Previous Next

Using Query Strings in an API

Estimated Time: 1-2 hours

Using Query Strings in an API

In this assignment you'll be extending the GET endpoint for posts to carry out simple queries. This code will allow you to find all of the posts with a title containing a particular string.

In order to perform the query you'll use query strings. Query strings are a part of a URL designed to contain additional information which will be passed to the server in a request. For example, take a look at the following URL:

```
http://example.com/message?from=Alice&to=Bob
```

Here the query string is the part which says `?from=Alice&to=Bob`. This is split into two pairs of keys and values: one saying that the message is from Alice, and one saying that the message should be sent to Bob.

Test first

Now that you know how query strings work, you can set up a test which will try to query the `/api/posts` endpoint to find posts which have a title containing the string "whistles". Try adding the following code to the `TestAPI` class in *tests/api_tests.py*.

```
    def test_get_posts_with_title(self):
        """ Filtering posts by title """
        postA = models.Post(title="Post with bells", body="Just a test
        postB = models.Post(title="Post with whistles", body="Still a
        postC = models.Post(title="Post with bells and whistles",
                            body="Another test")

        session.add_all([postA, postB, postC])
        session.commit()

        response = self.client.get("/api/posts?title_like=whistles",
            headers=[("Accept", "application/json")]
        )

        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.mimetype, "application/json")

        posts = json.loads(response.data.decode("ascii"))
        self.assertEqual(len(posts), 2)

        post = posts[0]
        self.assertEqual(post["title"], "Post with whistles")
        self.assertEqual(post["body"], "Still a test")

        post = posts[1]
        self.assertEqual(post["title"], "Post with bells and whistles"
        self.assertEqual(post["body"], "Another test")
```

This should all be pretty familiar from the previous assignment. First you add three posts to the database, two of which contain the word "whistles". Then you make a GET request to the endpoint, this time adding in your query string `?title_like=whistles`. Then check the response status and mimetype and make sure that the correct two posts have been returned.

Try running the test using `nosetests tests`. You should see that the endpoint returns a 200 status but is ignoring your query string and giving back all three posts.

Making it pass

Now try to change your endpoint to access the `title_like` value from the query string, and alter the database query so it will filter posts which don't match the string:

```
@app.route("/api/posts", methods=["GET"])
@decorators.accept("application/json")
def posts_get():
    """ Get a list of posts """
    # Get the querystring arguments
    title_like = request.args.get("title_like")

    # Get and filter the posts from the database
    posts = session.query(models.Post)
    if title_like:
        posts = posts.filter(models.Post.title.contains(title_like))
    posts = posts.order_by(models.Post.id)

    # Convert the posts to JSON and return a response
    data = json.dumps([post.as_dictionary() for post in posts])
    return Response(data, 200, mimetype="application/json")
```

Here you're using the `request.args.get` function to retrieve the value from your query string. If there is no `title_like` key in the string this function will return `None`.

You then construct our query in a couple of steps. First you construct a `Query` object without actually hitting the database. If the query string contained a `title_like` key you then add a filter, using the `contains` method to find titles which contain our string. Finally, the query is executed as you iterate over it in the list comprehension.

Try running your tests again to make sure that your code correctly returns the filtered posts. With this in place you now have a more powerful retrieval API which allows you to perform simple searches on our posts.

Try It!

Try adding a query for body text to your API. Ideally you should be able to combine the two query types, so for example a request to `/api/posts?title_like=whistles&body_like=bells` should only return posts which have both whistles in the title and bells in the body.

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

example.com/project

✔ Submit your project

Previous Next