## 自定义模板过滤器

创建文件结构

在app里面 创建**templatetags** 这是一个 python package 在 templatetags 创建 \_\_init\_\_.py xxxxfilter.py

自定义过滤器的思路

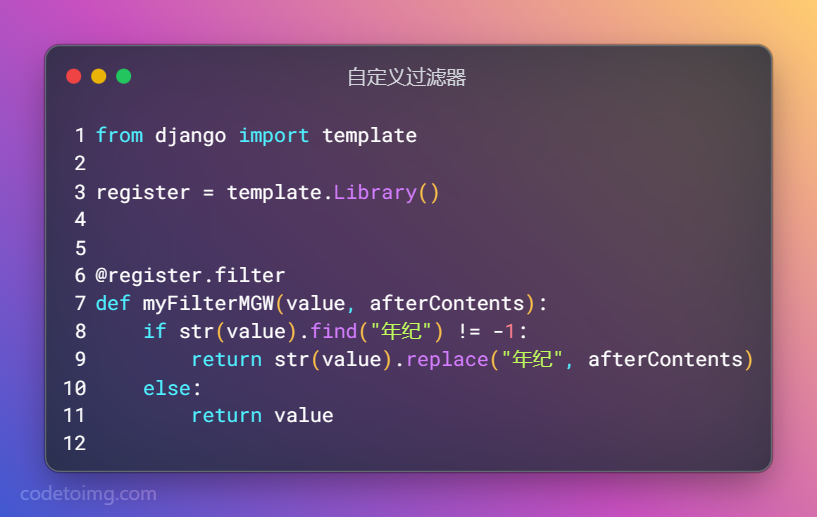
1. 导入模板库template
2. 实例化一个register模板注册器 注册过滤器
3. 用一个函数表示过滤器的内容

过滤器格式

@register.filter

Def 函数名(template 传递过来的变量， 过滤器传递参数1， 过滤器传递参数2):

Xxxxx



## 在 templates 中重命名变量

变量逻辑上重命名

{% with xxxxxxx as x%}

{% endwith %}

{%with aa=aaaaaaaaaa%} 等号之间不能有空格

{%endwith%}

## Sqlite3

**增 删 改 查**

**分析需要什么样的表以及数据字典**

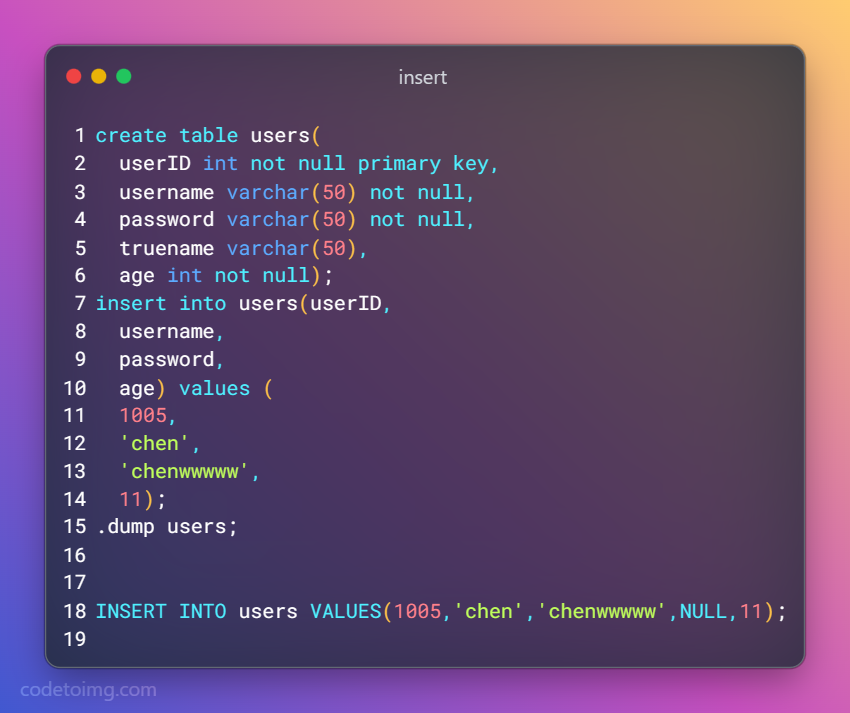
创建数据表

Create table tableName(字段1 字段类型 是否为空 是否是主键);

Insert into tableName(字段)values(值)

如果字段不是 NOT NULL 那么不输入字段的时候 默认的就是 NULL





更新

Update table\_name set 字段1=更新后的值 where 字段 = 值

**更新一定要提前确定好 注意where 后面的条件 千万不能是 整个表**

**在更新之前，最好先把需要更新的要素先 select 出来， 防止误操作**

update users set truename='chenwei' where userID = 1005;

删除

需要结合where 语句

Delete from table\_name 这个语句针对删除的是表中所有的数据 一般都是程序刚开始测试使用

一般都是结合 where 使用

一般都是 先查询 再删除

1. Select … from
2. Delete from

The Full Process



## 获取用户的信息并放到数据库里面

1. 在form 表单里 添加 {%csrf\_token%}
2. 在路由里面添加新的路由 register/add/ 定义路由
3. 将新路由定位到接受数据的方法 registerAdd view 里面添加方法
4. 通过 request.POST[‘这里写form内部的控件的name的名称’]，会返回对应的值，用一个变量来接受。

Django.db.connection 类

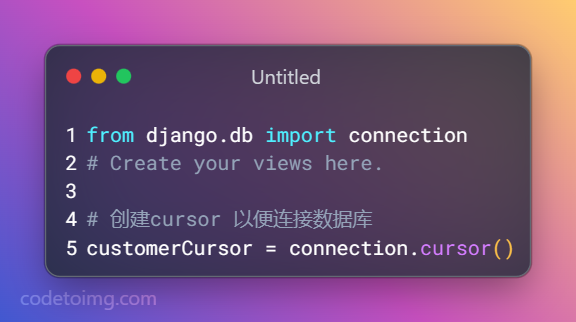
首先在settings.py ；里面查看自己使用的数据库

在 view 里面 导入数据库连接

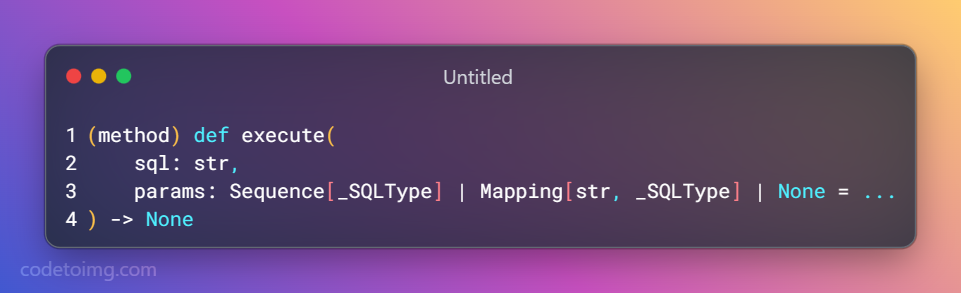
From Django.db import connetion

创建数据库 cursor = connetion.cursor()

在register 函数里 添加数据库指令





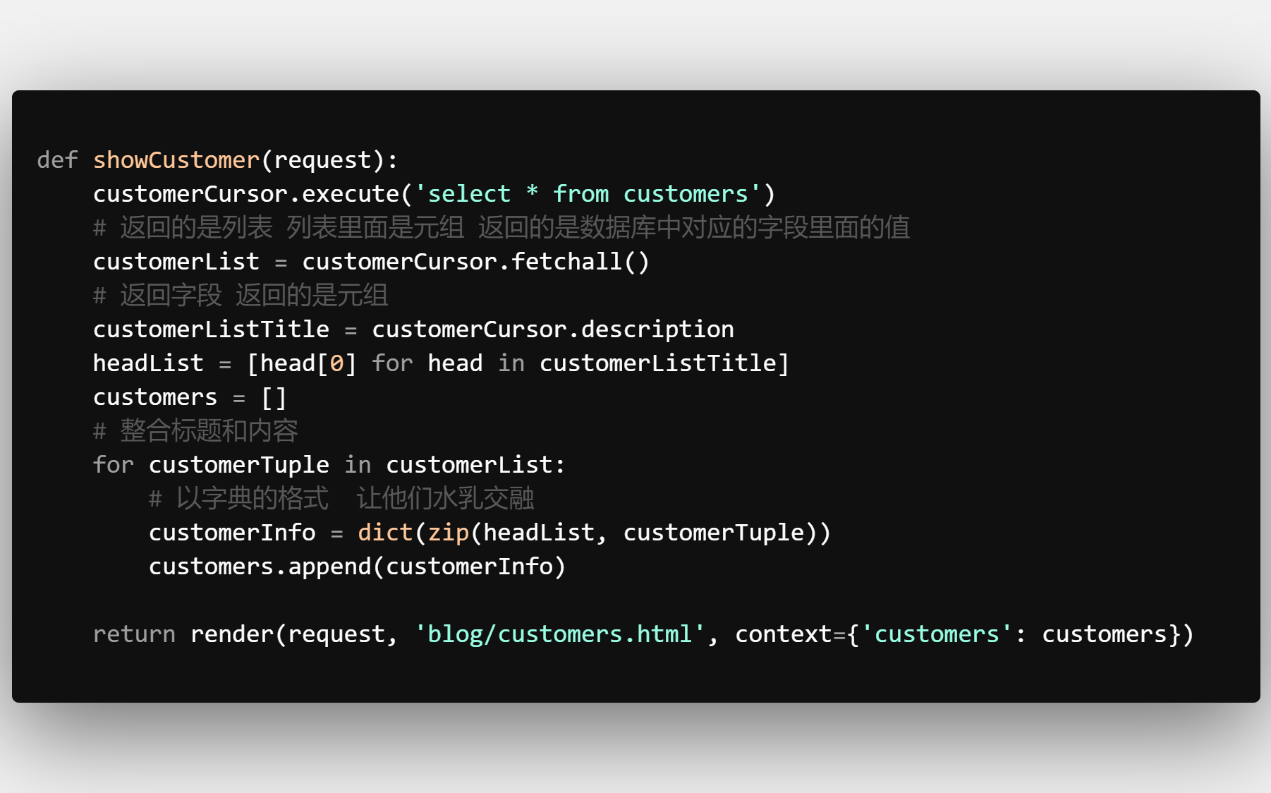


Execute()方法第一个参数是 sql 语句， 中间的变量用 %s 代替， 第二个参数是 元组， 值为之前的sql语句中的%s依次替换。

## 数据查询思路

1. 数据库连接
   1. 使用 Django.db 里面的 connection 类来连接 setting.py 里面的数据库
   2. 创建游标cursor
2. 查询数据库
   1. 使用 cursor.execute() 方法 执行 sql 的 select 语句
3. 得到数据
   1. 使用 cursor.fetchall() 得到所有查询的数据，然后可以输出到网页。 返回的是 列表 列表里面的组成是 元组
   2. **注意 cursor.fetchall() 只能返回一次数据 要想多次返回 就要多次执行 cursor.execute() cursor.fetchall()**

**数据整理思路**

****

zip(\*iterables) --> A zip object yielding tuples until an input is exhausted.

>>> list(zip('abcdefg', range(3), range(4)))  
[('a', 0, 0), ('b', 1, 1), ('c', 2, 2)]

The zip object yields n-length tuples, where n is the number of iterables passed as positional arguments to zip(). The i-th element in every tuple comes from the i-th iterable argument to zip(). This continues until the shortest argument is exhausted.

SQL 语句参数占位符

PYTHON 中sql 语句占位符统一使用%s

无论传入的是字符串还是数值，都不要再单独加入引号等符号。

更新数据开发步骤

1. 新建更新数据的路由， 对应后台函数用作更新数据
2. 页面上表单数据行为执行更新数据
3. 更新数据的函数中接受数据， 传入 SQL 语句
4. 执行SQL语句

迁移是非常强大的功能，它能让你在开发过程中持续的改变数据库结构而不需要重新删除和创建表 - 它专注于使数据库平滑升级而不会丢失数据。我们会在后面的教程中更加深入的学习这部分内容，现在，你只需要记住，改变模型需要这三步：

* 编辑 **models.py** 文件，改变模型。
* 运行 [**python manage.py makemigrations**](https://docs.djangoproject.com/zh-hans/4.1/ref/django-admin/#django-admin-makemigrations) 为模型的改变生成迁移文件。
* 运行 [**python manage.py migrate**](https://docs.djangoproject.com/zh-hans/4.1/ref/django-admin/#django-admin-migrate) 来应用数据库迁移。