Django - Урок 014. Вывод списка популярных статей на Django

[Django](https://evileg.com/ru/search/?search=%23Django), [статистика](https://evileg.com/ru/search/?search=%23%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0), [популярные статьи](https://evileg.com/ru/search/?search=%23%D0%BF%D0%BE%D0%BF%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%8B%D0%B5%20%D1%81%D1%82%D0%B0%D1%82%D1%8C%D0%B8)

На многих блогах и новостных сайтах для удержания внимания аудитории используются такие приёмы, как рейтинг популярных за неделю статей, похожие публикации, а у некоторых крупных ресурсов ещё и рекомендации по предпочтениям пользователей.

Первое, что было решено сделать - это вывод списка популярных статей. Первый вариант популярных статей был основан на общем счётчике просмотров и в итоге выводил статьи с самым большим количество просмотров. Это в целом плохой вариант, поскольку в ТОПе в итоге окажутся статьи просто набравшие самое большое количество просмотров за всё время.

Следовательно нужно было что-то менять. В результате был внедрен вывод популярных статей за последние 7 дней в самом простом варианте. То есть, добавлена была таблица в которую вносится количество просмотров статей по дням. Конечно, точность подсчётов при большой нагрузке может очень сильно варьироваться, но пока посещаемость не достигла 5000 - 10000 уникальных посетителей в сутки - это не настолько важно.

А теперь давайте разберём пример того, как сделать такой список популярных статей средствами [Django](https://evileg.com/ru/knowledge/django/).

Модели статей и статистики

Приведу урезанный вариант модели для статей, поскольку в данном случае меня интересует только заголовок статьи.

1. class Article(models.Model):
2. class Meta:
3. db\_table = "article"
5. title = models.CharField('Название статьи', max\_length=200)
7. def \_\_str\_\_(self):
8. return self.title

А для модели статистики просмотров по статьям код будет следующий.

1. class ArticleStatistic(models.Model):
2. class Meta:
3. db\_table = "ArticleStatistic"
5. article = models.ForeignKey(Article) # внешний ключ на статью
6. date = models.DateField('Дата', default=timezone.now) # дата
7. views = models.IntegerField('Просмотры', default=0) # количество просмотров в эту дату
9. def \_\_str\_\_(self):
10. return self.article.title

13. class ArticleStatisticAdmin(admin.ModelAdmin):
14. list\_display = ('\_\_str\_\_', 'date', 'views') # отображаемые поля в админке
15. search\_fields = ('\_\_str\_\_', ) # поле, по которому производится поиск

В данном коде сделано две модели:

1. Сама модель для сбора статистики
2. Модель для отображения данных в админке.

Эти модели будут написаны в файле **models.py.**

Не забываем также зарегистрировать эти модели в админке в файле **admin.py.**

1. from django.contrib import admin
3. from .models import Article, ArticleStatistic, ArticleStatisticAdmin
5. admin.site.register(Article)
6. admin.site.register(ArticleStatistic, ArticleStatisticAdmin)

После чего не забываем сделать миграцию базы данных.

python manage.db makemigrations

python manage.db migrate

Представление для отображения страницы

После того, как у нас добавлены модели, осталось добавить вывод популярных статей и сделать подсчёт статистики при запросе статьи с сервера.

1. from django.views import View
2. from django.shortcuts import render\_to\_response, get\_object\_or\_404
3. from django.utils import timezone
4. from django.db.models import Sum
6. # Для работы с моделями статей у меня используется модуль knowledge
7. from knowledge.models import Article, ArticleStatistic

10. # тогда как для отображения статей используется модуль post
11. # Это было сделано для того, чтобы URL статей был следующего вида
12. # /post/42/ - где 42 - это id статьи в базе данных
13. class EArticleView(View):
14. template\_name = 'knowledge/article.html' # Шаблон статьи
16. def get(self, request, \*args, \*\*kwargs):
17. article = get\_object\_or\_404(Article, id=self.kwargs['article\_id']) # Забираем статью из базы данных
18. context = {}
20. # Далее забираем объект сегодняшней статистики или создаём новый, если требуется
21. obj, created = ArticleStatistic.objects.get\_or\_create(
22. defaults={
23. "article": article,
24. "date": timezone.now()
25. },
26. # При этом определяем, забор объекта статистики или его создание
27. # по двум полям: дата и внешний ключ на статью
28. date=timezone.now(), article=article
29. )
30. obj.views += 1 # инкрементируем счётчик просмотров и обновляем поле в базе данных
31. obj.save(update\_fields=['views'])
33. # А теперь забираем список 5 последний самых популярных статей за неделю
34. popular = ArticleStatistic.objects.filter(
35. # отфильтровываем записи за последние 7 дней
36. date\_\_range=[timezone.now() - timezone.timedelta(7), timezone.now()]
37. ).values(
38. # Забираем интересующие нас поля, а именно id и заголовок
39. # К сожалению забрать объект по внешнему ключу в данном случае не получится
40. # Только конкретные поля из объекта
41. 'article\_id', 'article\_\_title'
42. ).annotate(
43. # Суммируем записи по просмотрам
44. # Всё суммируется корректно с соответствием по запрашиваемым полям объектов
45. views=Sum('views')
46. ).order\_by(
47. # отсортируем записи по убыванию
48. '-views')[:5] # Заберём последние пять записей
50. context['popular\_list'] = popular # Отправим в контекст список статей
52. return render\_to\_response(template\_name=self.template\_name, context=context)

Отображение в шаблоне

Ну а теперь просто отобразим в шаблоне страницы список популярных статей за неделю. Оговорюсь, что приведу только шаблон для отображения списка. Этого достаточно для понимания того, как вывести список статей на странице.

Не забываем также что я использую [django\_bootstrap3](https://evileg.com/ru/post/6/), поэтому шаблон выглядит соответствующим образом.

1. {% if popular\_list %}{% load bootstrap3 %}
2. <ul class="list-group">
3. <li class="list-group-item active"><strong>Популярные публикации за неделю</strong></li>
4. {% for pop\_article in popular\_list %}
5. <li class="list-group-item">
6. <a href="{% url 'post:article' pop\_article.article\_id %}">{{ pop\_article.article\_\_title }}</a>
7. </li>
8. {% endfor %}
9. </ul>
10. {% endif %}

Чтобы не было недомолвок, приведу шаблон url для запроса страниц с сайта:

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

Итог

Список популярных статей будет похож на тот, что используется на данном сайте, только без общего количества просмотров, а в админке статистика будет выглядеть следующим образом: