登录

2019年04月30日 阅读 591

关注

Flask框架从入门到精通之模型查询（十三）

> 知识点：1、模型查询

一、查询

其实我们对模型的主要操作就是查询，在Flask-SQLAlchemy中，支持了很多的查询方法。查询操作是通过query对象操作数据。最基本的查询是返回表中所有数据，可以通过过滤器进行更精确的数据库查询。

二、查询过滤器

Flask-SQLAlchemy中常用过滤器：

过滤器	说明
filter()	把过滤器添加到原查询上，返回一个新查询
filter_by()	把等值过滤器添加到原查询上，返回一个新查询
limit()	使用指定的值限定原查询返回的结果
offset()	偏移原查询返回的结果，返回一个新查询
order_by()	根据指定条件对原查询结果进行排序，返回一个新查询
group_by()	根据指定条件对原查询结果进行分组，返回一个新查询

三、查询执行器

Flask-SQLAlchemy中常用执行器：

方法	说明
all()	以列表形式返回查询的所有结果
first()	返回查询的第一个结果，如果未查到，返回None

first_or_404() 方法	返回查询的第一个结果，如果未查到，返回404 说明
get()	返回指定主键对应的行，如不存在，返回None
get_or_404()	返回指定主键对应的行，如不存在，返回404
count()	返回查询结果的数量
paginate()	返回一个Paginate对象，它包含指定范围内的结果

四、查询

我们在ipython3中测试:

```
from flask_db import *
```

- **查询全部**

```
Type.query.all()
#[<Type 4>, <Type 2>, <Type 1>, <Type 3>]
```

```
Hero.query.all()
#[<Hero 1>, <Hero 2>, <Hero 3>, <Hero 4>, <Hero 5>]
```

- **根据主键查询**

```
 Hero.query.get(1)
 # <Hero 1>
```

- **查询第一条**

```
Hero.query.first()
#   <Hero 1>
```

- **根据名字过滤**

```
# 查询名字是王昭君的对象
 Hero.query.filter_by(name='王昭君').all()
 # [<Hero 3>]
 Hero.query.filter_by(name='王昭君').first()
 # <Hero 3>
```

- **逻辑与**

```
# 查询名字以君结尾并且type_id等于3的
Hero.query.filter_by(name='王昭君',type_id=3).first()
# <Hero 3>
```

- **filter的逻辑与**

```
# 查询名字以君结尾并且type_id等于3的
# filter_by是=号, 在filter中是==号
# filter_by不需要指定类名, filter需要指定
Hero.query.filter(Hero.name=='王昭君',Hero.type_id==3).first()
# <Hero 3>
```

- **逻辑或**

```
# 查询名字以君结尾或type_id等于3的
from sqlalchemy import or_
Hero.query.filter(or_(Hero.name.endswith('君'),Hero.type_id==3)).all()
#  [<Hero 3>, <Hero 4>]
```

- **偏移查询**

```
# 跳过前2条数据, 从第三条数据开始取全部
Hero.query.offset(2).all()
# [<Hero 3>, <Hero 4>, <Hero 5>]

# 跳过前2条数据, 从第三条数据开始取2条
Hero.query.offset(2).limit(2).all()
#[<Hero 3>, <Hero 4>]
```

- **排序**

```
#降序查询
Hero.query.order_by(Hero.id.desc()).all()
# [<Hero 5>, <Hero 4>, <Hero 3>, <Hero 2>, <Hero 1>]

#升序查询
Hero.query.order_by(Hero.id.asc()).all()
# [<Hero 1>, <Hero 2>, <Hero 3>, <Hero 4>, <Hero 5>]
```

- **分组**

```
from sqlalchemy import func
# 根据type_id进行分组统计
 db.session.query(Hero.type_id,func.count(Hero.type_id)).group_by(Hero.type_id).all()
```

- **关联查询**

```
hero = Hero.query.get(1)
hero.type
#  <Type 1>

type = Type.query.get(1)
type.heros
# [<Hero 1>]
```

如果想让查询的过程中显示出自定义信息。可以在模型类中重写__repr__方法。例如我在两个模型加上如下代码：

```
def __repr__(self):
        return self.name
```

在进行查询：

```
type = Type.query.get(1)
type.heros
# [后羿]
```

复制代码

欢迎关注我的公众号：

文章分类

后端

文章标签

Flask

相关推荐