# ORM基本CRUD实现

> ORM对象关系模式中,操作对象就等同于操作数据库表

## 进入Shell Ipython环境

```
C:\Users\Administrator\Desktop\myblog>python manage.py shell
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.5.0 -- An enhanced Interactive Python. Type '?' for help.
# 导入要操作的model类
In [1]: from blog.models import Article
```

## 根据ID查询数据

```
In [2]: data = Article.objects.get(id = 2)
# 重写 __str__ 返回要输出的值即可
In [3]: data
Out[3]: <Article: 2 aaa bbb>
```

## 对象的插入与更新操作

```
In [4]: data = Article(title='我是标题',content='我是正文')
# 等同于数据库新增加一条记录
In [5]: data.save()

In [6]: data
Out[6]: <Article: 5 我是标题 我是正文>
In [7]: data.content = '更新正文内容'
# 持久态的数据调用save()执行的是update语句
In [8]: data.save()
```

## 对象的删除操作

```
In [12]: data = Article.objects.get(id = 5)

In [13]: data
Out[13]: <Article: 5 我是标题 更新正文内容>
# 把查询出来的数据删除掉
In [14]: data.delete()
Out[14]: (1, {'blog.Article': 1})
# 查询所有数据
In [15]: data = Article.objects.all()
# 对数据进行循环输出
In [16]: for row in data:
    ...:     print(row)
2 aaa bbb
4 my title my content
```

## 通过filter过滤查询

> 格式为 field__lookuptype=value 详细的操作参考官方API文档

```
# gte 代表 >= 多个条件and条件用逗号隔开
In [3]: data = Article.objects.filter(id__gte=2,title='aaa')
In [4]: data
Out[4]: <QuerySet [<Article: 2 aaa bbb>]>
```

```
# gt 代表 > lt 代表 <
In [5]: data = Article.objects.filter(id__gt=2,title='aaa')
In [6]: data
Out[6]: <QuerySet []>
```

# 文章的CRUD设计

## 查询所有文章并显示到页面中

- 取出数据库中所有文章对象

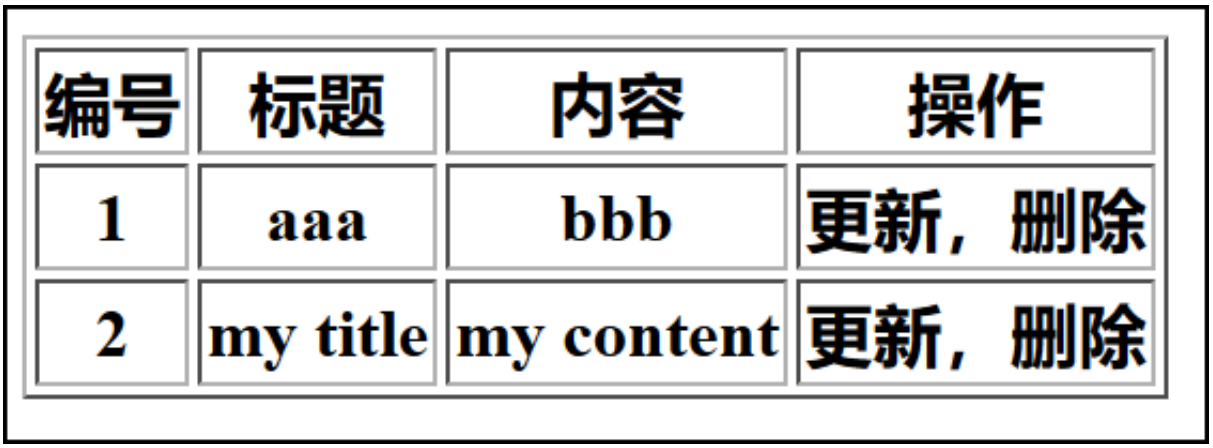
```
# Create your views here.
def index(request):
    # 查询所有文章对象
    articles = Article.objects.all()
    return render(request,'index.html',{'articles':articles})
```

- 将文章对象打包成列表，传递到前端 {% for item in list %} ..... {% endfor%}

```
<body>
    <!--
    <h2>{{ article.id }} | {{ article.title }} | {{ article.content }}</h2>
    -->
    <table border="1">
        <tr>
            <th>编号</th>
            <th>标题</th>
            <th>内容</th>
            <th>操作</th>
        </tr>
        {% for article in articles %}
        <tr>
            <!-- forloop 是for循环的内部变量-->
            <th>{{ forloop.counter }}</th>
            <th>{{ article.title }}</th>
            <th>{{ article.content }}</th>
            <th>更新，删除</th>
```

```
        </tr>
        {% endfor %}
    </table>
</body>
```

- 显示效果如下 (后面会采用bootstrap来实现外观)



| 编号 | 标题 | 内容 | 操作 |
| --- | --- | --- | --- |
| 1 | aaa | bbb | 更新，删除 |
| 2 | my title | my content | 更新，删除 |

## 根据id实现某篇文件删除操作

- 在页面传递参数,Django参数支持restful风格 /参数/参数.....

```
<!-- 无需参数变量名 -->
<th><a href="/delete/{{article.id}}">删除</a></th>
```

- 在urls中定义参数的类型，和view对应处理参数的函数名称

```
urlpatterns = [
    path('admin/', admin.site.urls),
    .......
    path('delete/<int:article_id>', bv.delete),
]
```

- 在view控制层根据ID查询要删除数据

```
def delete(request, article_id):
    # 根据id删除文章
    Article.objects.get(pk=article_id).delete()
```

```
    # 后续此处会采用ajax来实现
    articles = Article.objects.all()
    return render(request, 'index.html', {'articles': articles})
```

## 根据id更新某篇博客文章

- 单击更新时候要先通过ID查询要更新的数据

```
<th><a href="/get_id/{{ article.id }}">更新</a>
```

- 在urls.py中配置更新链接需要访问的view控制器方法

```
urlpatterns = [
    path('admin/', admin.site.urls),
    ......
    path('get_id/<int:article_id>', bv.get_id),
]
```

- 跳转到View控制权进行相应的数据库处理操作

```
def get_id(request, article_id):
    article = Article.objects.get(id=article_id)
    # 存储参数并且直接跳转到update.html页面
    return render(request, "update.html", {'article': article})
```

- 查询完要更新的数据，则把数据回显到更新页面

```
<!-- 提交时必须要 /路径/ 否则post会提交失败 -->
<form action="/update/" method="post">
    <!-- django默认支持csrf验证,如果不写请求会被中断-->
    {%csrf_token%}
    <label>
        文章标题
        <input type="text" name="title" value="{{ article.title }}"
/>
```

```
    </label>
    <br />
    <label>文章内容:
        <input type="text" name="content" value="{{ article.content }}" />
    </label>
    <br />
    <input type="submit" value="提交"/>
    <input type="hidden" name="id" value="{{ article.id }}" />
</form>
```

- 更新完毕后进行重定向操作

```
from django.http import HttpResponseRedirect

def update(request):
    article = Article.objects.get(id=request.POST.get('id'))
    article.title = request.POST.get('title')
    article.content = request.POST.get('content')
    article.save()
    return HttpResponseRedirect("/index/")
```

# 文章插入操作(课堂练习)