# Django学习

1. 项目创建过程：
2. 创建website项目： django-admin startproject website
3. 创建一个app(blog): python manahe.py startapp blog
4. 修改了setting.py，urls.py文件
5. 创建了模板，blog/templates/index.html
6. 修改views.py文件。
7. Python Manage.py runserver 启动服务器
8. 如何动态的获取数据：
9. 书写格式：

{{ 数据内容 }}

1. 控制语句：
2. if语句：

{% if %}

{% else %}

{% endif %}

1. For 语句：

{% for %}

{% empty %}//判断是否为空

{{ forloop.counter}}//循环的次数

{{ forloop.counter0 }}//循环次数，从零开始

{{} forloop.revcounter }}//剩余循环次数

{{} forloop.revcounter0 }}//剩余循环次数

{{ forloop.begin }} //循环的开始

{{ forloop.end }}//循环的结束

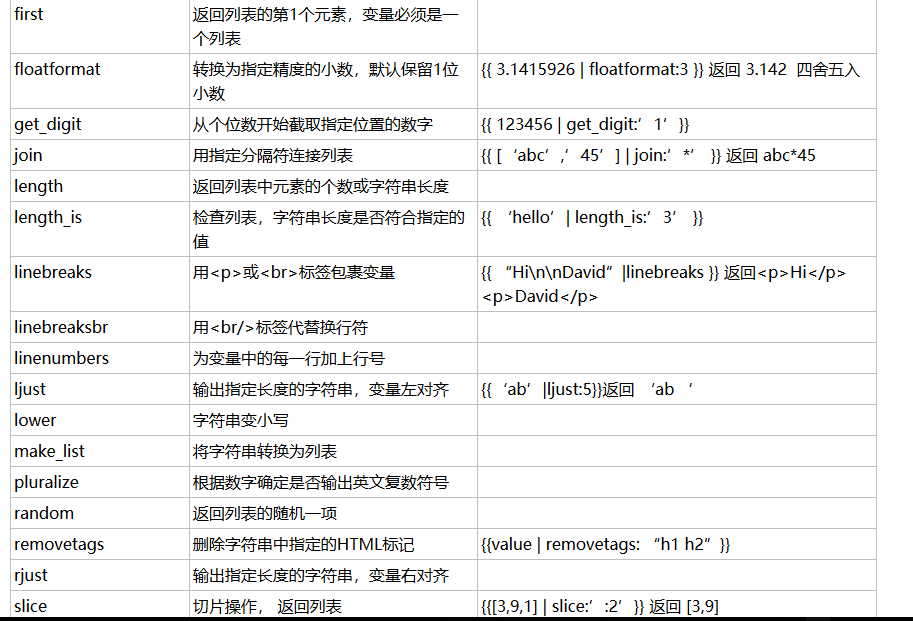
{% endfor %}

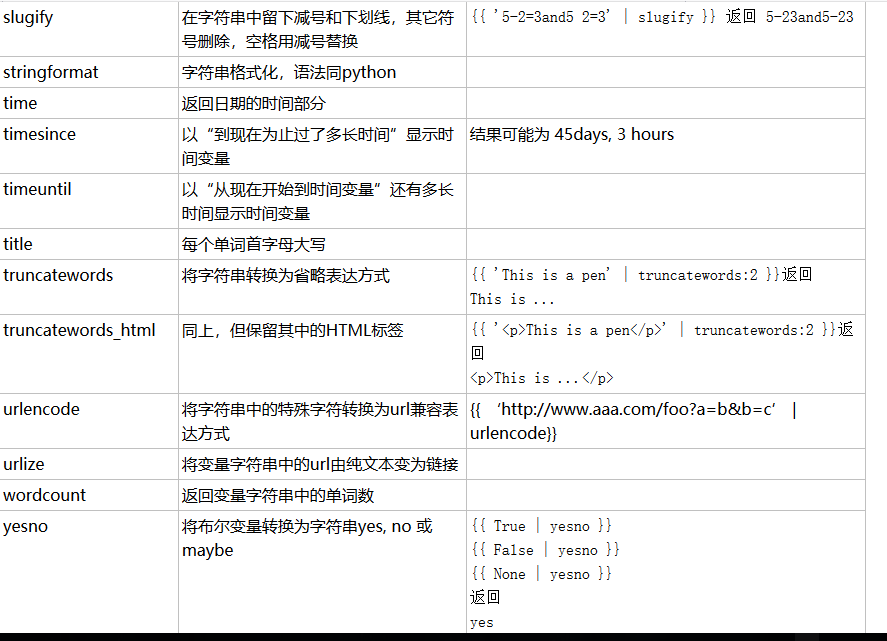
1. 过滤器：

{{输入 | 函数}}

1. django中的过滤器：







1. 新定义过滤器：

from django import template

register = template.Library()

@register.filter

def percent(value):

return value+"%"

1. Url配置：
2. 通过url传值：
3. id=request.GET.get('id')

<http://www.baidu.com?id=’yang’>

1. <http://time/123/abc>

url(r'^blog/helo/(\d{3})/(\w{3})$',views.helo)

规定参数：

url(r'^blog/helo/(?P<id>\d{3})/(?P<name>\w{3})$',views.helo

1. 和数据库交互：
2. orm:对象与关系的映射
3. 数据库的数据同步
4. 在models.py中创建一个类
5. 在命令行先执行 python manage.py makemigrations

然后执行 python manage.py migrate

然后将创建表

1. 反映射：

python manage.py inspectdb

1. 对数据库的操作：
2. class（代表类名）.objects.all()

等价于 select \* from class;

1. 升降序：

Class.objects.all().order\_by(‘字段’) --升序

Class.objects.all().order\_by(‘-字段’) ---降续

1. 查找特定条件数据-用过滤器：

Class.objects.filter(条件)

1. orm中逻辑判断的表示：

querySet.distinct() 去重复

\_\_exact 精确等于 like 'aaa'

\_\_iexact 精确等于 忽略大小写 ilike 'aaa'

\_\_contains 包含 like '%aaa%'

\_\_icontains 包含 忽略大小写 ilike '%aaa%'，但是对于sqlite来说，contains的作用效果等同于icontains。

\_\_gt 大于

\_\_gte 大于等于

\_\_lt 小于

\_\_lte 小于等于

\_\_in 存在于一个list范围内

\_\_startswith 以...开头

\_\_istartswith 以...开头 忽略大小写

\_\_endswith 以...结尾

\_\_iendswith 以...结尾，忽略大小写

\_\_range 在...范围内

\_\_year 日期字段的年份

\_\_month 日期字段的月份

\_\_day 日期字段的日

\_\_isnull=True/False

1. 获得一个单个对象：

Class.objects.get(条件) --每次获取一行数据

2.改 --对数据库进行修改

（1）对某一项数据进行修改：

student=Student.objects.get(id=1) //获取到这一条数据

student.name='杨' //修改name字段

student.age=21 //修改年龄字段

student.save() //保存到数据库

（2）对一部分数据进行批量更新：

Class.objects.filter(条件).update(要修改的数据) //返回受影响的行数

例子：

Student.objects.filter(age\_\_gte=20).update(name='abc')

1. 增加：
2. 例子：

newstu=Student(name='菜',age=40,intime='2010-01-02')

newstu.save()

1. 删除操作：

Student.objects.get(id=1).delete()

1. 实现和数据库一对一，一对多和多对多的关系：
2. models的类型：

1、models.AutoField　　自增列 = int(11)

如果没有的话，默认会生成一个名称为 id 的列，如果要显示 的自定义一个自增列，必须将给列设置为主键 primary\_key=True。

2、models.CharField　　字符串字段

必须 max\_length 参数

3、models.BooleanField　　布尔类型=tinyint(1)

不能为空，Blank=True

4、models.ComaSeparatedIntegerField　　用逗号分割的数字=varchar

继承CharField，所以必须 max\_lenght 参数

5、models.DateField　　日期类型 date

对于参数，auto\_now = True 则每次更新都会更新这个时间；auto\_now\_add 则只是第一次创建添加，之后的更新不再改变。

6、models.DateTimeField　　日期类型 datetime

同DateField的参数

7、models.Decimal　　十进制小数类型 = decimal

必须指定整数位max\_digits和小数位decimal\_places

8、models.EmailField　　字符串类型（正则表达式邮箱） =varchar

对字符串进行正则表达式

1. models.FloatField　　浮点类型 = double
2. models.IntegerField　　整形

11、models.BigIntegerField　　长整形

12、models.IPAddressField　　字符串类型（ip4正则表达式）

13、models.GenericIPAddressField　　字符串类型（ip4和ip6是可选的）

　　参数protocol可以是：both、ipv4、ipv6

验证时，会根据设置报错

1. models.NullBooleanField　　允许为空的布尔类型
2. models.PositiveIntegerFiel　　正Integer
3. models.PositiveSmallIntegerField　　正smallInteger
4. models.SlugField　　减号、下划线、字母、数字

18、models.SmallIntegerField　　数字

数据库中的字段有：tinyint、smallint、int、bigint

19、models.TextField　　字符串=longtext

20、models.TimeField　　时间 HH:MM[:ss[.uuuuuu]]

21、models.URLField　　字符串，地址正则表达式

22、models.BinaryField　　二进制

23、models.ImageField 图片

24、models.FilePathField 文件

1. admin:
2. 创建数据库超级用户：

Python manage.py createsuperuser (name:monty password:monty1997)