Django数据库操作API详解（二）

Django 不仅提供了返回 QuerySet 类型的 API，而且还向来发者提供了一些其他有用的 API，它们会返回整数或者布尔类型的值，下面就让我们一起来认识这些常用方法。

1. 常用API总结

1) len()与count()统计对象数量

这两个方法都可以获取 QuerySet 对象的数量，但是它们稍微有一些不同，len 方法相对于 count 方法效率较低。示例如下：

```
#len方法查询所有数据再计算迭代对象的数量
In [1]: len(Book.objects.all())
Out[1]: 6
#count()相当于执行select count(字段)直接返回统计结果
In [2]: Book.objects.all().count()
Out[2]: 6
```

所以经过上面的示例说明，建议在需要获取 QuerySet 数量的时候使用 count()方法。

2) exists()条件判断数据记录是否存在

在很多情况下，我们需要根据给定的条件判断数据是否在 Model 中存在。Django 提供了一个简便的方法 exists() 方法，它的返回值是一个布尔值，如果存在的情况下将返回 Ture，反之返回 False。示例如下：

```
In [3]: Book.objects.filter(title__exact="Python").exists()
Out[3]: True
```

3) update方法更新Model实例

使用 update 方法来更新 Model 实例是非常简便的，它相比 save 方法来说，对开发者更加友好。update 方法可以可以一次更新多个对象，并返回一个整数，标识被改动的数据记录数量，示例说明如下：

```
In [4]: Book.objects.filter(title__exact="Python").update(title="Python Django")
Out[4]: 1
```

4) delete方法删除数据记录

如果想删除一条数据记录，我们可以调用 delete() 方法。举例如下：

```
#调用delete方法删除
In [4]: Book.objects.filter(title__exact="Tornado").delete()
Out[4]: (3, {'index.Author_books': 2, 'index.Book': 1})
```

delete 方法返回一个二元组：第一个元素代表删除实例的总个数，第二个元素是字典类型，记录每一个 Model 类型删除的实例个数。由于作者表和书籍表之间存在多对多的关系，所以此处删除了两个 Model 实例，而书籍表中删除 title=Tornado的数据记录。

本节知识作为《Django查询数据库操作详解（一）》的第二部分。通过这两节知识的学习，大家对这些 API 一定不再陌生，只要勤加练习，就会熟练掌握它们，学会使用这些 API 会给我们的开发工作带来极大的方便。