Flask SQLAlchemy

Quik Reference

Ferreira Juan David

5 de febrero de 2021

Resumen

Ahora que ha declarado modelos , es hora de consultar los datos de la base de datos. Usaremos las definiciones de modelo del capítulo Inicio rápido.

Insertar registros

Antes de que podamos consultar algo, tendremos que insertar algunos datos. Todos sus modelos deben tener un constructor, así que asegúrese de agregar uno si lo olvidó. Los constructores solo los usa usted, no `SQLAlchemy` internamente, por lo que depende completamente de usted cómo los defina.

Insertar datos en la base de datos es un proceso de tres pasos:

1. Creamos el objeto `Python`.
2. Agregamola sesión.
3. Realizamos un *commit* del objeto en la sesión.

La sesión aquí no es la sesión de `Flask`, sino la de `Flask-SQLAlchemy`. Es esencialmente una versión reforzada de una transacción de base de datos. Así es como funciona:

```
>>> from yourapp import User
>>> me = User('admin', 'admin@example.com')
>>> db.session.add(me)
>>> db.session.commit()
```

Muy bien, eso no fue difícil. ¿Qué pasa en qué momento? Antes de agregar el objeto a la sesión, `SQLAlchemy` básicamente no planea agregarlo a la transacción. Eso es bueno porque aún puede descartar los cambios. Por ejemplo, piense en crear la publicación en una página, pero solo desea pasar la publicación a la plantilla para obtener una vista previa en lugar de almacenarla en la base de datos.

La llamada a la función `add()` agrega el objeto. Emitirá una declaración `INSERT` para la base de datos, pero debido a que la transacción aún no está confirmada, no obtendrá una identificación de inmediato. Si realiza la confirmación, su usuario tendrá una `ID`:

```
>>> me.id
1
```

Eliminar registros

Eliminar registros es muy similar, en lugar de `add()` usar `delete()`:

```
>>> db.session.delete(me)
>>> db.session.commit()
```

Consultando registros

Entonces, ¿cómo recuperamos los datos de nuestra base de datos?. Para este propósito, `Flask-SQLAlchemy` proporciona un queryatributo en su clase `Model`. Cuando acceda, obtendrá un nuevo objeto de consulta sobre todos los registros. A continuación, puede utilizar métodos como `filter()` filtrar los registros antes de activar la selección con `all()` o `first()`. Si desea utilizar la clave principal, también puede utilizar `get()`.

Las siguientes consultas asumen las siguientes entradas en la base de datos:

Flask SQLAlchemy

id	username	email
1	admin	admin@example.com
2	peter	peter@example.org
3	guest	guest@example.com

Recuperar un usuario por nombre de usuario:

```
>>> peter = User.query.filter_by(username='peter').first()
>>> peter.id
2
>>> peter.email
u'peter@example.org'
```

Igual que el anterior, pero para un nombre de usuario no existente da `None`:

```
>>> missing = User.query.filter_by(username='missing').first()
>>> missing is None
True
```

Seleccionar un grupo de usuarios mediante una expresión más compleja:

```
>>> User.query.filter(User.email.endswith('@example.com')).all()
[<User u'admin'>, <User u'guest'>]
```

Ordenar usuarios por algo:

```
>>> User.query.order_by(User.username).all()
[<User u'admin'>, <User u'guest'>, <User u'peter'>]
```

Limitar usuarios:

```
>>> User.query.limit(1).all()
[<User u'admin'>]
```

Obteniendo usuario por clave primaria:

```
>>> User.query.get(1)
<User u'admin'>
```

Consultas en vistas

Si escribimos una función de vista de `Flask`, a menudo es muy útil devolver un error 404 para las vistas no definidas por el backend. Debido a que este es un idioma muy común, `Flask-SQLAlchemy` proporciona unos helpers para este propósito. En lugar de `get()` uno puede usar `get_or_404()` y en lugar de `first()` `first_or_404()`. Esto generará errores 404 en lugar de devolver `None`:

```
@app.route('/user/<username>')
def show_user(username):
    user = User.query.filter_by(username=username).first_or_404()
    return render_template('show_user.html', user=user)
```

Además, si desea agregar una descripción con `abort()`, también puede usarla como argumento.

```
>>> User.query.filter_by(username=username).first_or_404(description='There is no data with {}'.format(username))
```