Queries and templates

In this exercise we combine database queries with Jinja templates. In the **routes.py** file, besides the `home()` route listing all the books in the database, we also added several routes for displaying web pages with filtered entries from the database. For example,

```
@app.route('/books/<year>')
def books(year):
    books = Book.query.filter_by(year = year)
    return render_template('display_books.html', year = year, books = books)
```

is a dynamic route with the `year` variable that can be set to some valid year in the URL. Next, we filter out the books from the asked year and display them using the **display_books.html** file (you can find in the templates folder). The template expects the provided `year` and a list of `books` to display.

In the URL bar of the integrated Web browser on the right, type or copy-paste the following: **http://localhost:8000/books/2020**. You see all the books suggested in 2020, together with all the reviews for each book. Cool, ha?

If the list of the retrieved queries is empty, we handle that in the **display_books.html** template by outputting that there are no books suggested in that year. Alternatively, one can use `.first_or_404()` as we do in the following:

```
reader = Reader.query.filter_by(id = user_id).first_or_404(description =
"There is no user with this ID.")
```

If a reader with some `id` does not exist in the database, 404 error is returned with a custom made message.

Take some time to observe the complete code provided on the right paying attention to **routes.py**, and explore the templates in the `template` folder.

Instructions

1.
In the `routes.py` file, in the `reviews(review_id)` method, just before rendering template with `render_template`, assign to a variable called `review` the result of a query that by using the `filter_by()` method filters the `Review` table by the value of the provided `review_id` parameter in the route.

Stuck? Get a hint

2.

Extend your query from Checkpoint1 to provide a custom `404 error` by using the `first_or_404()` and providing any description you would like.

```
from app import app
from app import db, Reader, Book, Review, Annotation
from flask import render_template, request, url_for, redirect

@app.route('/home')
@app.route('/')
def home():
  books = Book.query.all()
  return render_template('home.html', books = books)

@app.route('/profile/<int:user_id>')
def profile(user_id):
   reader = Reader.query.filter_by(id = user_id).first_or_404(description =
"There is no user with this ID.")
   return render_template('profile.html', reader = reader)

@app.route('/books/<year>')
def books(year):
   books = Book.query.filter_by(year = year)
   return render_template('display_books.html', year = year, books = books)

@app.route('/reviews/<int:review_id>')
def reviews(review_id):
   review = Review.query.filter_by(id = review_id).first_or_404(description
= "There is no review with this ID.")
   return render_template('_review.html', review = review)
```