Flask Python (7) – Adding and deleting Data

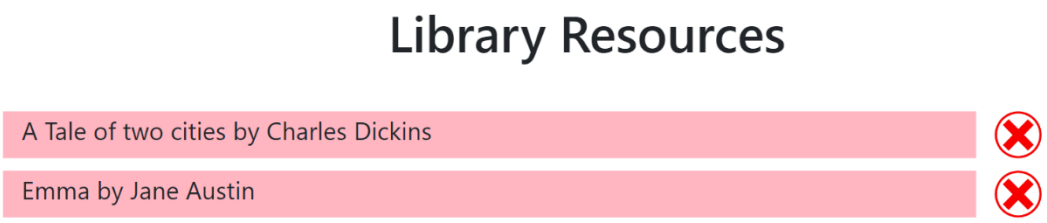
Deleting Books

Step 1: The *library.html* lists the books in our database but supposing that we wanted to delete them. Let's change the code on the template so that it has a red cross which when clicked will delete the book from the database and then refresh the page.

```
{% for e in all_books %}
<div class="col-10">{{e[1]}} by {{e[2]}} </div>
<div class="col-2">
<a class="button" href="((url_for('del_book', id=e[0])}}"> x </a>
</div>
```

Once you have done some Bootstrap styling your page should look like the picture below. You will need to download a free red cross image and replace the x with something like:

```
<img height=30px src="/static/images/red-cross.png" />
```



This creates a button which when clicked uses the `url_for()` function and goes to the route:

```
del_book/isbn
```

The ISBN number is passed as a parameter. We will look at the code for the route `del_book` later.

Step 2: When we click the cross, we want to delete that book from the database. To do this we need to provide an SQL query to delete the book. This goes in our *model.py* file and create that function.

```
def del_books(isbn):
    mydb = create_connection
    mycursor = mydb.cursor()
    sql_txt = "DELETE FROM T_books WHERE ISBN = '" + isbn + "'"
    mycursor.execute(sql_txt)
    mydb,commit()
    close_connection(mydb)
```

All we need to do is provide the isbn number of our book and it will be deleted from the database.

Step 3: Our `url_for()` routes to `del_book`. We can add this to our *flask_app.py* file:

```
@app.route('/del_book/<id>')
def del_book(id):
        del_books(id)
        return redirect(url_for('library'))
```

This calls the `del_books(id)` function which we declared in *model.py* file earlier. Once we have deleted the book, we return to the library route and the page will be refreshed.

In order to use the `del_books()` function from *models.py* we will need to import it. We do this as we did with the `add_books()` function in the previous task:

```
from library.models import get_books
```

Adding Books

Step 1: In order to add books you will need to hold the data. Create a class like we did when we looked at the SignUp form. Here's the code in the *form.py* file:

```
class AddBook(FlaskForm):
    isbn = StringField('ISBN')
    title = StringField('Book Title')
    author = StringField('Author')
    submit = SubmitField('Add the Book')
```

Step 2: We can now create a form on our *library.html* page underneath the list of books:

```
<h3> Add a Book </h3>
<form method="POST">
{{form.hidden_tag()}}
<div class="row p-2">
    <div class="col-sm-5">{{form.isbn.label}}</div>
    <div class="col-sm-5">{{form.isbn}}</div>
</div>
<div class="row p-2">
    <div class="col-sm-5">{{form.title.label}}</div>
    <div class="col-sm-5">{{form.title}}</div>
</div>
<div class="row p-2">
    <div class="col-sm-5">{{form.author.label}}</div>
    <div class="col-sm-5">{{form.author}}</div>
</div>
<div class="row p-3">
    {{form.submit}}
```

```
</div>
    </form>
```

The `<form>` element lets us collect data and pass it to a python file. When the form is submitted, the data from the form is passed to the python code using the POST method – we have included that in our route. The GET method we will look at later.

Step 3: In order for this to work we just pass a form object to the library.html page. Here is the updated code from the *flask_app.py file*.

```
@app.route('/library',methods=["GET","POST"])
def library():
    form = AddBook()
    if request.method == "POST":
        add_book(form.isbn.data, form.title.data, form.author.data)
    all_books = get_books()
    return render_template('library.html', all_books=all_books, form=form)
```

Create from which is an object of the class `AddBook`. We pass this to the library template so our form will be rendered. There is also some code which tells us to add the book if a valid form has been submitted and it uses the data from the fields on the form (e.g. `form.isbn.data` is the ISBN number entered into the form. Note that we used the `add_book()` function we defined in the model.py in a previous section- see data.

Makes sure you have the flask `request` module imported into flask_app.py file and also the `add_book` function from *model.py*:

```
from flask import Flask, render_template, url_for, redirect,request
from library.models import get_books,del_books,add_book
```

Step 4: Check your library resources page appears as below and works: