SQLAlchemy 2.0 version of User.query.get(1) in Flask-SQLAlche

Asked 6 months ago Modified 2 days ago Viewed 3k times

9

The Problem

The `Query.get()` method is deprecated in SQLAlchemy 2.0. Accordingly, the Flask-SQLAlchemy query interface is considered legacy. Thus, running `User.query.get(1)` in my Flask-SQLAlchemy project gives the legacy warning shown below:

```
>>> User.query.get(1)
<stdin>:1: LegacyAPIWarning: The Query.get() method
is considered legacy as of the 1.x series of SQLAlchemy
and becomes a legacy construct in 2.0. The method is
now available as Session.get() (deprecated since: 2.0)
(Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
<User spongebob>
```

My Question

What is the new, SQLAlchemy 2.0-compatible version of `User.query.get(1)` in Flask-SQLAlchemy? More specifically, why does the Flask-SQLAlchemy documentation recommend Approach #2 below, even though Approach #1 appears to be the new version based on my reading of the SQLAlchemy 2.0 migration guide?

Approach #1: `db.session.get(User, 1)`

This first approach comes from the SQLAlchemy docs, specifically the SQLAlchemy 2.0 Migration - ORM Usage guide. Translating the "2.0 style" example in that guide to my Flask-SQLAlchemy project yields the following code, which works fine:

```
>>> db.session.get(User, 1)
<User spongebob>
```

This approach with `session.get()` isn't mentioned in the Flask-SQLAlchemy 3.0.x documentation as far as I can tell, except briefly in the API reference section on `get_or_404`.

Approach #2: `db.session.execute(db.select(User).filter_by(id=1)).scalar()`

This approach comes from the Flask-SQLAlchemy documentation, which suggests using `session.execute(select(...))` as a replacement for the legacy `Model.query` and `session.query`. This works fine, too:

```
>>> db.session.execute(db.select(User).filter_by(id=1)).scalar()
<User spongebob>
```

Approach #1 vs. Approach #2 vs. Legacy Approach

Approach #1 (`db.session.get(User, 1)`) appears to be most similar to the Legacy Approach (`User.query.get(1)`) because it caches the result in the `session` the first time it runs and won't emit additional calls to the database unnecessarily. This can be seen in the REPL with the echo turned on, i.e. `db.engine.echo = True`. In contrast, Approach #2 (`session.execute(select(...))`) goes to the database each time, as expected.

My Set Up / Environment

1. Versions: Flask 2.2.2, Flask-SQLAlchemy 3.0.3, and SQLAlchemy 2.0.1 in a virtual environment with Python 3.11.
2. I'm using the project structure defined in the Flask Mega-Tutorial, specifically Part IV Database.

flask sqlalchemy flask-sqlalchemy

Share Edit Follow

asked Feb 6 at 18:23
Patrick Cantwell
126 • 2 • 6

3 Approach #1 is the way, for the reason that you have described - it checks the session before querying the database. – snakecharmerb Feb 6 at 18:39

Add a comment

4 Answers

Sorted by: Highest score (default)

5

replacing all `Model.query.get(id)` by `db.session.get(Model, id))` is scary and not elegant: clear loss of readability.

We need a better solution !

Share Edit Follow

answered Apr 3 at 16:14
emmv
109 • 2

Add a comment

4

According to https://docs.sqlalchemy.org/en/14/changelog/migration_20.html#orm-query-get-method-moves-to-session

The Query.get() method remains for legacy purposes, but the primary interface is now the Session.get() method:

```
# legacy usage
user_obj = session.query(User).get(5)
```

Migration to 2.0

In 1.4 / 2.0, the Session object adds a new Session.get() method:

```
# 1.4 / 2.0 cross-compatible use
user_obj = session.get(User, 5)
```

In which compatible with flask_sqlalchemy, this should work well

```
user_obj = db.session.get(User, 5)
```

Share Edit Follow

answered Apr 10 at 8:50

Nguyễn Đức Huy

43 ● 4

This is a solution! Thanks! – Artem S. Zhelonkin May 23 at 12:40

Add a comment