INFO3180 - LECTURE 4

FORMS CONT'D AND DATABASES

GETTING FORM DATA IN FLASK

The key thing is the `request` *object.*

You might remember that file upload data is stored in the `request.file` *object.*

Form data is stored in the `request.form` *object.*

EXAMPLE OF GETTING FORM DATA FROM A FORM FIELD

This field:

```
<input type="text" name="username" />
```

Can be retrieved via:

```
request.form['username']
```

FLASK-WTF

Flask-WTF provides simple integration between Flask and a python library called WTForms.

FEATURES OF FLASK-WTF

- Integration with WTForms.
- Secure Form with CSRF token.
- Global CSRF protection.
- Provides some built-in validators to help with form validation.
- reCAPTCHA support.
- File upload that works with Flask-Uploads.
- Internationalization using Flask-Babel.

When you are working with WTForms you have to define your forms as classes first.

EXAMPLE OF A FORM CLASS

```
from flask_wtf import FlaskForm
from wtforms import StringField
from wtforms.validators import InputRequired

class MyForm(FlaskForm):
    name = StringField('name',
validators=[InputRequired()])
```

EXAMPLE OF FORM VALIDATION

```
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    form = MyForm()
    if form.validate_on_submit():
        return redirect('/success')
    return render_template('submit.html',
form=form)
```

EXAMPLE OF A FORM RENDERED IN A TEMPLATE

```
<form method="POST" action="/process">
    {{ form.csrf_token }}
    {{ form.name.label }}
    {{ form.name(size=20) }}
    <button type="submit">Go</button>
</form>
```

DATABASES

QUICK REFRESHER ON DATABASES

- SQL stands for Structured Query Language
- It allows you to execute queries that retrieve, insert, update and delete data. Also you can create new databases, new tables, etc.
- A database contains tables and data is stored in those tables. A table is a collection of related data which consist of columns (fields/attributes) and rows (records).
- Some types of Relational Database Management Systems (RDMS) are SQLite, PostgreSQL, MySQL, Oracle, MS-SQL, Firebird, Sybase and others.

Database

Column

Table

Row

id	Name	Email	Gender
1	John Brown	jbrown@example.com	Male
2	Mary Jane	mjane@example.com	Female
3	Peter Parker	spiderman@example.com	Male
4	Tammy Chin	tchin@example.com	Female

EXAMPLE OF USING SQLITE DATABASE

```
DROP TABLE IF EXISTS users;

CREATE TABLE users (
  id integer primary key autoincrement,
  name string not null,
  email string not null
);
```

EXAMPLE OF USING SQLITE DATABASE

We'll use sqlite3 for our examples. Assuming we have sqlite3 installed and the previous slide represented a file called schema.sql, to initialize an sqlite3 database we could run the following command:

```
sqlite3 mydatabase.db < schema.sql
```

EXAMPLE OF USING SQLITE DATABASE IN PYTHON

```
import sqlite3
def connect_db():
    return sqlite3.connect('./mydatabase.db')

name = raw_input('Enter your name please: ')
email = raw_input('and your email address: ')

db = connect_db()
db.execute('insert into users (name, email) values
(?, ?)', [name, email])

db.commit()
cur = db.execute('select name, email from users order
by id desc')
print [dict(name=row[0], email=row[1]) for row in
cur.fetchall()]
```

How would we do this in Flask?

Instead of using `raw_input()` *you would create a form to accept user input. And you would then have your respective routes to display and process the form data.*

EXAMPLE OF ADDING A USER

```
@app.route('/add-user', methods=['POST', 'GET'])
def add_user():
    if request.method == "POST":
        db = connect_db()
        db.execute('insert into users (name, email)
values (?, ?)', [request.form['name'],
request.form['email']])

        db.commit()
        flash('New user was successfully added')
        return redirect(url_for('show_users'))

    return render_template('add_user.html')
```

EXAMPLE OF QUERYING TO GET A LIST OF USERS

```
@app.route('/users')
def show_users():
    db = connect_db()
    cur = db.execute('select name, email from users
order by id desc')
    users = cur.fetchall()

    return render_template('show_users.html',
users=users)
```

FLASK-SQLALCHEMY

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper (ORM) that gives application developers the full power and flexibility of SQL.

Flask-SQLAlchemy is an extension for Flask that adds support for the python library SQLAlchemy to your application.

What is an ORM?

ORMs allow database applications to work with objects instead of tables and SQL. The operations performed on the objects are translated into database commands transparently by the ORM.

EXAMPLE OF CHANGES TO YOUR __INIT__.PY

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SECRET_KEY'] = 'some$3cretKey'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:////
tmp/mydatabase.db'

db = SQLAlchemy(app)

from app import views, models
```

EXAMPLES OF CONNECTING TO OTHER DBMS

General format: driver://username:password@server/db

PostgreSQL: postgresql://username:password@localhost/mydatabase

MySQL: mysql://username:password@localhost/mydatabase

EXAMPLE OF A SIMPLE MODEL

```
from . import db

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    email = db.Column(db.String(120), unique=True)

    def __init__(self, username, email):
        self.username = username
        self.email = email

    def __repr__(self):
        return '<User %r>' % self.username
```

As a convention we will store all our Model Classes in a file called **models.py** in our **app** folder

EXAMPLE OF USING A MODEL IN YOUR ROUTE FUNCTIONS

```
# do your usual imports for flask but then add these
from app import db
from app.models import User

@app.route('/users')
def show_users():
    users = db.session.query(User).all()
    return render_template('show_users.html',
users=users)
```

EXAMPLE OF SAVING DATA TO DATABASE WITH SQLALCHEMY

```
# do your usual imports for flask but then add these
from app import db
from app.models import User

@app.route('/add-user', methods=['POST'])
def add_user():
    user = User(request.form['username',
request.form['email'])
    db.session.add(user)
    db.session.commit()
    flash('New user was successfully added')
    return redirect(url_for('show_users'))
```

Another good thing about using an ORM like SQLAlchemy is that whenever you need to change your DBMS system, you don't need to change any of your backend code.

In future labs, tutorials and projects we will use PostgreSQL instead of SQLite. We will also look at how we can make changes to our database using Migrations.

RESOURCES

- Flask-SQLAlchemy - http://flask-sqlalchemy.pocoo.org/2.1/
- Flask-WTF - https://flask-wtf.readthedocs.io/en/stable/

DEMO