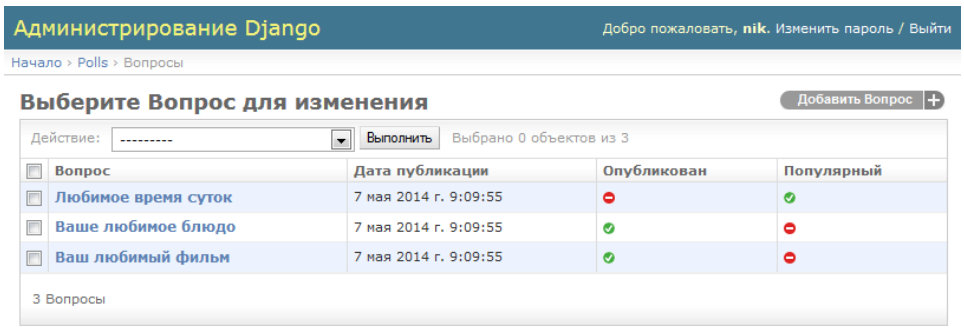
Обо мне

yesnik
Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.
Просмотреть профиль

5. Чтобы метод *is_popular()* возвращал html, отредактируем файл *polls/model.py*:

```
# ...

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

    def is_popular(self):
        answers = Answer.objects.filter(question_id=self.id)
        votes_total = sum([answer.votes for answer in answers])
        if votes_total > settings.POLLS_POPULAR_VOTES_LIMIT:
            img_path = settings.IMG_TRUE_PATH
        else:
            img_path = settings.IMG_FALSE_PATH
        return '<img alt="" src="{}" />'.format(img_path)

    is_popular.short_description = 'Популярный'
    # Важно указать эту настройку, чтобы django не экранировал тэги
    is_popular.allow_tags = True
```

Обратите внимание, мы укзали настройку is_popular.**allow_tags** = True. Она позволяет нам избежать экранирования и вывести html теги прямо из модели:



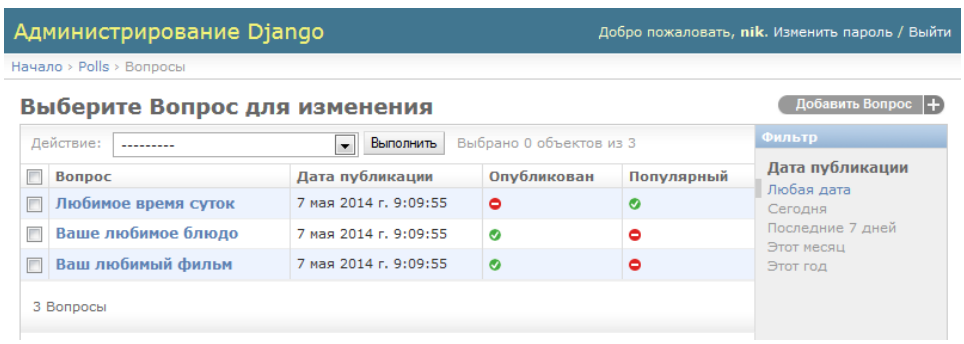
6. Искушенный любитель Django (кто постоянно читает документацию) заметит, что шаги 4, 5 можно было заменить одной строчкой кода. Оказывается, для метода есть настройка, которая позволяет True/False заменять на изображения автоматически. Для этого нужно в *polls/models.py* добавить сточку:

```
class Question(models.Model):
    # ...

    def is_popular(self):
        answers = Answer.objects.filter(question_id=self.id)
        votes_total = sum([answer.votes for answer in answers])
        return votes_total > settings.POLLS_POPULAR_VOTES_LIMIT

    is_popular.short_description = 'Популярный'
    # Вот настройка, заменяющая False/True на иконки в админке
    is_popular.boolean = True
```

7. Пусть нам нужно в админке добавить фильтр по дате. Добавьте строчку в *polls/admin.py*:

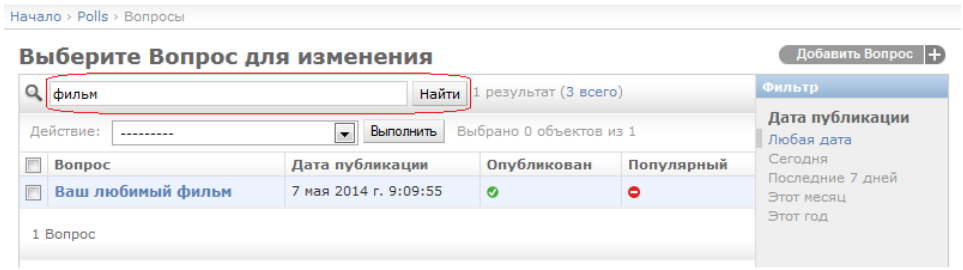
```
# ...
class QuestionAdmin(admin.ModelAdmin):
    # ...
    list_filter = ['date_published']
```

Тогда в админке справа появится фильтр по дате публикации:



8. Когда опросов много, удобно если есть возможность поиска. Добавим форму поиска по формулировкам вопросов. Добавьте строчку в *polls/admin.py*:

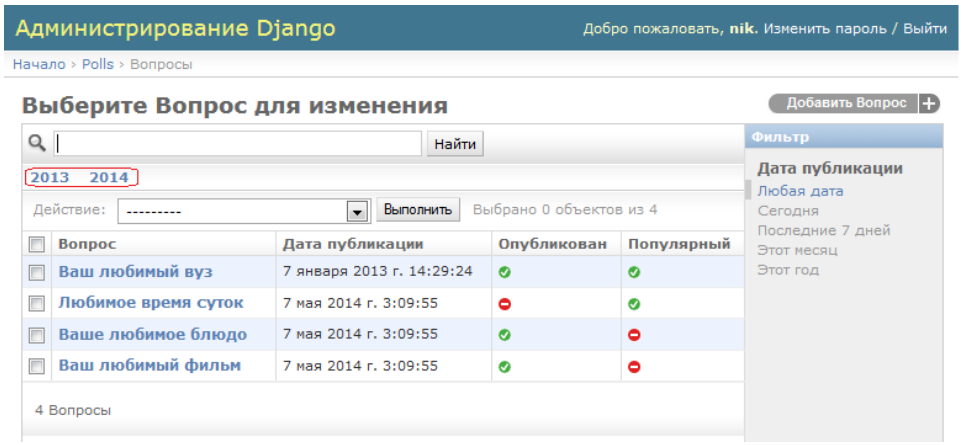
```
# ...
class QuestionAdmin(admin.ModelAdmin):
    # ...
    search_fields = ['title']
```



9. Добавим в админку иерархию по дате: год, месяц… Добавьте строчку в *polls/admin.py*:

```
# ...
class QuestionAdmin(admin.ModelAdmin):
    # ...
    date_hierarchy = ['date_published']
```

При использовании этой настройки вы можете столкнуться с ошибкой:
`ImproperlyConfigured at /admin/polls/question/`
`This query requires pytz, but it isn't installed.`
Для устранения этой ошибки нужно установить питоновский пакет pytz (PYthon TimeZone). Если пакет установлен, вы получите такой результат:



Продолжение - Опрос (ч.4)

Автор: yesnik на 05:56 0 Comments

Ярлыки: admin

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

суббота, 25 февраля 2012 г.

Опрос (ч.4)

В части 3.2 мы ознакомились с основными возможностями, которые нам дает админка Django. В этой части мы отобразим список последних двух свежих опросов, который будет доступен пользователям сайта.

1. Отредактируем файл роутов нашего проекта - *firstsite/urls.py*:

```
# -*- coding: utf-8 -*-
from django.conf.urls import patterns, include, url

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # Examples:
    # url(r'^$', 'firstsite.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

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

    # Приложение Опросы
    url(r'^polls/', include('polls.urls', namespace="polls") ),
)
```

Здесь мы делаем подключение роутов приложения polls, если пользователь переходит по адресу http://127.0.0.1:8000/polls .
Параметр **namespace**="polls" мы используем, чтобы впоследствии использовать неймспейсы в шаблонах.

2. Создадим файл роутов нашего приложения: *polls/urls.py*:

```
# -*- coding: utf-8 -*-
from django.conf.urls import patterns, include, url
from .views import *

urlpatterns = patterns('',

    url(r'^$', IndexView.as_view(), name='index'),
)
```

Данный роут означает, что если пользователь перейдет по адресу http://127.0.0.1:8000/polls/ , то будет задействовано представление **IndexView**. Это так называемое *class based view* представление, т.е. представление, основанное на классах.

3. Создадим представление *IndexView* главной страницы в *polls/views.py*:

```
# -*- coding: utf-8 -*-
from django.shortcuts import render
from django.views.generic import ListView, DetailView
from .models import Question, Answer

class IndexView(ListView):
    template_name = 'polls/index.html'
    context_object_name = 'latest_questions_list'

    def get_queryset(self):
        """Вернуть 2 последних свежих опроса"""
        return Question.objects.order_by('-date_published')[:2]
```

Обратите внимание:
template_name = 'polls/index.html' - мы не стали писать просто 'index.html', т.к. в других приложениях тоже может быть шаблон с таким именем. Можно считать что '*polls*/...' здесь своего рода неймспейс.
context_object_name = 'latest_questions_list' - именно такое название в шаблоне будет иметь переменная с данными.

4. Шаблон, используемый представлением IndexView относится к приложению, а не проекту. Поэтому разумно поместить этот шаблон в приложение. Создадим шаблон по адресу *polls/templates/polls/index.html*:

```
<h1>Последние опросы</h1>
```

Архив блога
► 2014 (13)
► 2013 (38)
▼ 2012 (6)
▼ Февраль (6)
Опрос (ч.4)
Опрос (ч.3.2)
Опрос (ч.3.1)
Опрос (ч.2)
Опрос (ч.1)
Пролог

Обо мне

yesnik
Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.
Просмотреть профиль

```
{% if latest_questions_list %}
    <ul>
        {% for question in latest_questions_list %}
            <li>{{ question.title }}</li>
        {% endfor %}
    </ul>
{% else %}
    <p>Опросы не найдены.</p>
{% endif %}
```

Перейдите по адресу, чтобы посмотреть результат: http://127.0.0.1:8000/polls/

Продолжение - Опрос (ч.5)

Автор: yesnik на 06:04 0 Comments

Ярлыки: admin

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

среда, 7 мая 2014 г.

Опрос (ч.5)

В части 4 мы создали роут, по которому стали доступен список последних двух опросов. В этой части мы сделаем детальное отображение опроса.

1. Добавим роут в файл *polls/urls.py*:

```
# ...
urlpatterns = patterns('',

    # ...
    url(r'^(?P<pk>\d+)/$', PollDetailView.as_view(), name='detail'),
)
</pk>
```

Это значит, что представление PollDetailView будет вызываться, когда пользователь переходит, к примеру, по адресу: http://127.0.0.1:8000/polls/1

2. Создадим представление в файле *polls/views.py*:

```
# -*- coding: utf-8 -*-
from django.shortcuts import render
from django.views.generic import ListView, DetailView
from .models import Question, Answer

# ...

class QuestionDetailView(DetailView):
    model = Question
    template_name = 'polls/detail.html'
```

Здесь мы создали класс представления, который расширяет встроенный в django класс *DetailView*, содержащий нужный функционал для отображения детальной страницы.

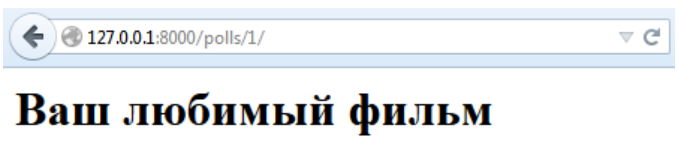
3. Детальное представление опроса должно быть формой с радиокнопками, чтобы ответ пользователя можно было отправить на определенный url, при обращению к которому будет вызвано представление, где данные запишутся в БД. Создадим такой шаблон по адресу *polls/templates/polls/detail.html*:

```
<h1>{{ question.title }}</h1>

{% if error_message %}
    <p><strong>{{ error_message }}</strong></p>
{% endif %}

<form action="" method="post">
    {% csrf_token %}
    {% for answer in question.answer_set.all %}
        <input type="radio" name="answer" id="answer{{ forloop.counter }}" value="{{ answer
        <label for="answer{{ forloop.counter }}">{{ answer.answer }}</label>

    {% endfor %}
    <input type="submit" value="Голосовать" />
</form>
```

Обратите внимание на выражение: `question.answer_set.all`

Вот таким образом в шаблоне можно получить список всех элементов связанной таблицы. В нашем случае у каждого Вопроса есть несколько Ответов. При помощи приведенного выражения мы можем по экземпляру вопроса получить список соответствующих ему ответов.

Перейдем по адресу http://127.0.0.1:8000/polls/1/ и увидим результат:

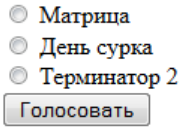


Обо мне

yesnik

Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.

Просмотреть профиль



4. Однако приведенная форма пока не работает.Чтобы она заработала, нужно добавить роут и представление для обработки POST-данных отправляемой формы. Создадим новый роут в файле *polls/urls.py*:

```
urlpatterns = patterns('',
    # ...
    url(r'^(?P<poll_id>\d+)/vote/$', vote, name='vote'),
)
</poll_id>
```

5. Создадим представление, но уже не в виде класса, а в виде функции. Отредактируем файл *polls/views.py*:

```
from django.shortcuts import render, get_object_or_404
from django.http import HttpResponseRedirect
from django.core.urlresolvers import reverse
from django.views.generic import ListView, DetailView
from .models import Question, Answer

# ...

def vote(request, poll_id):

    question = get_object_or_404(Question, pk=poll_id)

    if request.POST.get('answer'):
        selected_answer = question.answer_set.get(pk=request.POST['answer'])
        selected_answer.votes += 1
        selected_answer.save()
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
    else:
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "Вы не выбрали ответ.",
        })
```

Здесь мы проверяем наличие переменной *answer* в POST-запросе. Если она есть, мы по коду ответа достаем запись из базы данных и увеличиваем счетчик *votes* на 1. После этого делаем редирект на страницу результатов (пока еще не создали).

6. Добавим новый роут страницы результатов опроса в *polls/urls.py*:

```
urlpatterns = patterns('',
    # ...
    url(r'^(?P<pk>\d+)/results/$', ResultsView.as_view(), name='results'),
)
</pk>
```

7. Для созданного роута создадим представление в *polls/views.py*:

```
# ...
class ResultsView(DetailView):
    model = Question
    template_name = 'polls/results.html'
```

8. Для созданного представления создадим шаблон *polls/templates/polls/results.html*:

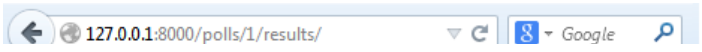
```
<h1>{{ question.title }}</h1>

<ul>
{% for answer in question.answer_set.all|dictsortreversed:"votes" %}
    <li>{{ answer.answer }} - голосов: {{ answer.votes }}</li>
{% endfor %}
</ul>

<a href="{% url 'polls:detail' question.id %}">Голосовать еще раз</a>
```

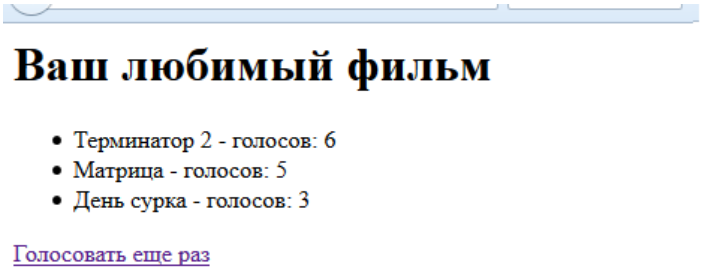
Здесь мы при помощи фильтра Django `dictsortreversed:"votes"` сортируем ответы в порядке убывания числа голосов.

Выражение `{% url 'polls:detail' question.id %}` позволяет сформировать ссылку на детальную страницу просмотра опроса, где находится форма голосования. Здесь `polls` - неймспейс, указанный при подключении роутов приложения *polls* в главном файле роутов проекта *firstsite/urls.py*:

```
urlpatterns = patterns('',
    # ...
    # Задаем неймспейс "polls" для импортируемых роутов приложения
    url(r'^polls/', include('polls.urls', namespace="polls")),
)
```

9. Проверим работу, перейдя в опрос: http://127.0.0.1:8000/polls/1/
Там мы увидим форму голосования. После выбора варианта ответа и нажатия "Голосовать" мы будем перенаправлены на страницу результатов опроса:





Обратите внимание: ответы отсортированы в порядке убывания голосов.

Продолжение - Опрос (ч.6)

Автор: yesnik на 18:51 0 Comments

Ярлыки: django

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Технологии Blogger.

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

понедельник, 12 мая 2014 г.

Опрос (ч.6)

В части 5 нашего Django урока по созданию опроса мы заставили работать нашу форму голосования. В админке можно создавать опросы, и они будут отображаться. В этой части мы обсудим вопросы безопасности созданного приложения.

1. Попробуйте перейти на детальную страницу опроса, снятого с публикации. Система спокойно позволяет вам это. Давайте сделаем так, что при попытке обращения пользователя по url к опросу, снятого с публикации, он видел сообщение: "Извините, опрос снят с публикации".
Для этого отредактируем файл шаблона: *polls/templates/polls/detail.html*:

```
{% if question.is_active %}

    <h1>{{ question.title }}</h1>
    {% if error_message %}
        <p><strong>{{ error_message }}</strong></p>
    {% endif %}

    <form action="{% url 'polls:vote' question.id %}" method="post">
        {% csrf_token %}
        {% for answer in question.answer_set.all %}
            <input type="radio" name="answer" id="answer{{ forloop.counter }}" value="{{ ar
            <label for="answer{{ forloop.counter }}">{{ answer.answer }}</label>

        {% endfor %}
        <input type="submit" value="Голосовать" />
    </form>

{% else %}
    <p>Извините, опрос снят с публикации.</p>
{% endif %}
```

2. На странице просмотра опроса пользователь может указать id несуществующего ответа, что породит ошибку DoesNotExist at /polls/1/vote/ Answer matching query does not exist.
Чтобы ее избежать, отредактируем *polls/views.py*:

```
def vote(request, poll_id):

    question = get_object_or_404(Question, pk=poll_id)

    if request.POST.get('answer'):
        try:
            selected_answer = question.answer_set.get(pk=request.POST['answer'])
        except(Answer.DoesNotExist):
            return render(request, 'polls/detail.html', {
                'question': question,
                'error_message': "Указан недопустимый ответ",
            })
        selected_answer.votes += 1
        selected_answer.save()
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
    else:
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "Вы не выбрали ответ.",
        })
```

Теперь, если хакер в форме укажет id ответа другого опроса и отправит форму, наше приложение не увеличит счетчик голосов в другом опросе и вернет ошибку: *"Указан недопустимый ответ"*.

3. Если хакер в форме голосования вместо числового *id* ответа укажет русские символы, например:

```
<input type="radio" value="ЭТО_ХАКЕР" id="answer1" name="answer">
```

то это вызовет ошибку:
UnicodeEncodeError at /polls/2/vote/ 'decimal' codec can't encode characters in position 0-24: invalid decimal Unicode string

Если пользователь укажет вместо числового id знак минуса "-", то это породит ошибку:
ValueError at /polls/2/vote/ invalid literal for int() with base 10: '-'

Обо мне

yesnik

Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.

Просмотреть профиль

Теперь, зная типы возникающих ошибок, добавим их в блок *except* в файле *polls/views.py*:

```
def vote(request, poll_id):

    question = get_object_or_404(Question, pk=poll_id)

    if request.POST.get('answer'):
        try:
            selected_answer = question.answer_set.get(pk=request.POST['answer'])
        except(Answer.DoesNotExist, UnicodeEncodeError, ValueError):
            # ...
```

Таким образом, мы подавили ошибки при передаче некорректных данных.

Примечание: По идее, мы не должны заботиться о хороших сообщениях для хакеров, усложняя тем самым логику нашего приложения. Для хакера сойдет сообщение **Server Error (500)**, которое будет если в *firstsite/settings.py* установить:

```
DEBUG = False

TEMPLATE_DEBUG = False

ALLOWED_HOSTS = ['127.0.0.1']
```

4. Тем не менее мы все равно имеем шанс проголосовать, к примеру за опрос id = 2, находясь на странице опроса с id = 1. Для этого нам достаточно изменить у формы атрибут action и изменить id ответа. Пример:

```
<form method="post" action="/polls/3/vote/">
    <input type="radio" value="7" id="answer1" name="answer">
    <label for="answer1">Картошка</label><br>
    <!-- Другие поля формы... -->
</form>
```

Да, пусть хакер это делает, т. к. это равносильно тому, что он просто откроет открытый опубликованный опрос и там проголосует. Но нехорошо будет если хакер сможет таким образом увеличивать число голосов по уже закрытым опросам. Чтобы этого не произошло, отредактируем *polls/views.py*:

```
def vote(request, poll_id):

    question = get_object_or_404(Question, pk=poll_id)
    if not question.is_active:
        return HttpResponse('Опрос снят с публикации')
    # ...
```

Таким образом, если кто-то попробует увеличить счетчик снятого с публикации опроса, он получит сообщение *"Опрос снят с публикации"*.

В следующей части урока по Django по созданию опроса мы защитим наши опросы от повторного голосования.

Продолжение - Опрос (ч.7)

Автор: yesnik на 12:35 0 Comments

Ярлыки: django

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)

Уроки по Django

Уроки по python-фреймворку Django. В уроках по шагам раскрываются важные аспекты данного фреймворка, освещаются ошибки и сложности, с которыми может столкнуться Django разработчик.

понедельник, 12 мая 2014 г.

Опрос (ч.7)

В предыдущей части урока по Django мы рассмотрели, как можно защитить наше приложение опросов от хакера. В этом уроке мы защитим наши опросы от повторного голосования.

1. Чтобы пользователь не мог повторно отправить форму, нужно его идентифицировать и где-то в системе записать, в каких опросах он проголосовал.
Идентификацию пользователя мы можем делать по ip. Хранить данные о пользователе можно в таблице с двумя полями: ip, question_id. Для создания этой таблицы напишем модель User в файле *polls/models.py*:

```
# ...
class User(models.Model):
    """Пользователь, участвующий в опросе"""
    ip = models.GenericIPAddressField(verbose_name='IP пользователя')
    question = models.ForeignKey(Question, verbose_name='Вопрос голосования')

    def __unicode__(self):
        return self.ip

    def get_user_ip(self, request):
        x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
        if x_forwarded_for:
            ip = x_forwarded_for.split(',')[-1].strip()
        else:
            ip = request.META.get('REMOTE_ADDR')
        return ip

    def voted_already(self):
        """Голосовал ли пользователь с данным ip в опросе"""
        user_list = User.objects.filter(ip=self.ip, question = self.question)
        return len(user_list) > 0

    class Meta:
        verbose_name = 'Пользователь'
        verbose_name_plural = 'Пользователи'
```

Теперь дадим команду на генерацию таблицы этой модели в БД: `manage.py syncdb`

2. Чтобы наша модель отобразилась в админке, зарегистрируем ее в файле *polls/admin.py*:

```
# ...
class UserAdmin(admin.ModelAdmin):
    list_display = ('ip', 'question')

admin.site.register(User, UserAdmin)
```

3. Добавим в представления логику отслеживания пользователя - файл *polls/views.py*:

```
# ...

class PollDetailView(DetailView):
    model = Question
    template_name = 'polls/detail.html'

    def get_context_data(self, **kwargs):
        context = super(PollDetailView, self).get_context_data(**kwargs)
        user = User()
        user.ip = user.get_user_ip(self.request)
        user.question = Question.objects.get(pk=self.kwargs['pk'])
        # Передаем в шаблон переменную
        context['voted_already'] = user.voted_already()
        return context


def vote(request, poll_id):
    """Обработка данных формы опроса"""
    question = get_object_or_404(Question, pk=poll_id)
    if not question.is_active:
        return HttpResponse('Опрос снят с публикации')

    user = User()
    user.ip = user.get_user_ip(request)
    user.question = question
    if user.voted_already():
        return HttpResponse('Вы уже голосовали в этом опросе')
```


Обо мне

yesnik

Веб-программист, занимаюсь разработкой сайтов на PHP, Python, Javascript. Помимо программирования, занимаюсь изучением юзабилити, чтобы сделать создаваемые приложения максимально ценными для конечного пользователя. Вношу свой вклад в развитие веб, делясь опытом на этом сайте, а также на stackoverflow.com.

Просмотреть профиль

```
    if request.POST.get('answer'):
        try:
            selected_answer = question.answer_set.get(pk=request.POST['answer'])
        except(Answer.DoesNotExist, UnicodeEncodeError, ValueError):
            return render(request, 'polls/detail.html', {
                'question': question,
                'error_message': "Указан недопустимый ответ",
            })

        selected_answer.votes += 1
        selected_answer.save()

        user.save()

        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
    else:
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "Вы не выбрали ответ.",
        })
```

4. Отредактируем шаблон детального представления опроса: *polls/templates/polls/detail.html*:

```
{% if question.is_active %}

    {% if voted_already %}
    <p>Вы уже голосовали в этом опросе.</p>
    {% else %}
        <p>Просим вас принять участие в опросе</p>

        <h1>{{ question.title }}</h1>
        {% if error_message %}
            <p><strong>{{ error_message }}</strong></p>
        {% endif %}

        <form action="{% url 'polls:vote' question.id %}" method="post">
            {% csrf_token %}
            {% for answer in question.answer_set.all %}
                <input type="radio" name="answer"
                    id="answer{{ forloop.counter }}" value="{{ answer.id }}" />
                <label for="answer{{ forloop.counter }}">
                    {{ answer.answer }}
                </label>

            {% endfor %}
            <input type="submit" value="Голосовать" />
        </form>
    {% endif %}

{% else %}
    <p>Извините, опрос снят с публикации.</p>
{% endif %}
```

Проверьте теперь работу приложения опросов. Вам не удастся дважды проголосовать за один и тот же опрос. Чтобы проголосовать еще, вам нужно удалить в админке соответствующие записи модели User.

Автор: yesnik на 18:33 0 Comments

Ярлыки: django

Следующее Главная страница Предыдущее

Подписаться на: Комментарии к сообщению (Atom)