

Создаем первое приложение на Django, часть 7

Продолжаем начатое в шестой части. Мы создали протестированное веб приложение для опросов и теперь нам необходимо добавить интерфейс администратора, который мы изучали во второй части.

Настройка форм в админке

После регистрации модели `Question` с помощью `admin.site.register(Question)` Django автоматически построил форму, которая отображает модель. Скорее всего вам понадобится настроить вид и работу этой формы. Это можно сделать, указав различные опции при регистрации модели.

Давайте посмотрим, как это работает, поменяв порядок полей в форме. Замените `admin.site.register(Question)` на:

```
polls/admin.py
from django.contrib import admin

from .models import Question

class QuestionAdmin(admin.ModelAdmin):
    fields = ['pub_date', 'question_text']

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



Если вам необходимо настроить интерфейс администратора для модели, создайте `ModelAdmin`, затем передайте его в `admin.site.register()`.

Пример выше переместит поле “Publication date” перед “Question”:

Home > Polls > Questions > What's up?

Change question

Date published:

Date: 2015-09-06 Today 
Time: 21:16:20 Now 

Question text:

What's up?

Для простой формы это не так важно, но, если форма содержит большое количество полей, важно разместить их в удобном для пользователя порядке.

Также удобно разбить большую форму на несколько наборов полей:

```
polls/admin.py
from django.contrib import admin

from .models import Question

class QuestionAdmin(admin.ModelAdmin):
    fieldsets = [
        (None, {'fields': ['question_text']}),
        ('Date information', {'fields': ['pub_date']}),
    ]

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

Первый элемент кортежа в `fieldsets` – это название набора полей. Вот как выглядит наша форма сейчас:


Home › Polls › Questions › What's up?


Change question

Question text:

Date information

Date published:

Date: Today 

Time: Now 

Добавляем связанные объекты

Теперь у нас есть страница вопросов в админке, но `Question` содержит несколько `Choice`, которые не показываются в админке.

Пока что.

Есть два способа решить эту проблему. Первый – зарегистрировать `Choice` в админке, как мы сделали это для `Question`. Это просто:



```
polls/admin.py
from django.contrib import admin

from .models import Choice, Question
# ...
admin.site.register(Choice)
```

Теперь “Choices” доступны в админке. Форма добавления выглядит следующим образом:

Home › Polls › Choices › Add choice

Add choice

Question:	<div>-----</div> <div> </div>
Choice text:	<div></div>
Votes:	<div>0</div>

В этой форме поле “Question” представлено “select”, который содержит все вопросы в базе данных. Django знает, что поля `ForeignKey` должны быть представлены как `<select>`. В нашем примере присутствует только один вопрос в базе данных.

Также обратите внимание на кнопку “Add Another” возле “Question.” Для каждой связи `ForeignKey` будет показана такая кнопка. После нажатия “Add Another” будет показана форма добавления вопроса в новом окне. Если вы добавите вопрос в этой форме и нажмете “Save”, Django сохранит вопрос и динамически добавит его в поле выбора вопроса.

Но это не эффективный способ добавлять объекты `Choice`. Было бы удобно добавить варианты ответов при создании объекта `Question`. Давайте сделаем это.

Удалите `register()` для модели `Choice`. Теперь поменяйте код для модели `Question`:

```
polls/admin.py
from django.contrib import admin

from .models import Choice, Question

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)
```

Этот код говорит Django: “Объекты `Choice` редактируются на странице `Question`. По умолчанию показывай формы для трех ответов.”

Давайте посмотрим на форму добавления вопроса:

Home > Polls > Questions > Add question

Add question

Question text:

Date information (Hide)

Date published:

Date:

Today



Time:

Now



CHOICES

Choice: #1

Choice text:

Votes:

0

Choice: #2

Choice text:

Votes:

0

Choice: #3

Choice text:

Votes:

0

[+ Add another Choice](#)

Save and add another

Save and continue editing

SAVE

Она работает следующим образом: есть три формы для добавления ответов — как это указано *extra*. И на странице редактирования вопросов каждый раз будет доступно три формы для добавления.

В конце вы можете увидеть кнопку “Add another Choice”. Если вы нажмете на неё, будет добавлена еще одна форма. Если вы хотите удалить форму, нажмите “X” в правом верхнем углу появившейся формы. Обратите внимание, вы не можете удалить начальные три формы. Это изображение показывает, как выглядит добавленная форма:

CHOICES	
Choice: #1	
Choice text:	<input type="text"/>
Votes:	<input type="text" value="0"/>
Choice: #2	
Choice text:	<input type="text"/>
Votes:	<input type="text" value="0"/>
Choice: #3	
Choice text:	<input type="text"/>
Votes:	<input type="text" value="0"/>
Choice: #4 ✕	
Choice text:	<input type="text"/>
Votes:	<input type="text" value="0"/>
+ Add another Choice	

Одна маленькая проблема. Формы добавления `Choice` занимают очень много места на экране. Но Django предоставляет возможность отображать их во вкладках. Поменяйте определение `ChoiceInline`:

```
polls/admin.py
class ChoiceInline(admin.TabularInline):
    #...
```

С `TabularInline` (вместо `StackedInline`) связанные объекты отображаются в более компактном виде:

CHOICES			
CHOICE TEXT		VOTES	DELETE?
<input type="text"/>		<input type="text" value="0"/>	
<input type="text"/>		<input type="text" value="0"/>	
<input type="text"/>		<input type="text" value="0"/>	
+ Add another Choice			
		Save and add another	Save and continue editing
		SAVE	

Обратите внимание на колонку “Delete?”, которая позволяет удалить, как только что добавленные формы, так и существующие объекты.

Настройка списка объектов

Теперь, когда страница добавления и редактирования вопроса выглядит хорошо, мы можем улучшить страницу списка вопросов.

Сейчас она выглядит следующим образом:

Home > Polls > Questions

Select question to change ADD QUESTION +

Action: 0 of 1 selected

<input type="checkbox"/>	QUESTION TEXT
<input type="checkbox"/>	What's up?

1 question

По умолчанию Django показывает результат `str()` для каждого объекта. Но было бы на много удобнее видеть отдельные поля. Для этого используйте настройку `list_display`, которая содержит кортеж полей, которые будут представлены отдельными колонками:

```
polls/admin.py
class QuestionAdmin(admin.ModelAdmin):
    # ...
    list_display = ('question_text', 'pub_date')
```

Также для удобства добавим метод `was_published_recently()`:

```
polls/admin.py
class QuestionAdmin(admin.ModelAdmin):
    # ...
    list_display = ('question_text', 'pub_date', 'was_published_recently')
```

Теперь страница списка вопросов выглядит следующим образом:

Home > Polls > Questions

Select question to change

Action: 0 of 1 selected

<input type="checkbox"/>	QUESTION TEXT	DATE PUBLISHED	WAS PUBLISHED RECENTLY
<input type="checkbox"/>	What's up?	Sept. 3, 2015, 9:16 p.m.	False

1 question

Вы можете нажать на заголовок колонки, чтобы отсортировать записи, кроме колонки `was_published_recently`, т.к. сортировка по методу не поддерживается. Также обратите внимание, что название колонки для `was_published_recently` по умолчанию содержит название метода (где подчеркивание заменено на пробелы), а значения являются результатами выполнения метода.

Вы можете исправить это, указав несколько атрибутов для метода (в файле `polls/models.py`):

```
polls/models.py
class Question(models.Model):
    # ...
    def was_published_recently(self):
        return self.pub_date >= timezone.now() - datetime.timedelta(days=1)
    was_published_recently.admin_order_field = 'pub_date'
    was_published_recently.boolean = True
    was_published_recently.short_description = 'Published recently?'
```

Подробнее об этих атрибутах можно узнать в описании `list_display`.

Давайте снова отредактируем файл `polls/admin.py` и улучшим страницу списка объектов `Question`: добавим фильтрацию, используя: `attr:~django.contrib.admin.ModelAdmin.list_filter`. Добавьте следующие строки в `QuestionAdmin`:

```
list_filter = ['pub_date']
```

Это добавило справа панель фильтрации, которая позволит фильтровать записи по полю `pub_date`:

Home > Polls > Questions

Select question to change ADD QUESTION +

Action: 0 of 1 selected

<input type="checkbox"/>	QUESTION TEXT	DATE PUBLISHED	PUBLISHED RECENTLY?
<input type="checkbox"/>	What's up?	Sept. 3, 2015, 9:16 p.m.	

1 question

FILTER

By date published

[Any date](#)

[Today](#)

[Past 7 days](#)

[This month](#)

[This year](#)

Вид фильтра зависит от типа поля, к которому он применяется. Так как `pub_date` представлено `DateTimeField`, Django отображает подходящие опции для фильтрации: “Any date”, “Today”, “Past 7 days”, “This month”, “This year”.

Теперь добавим возможность поиска по записям:

```
search_fields = ['question_text']
```

Теперь в верхней части страницы появилось поле для поиска. При вводе запроса в это поле Django выполнит поиск по полю `question_text`. Вы можете указать сколько угодно полей, но, учитывая, что используется запрос `LIKE`, их количество должно быть разумным, чтобы запрос не выполнялся слишком долго.

Также страница списка объектов содержит постраничное отображение. По умолчанию показывается 100 записей на страницу.

Изменение постраничного отображения, поиска, фильтров, иерархия по дате и сортировка от лично работают вместе.

Настройка внешнего вида интерфейса администратора

“Django administration” на каждой странице выглядит не очень привлекательно. Это просто текст по умолчанию.

Его легко поменять, используя систему шаблонов Django. Админка использует стандартную систему шаблонов Django.

Настройка шаблонов проекта

Создайте каталог `templates` в каталоге вашего проекта (это тот, который содержит файл `manage.py`). Шаблоны могут находиться где угодно в файловой системе, пока у Django есть доступ к ним. (Django запускается от пользователя, от которого запущен веб-сервер.) Одна, хранить шаблоны в проекте удобно и является хорошей практикой.

Откройте файл настроек (`mysite/settings.py`) и добавьте опцию `DIRS` в настройку `TEMPLATES`:

```
mysite/settings.py
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    ],
]
```

`DIRS` содержит список каталогов с шаблонами.

Организация шаблонов

Как и в случае со статическими файлами, мы *можем* хранить все шаблоны вместе в одном большом каталоге, и это может отлично работать. Однако, шаблоны, которые относятся к определенному приложению, необходимо хранить в каталоге шаблонов приложения (например `polls/templates`), а не каталоге шаблонов проекта (`templates`).

Теперь создайте каталог `admin` внутри `templates`, и скопируйте в этот каталог шаблон `admin/base_site.html` из каталога шаблонов админки непосредственно из исходного кода Django (`django/contrib/admin/templates`).

Где найти исходный код Django?

Если вы не можете найти исходный код Django, выполните следующую команду:

```
$ python -c "import django; print(django.__path__)"
```


Теперь отредактируйте файл и замените `{{ site_header|default:_('Django administration') }}` (включая скобки) на название вашего сайта. Должно получиться приблизительно следующее:

```
{% block branding %}
<h1 id="site-name"><a href="{% url 'admin:index' %}">Polls Administration</a></h1>
{% endblock %}
```

Мы использовали такой подход, чтобы показать, как переопределять шаблоны. На самом деле вы можете использовать атрибут `django.contrib.admin.AdminSite.site_header` для этого.

Этот файл шаблонов содержит много текста вида `{% block branding %}` и `{{ title }}`. `{%}` и `{{}}` являются частью языка шаблонов Django. Когда Django рендерит `admin/base_site.html`, язык шаблонов выполнится, чтобы получить HTML страницы.

Вы можете переопределить любой шаблон админки Django. Чтобы переопределить шаблон, сделайте все то же, что и с шаблоном `base_site.html` – скопируйте шаблон в ваш каталог шаблонов и измените его.

Настройка шаблонов вашего приложения

Внимательные читатели спросят: если `DIRS` по умолчанию ничего не содержит, как Django находит стандартные шаблоны админки? Ответ следующий: так как `APP_DIRS` равен `True`, Django автоматически ищет каталог `templates/` в каждом приложении (не забываем, что `django.contrib.admin` является приложением).

Наше приложение для опросов не сложное и не требует переопределения шаблонов. Но если оно станет сложнее и потребует переопределения шаблонов админки, лучше переопределять их для *приложения*, а не всего *проекта*. В таком случае вы можете добавить приложение к любому проекту, и быть уверенным, что будут использоваться модифицированные шаблоны.

Настройка главной страницы интерфейса администратора

Вам может понадобиться настроить главную страницу админки Django.

По умолчанию она показывает в алфавитном порядке все приложения из `INSTALLED_APPS`, которые были зарегистрированы в админке. Возможно вы захотите поменять структуру страницы. В конце концов главная страница является самой важной, и она должна быть удобна для использования.

Необходимо переопределить шаблон `admin/index.html`. (Как и для `admin/base_site.html`, скопируйте этот шаблон в свой проект). Откройте файл и вы увидите, что там используется переменная `app_list`. Она содержит все установленные приложения. Вместо использования этой переменной, вы можете “захардкодить” ссылки к страницам, если считаете, что так будет лучше.

На этом учебник для новичков окончен. Итак, вы прочитали весь вводный материал и решили, что хотите продолжать использовать Django. Мы коснулись только поверхности документации по Django. **Итак, что дальше?** На данный момент вы должны знать достаточно, чтобы создать собственный проект и начать с ним разбираться. Здесь приложено немало усилий, чтобы сделать документацию по Django полезной, легко читаемой и как можно более полной.

А теперь ПРОЙДИТЕ ТЕСТ ПО ПРОЙДЕННОМУ МАТЕРИАЛУ!

