Session: add and rollback

A set of operations such as addition, removal, or updating database entries is called a database transaction. A *database session* consists of one or more transactions. The act of committing ends a transaction by saving the transactions permanently to the database. In contrast, *rollback* rejects the pending transactions and changes are not permanently saved in the database.

In Flask-SQLAlchemy, a database is changed in the context of a session, which can be accessed as the `session` attribute of the database instance. An entry is added to a session with the `add()` method. The changes in a session are permanently written to a database when `.commit()` is executed.

For example, we create new readers and would like to add them to our database:

```
from app import db, Reader
new_reader1 = Reader(name = "Nova", surname = "Yeni", email =
"nova.yeni@example.com")
new_reader2 = Reader(name = "Nova", surname = "Yuni", email =
"nova.yeni@example.com")
new_reader3 = Reader( name = "Tom", surname = "Grey", email =
"tom.grey@example.edu")
```

Note that we didn't specify the primary key `id` value. Primary keys don't have to be specified explicitly, and the values are automatically generated after the transaction is committed.

Adding each new entry to the database has the same pattern:

```
db.session.add(new_reader1)
try:
    db.session.commit()
except:
    db.session.rollback()
```

Notice that we surrounded `db.session.commit()` with a try-except block. Why did we do that? If you look more carefully, `new_reader1` and `new_reader2` have the same e-mail, and when we declared the `Reader` model, we made the e-mail column unique (see the **app.py** file). As a consequence, we want to undo the most recent addition to the transaction by using `db.session.rollback()` and continue with other additions without interruption.

Instructions

1.

In the `playground.py` file, create a variable called `new_reader` that is assigned an instance of the `Reader` class with `name = “Peter”`, `surname = “Johnson”`, and `email =`peter.johnson@example.com. Do not add an assignment for `id`(the `id` will be automatically generated once the entry is added to the database)

2.

In the “playground.py” file, using `session.add()` add the `new_reader` entry to the database.

Stuck? Get a hint

3.

In the “playground.py” file, commit the `new_reader` object to the database. Enclose it using `try-except`. If `except` happens, perform the rollback.

```
from app import db, Reader #notice we import db here as well
import add_data #we use this script to recreate the database, put all the
entries so every time you run this script
                #you get the same result

#creating new readers
new_reader1 = Reader(name = "Nova", surname = "Yeni", email =
"nova.yeni@sample.com")
new_reader2 = Reader(name = "Nova", surname = "Yuni", email =
"nova.yeni@sample.com")
new_reader3 = Reader(name = "Tom", surname = "Grey", email =
"tom.grey@example.edu")

print("Before addition: ")
for reader in Reader.query.all():
  print(reader.id, reader.email)

print("\nNote that before committing, the id of the new readers is: ",
new_reader1.id, "\n")

#adding the first reader - the commit should succeed
db.session.add(new_reader1)
try:
    db.session.commit()
    print("Commit succeded.", new_reader1, "added to the database
permanently. The exception was not raised.\n")
except:
    db.session.rollback()
```

```
#adding the second reader - the commit should fail because e-mails should be
unique
db.session.add(new_reader2)
try:
    db.session.commit()
except Exception as ex:
    print("The commit of", new_reader2,"didn't succeed. Duplicate primary
key values. We will empty the current session.\n")
    print("The error is the following:", ex)
    db.session.rollback()

#adding the third reader - the commit should succeed
db.session.add(new_reader3)
try:
    db.session.commit()
    print("Commit succeded.", new_reader3, "added to the database
permanently. The exception was not raised.\n")
except Exception as ex:
    db.session.rollback()

print("\nNote that after committing, the id of the new readers is now: ",
new_reader1.id, "\n")

#print all the readers after the addition, and we see nova.yeni@sample.com
there, but not twice
for reader in Reader.query.all():
  print(reader.id, reader.email)
print("\nThe new readers Nova Yeni and Tom Grey are in the database. Notice
that Nova Yeni doesn't appear twice.\n")

print("\nCheckpoint 1: create a new_reader:")
new_reader = Reader(name='Peter', surname='Johnson',
email='peter.johnson@example.com')

print("\nCheckpoint 2: add the new reader to the database:")
db.session.add(new_reader)

print("\nCheckpoint 3: commit and rollback if exception is raised:")
try:
  db.session.commit()
except:
  db.session.rollback()
```

```
print("After addition: ")
for reader in Reader.query.all():
  print(reader.id, reader.email)
```