Advanced Form Validation

Another common validation that needs to be performed is checking whether an email address is valid. You can achieve this by using something known as a regular expression or regex.

A regex is a sequence of characters that defines a search pattern. It can be used to match a string that follows a pattern. For example, every email has a series of alphanumeric characters followed by an @ symbol followed by another series of alphanumeric characters followed by a "." and finally another series of alphanumeric characters. You don't need to know how to create regex at this point, but understanding what they are and what they are used for is definitely important. The Python regex for matching an email address based on the above criteria looks something like this:

```
r'^[a-zA-Z0-9.+_-]+@[a-zA-Z0-9._-]+\.[a-zA-Z]+$'
```

Let's create a simple application similar to the basic_validation example where we can input an email and check if it is valid!

Let's start with a basic index.html page:

/advanced_validation/templates/index.html

```
  Advanced Validation Example
  Enter a Valid(Any) Email!
  {% with messages = get_flashed_messages() %}
    {% if messages %}
      {% for message in messages %}
        {{message}}
      {% endfor %}
    {% endif %}
  {% endwith %}

    Email:
```

And don't forget our server.py!

/advanced_validation/server.py

```
# import Flask
from flask import Flask, render_template, redirect, request, session, flash
# the "re" module will let us perform some regular expression operations
import re
# create a regular expression object that we can use run operations on
EMAIL_REGEX = re.compile(r'^[a-zA-Z0-9.+_-]+@[a-zA-Z0-9._-]+\.[a-zA-Z]+$')
app = Flask(__name__)
app.secret_key = "ThisIsSecret!"
@app.route('/', methods=['GET'])
def index():
  return render_template("index.html")
@app.route('/process', methods=['POST'])
```

```
def submit():
    if len(request.form['email']) < 1:
        flash("Email cannot be blank!")
    # else if email doesn't match regular expression display an "invalid email address" messag
    else:
        flash("Success!")
    return redirect('/')
app.run(debug=True)
```

Now let's modify our validation conditional to include a case for an invalid email!

```
@app.route('/process', methods=['POST'])
def submit():
    if len(request.form['email']) < 1:
        flash("Email cannot be blank!")
    elif not EMAIL_REGEX.match(request.form['email']):
        flash("Invalid Email Address!")
    else:
        flash("Success!")
    return redirect('/')
```

Let's see what is going on here:

- We are using if/else statements just like we did before to perform validations
- We are using the EMAIL_REGEX object that we created and running the .match() method that will return *None* if no match can be found. If the argument matches the regular expression, a match object instance is returned.

Other Useful Validation Tools:

str.isalpha() -- Returns a boolean that shows whether a string contains only alphabetic characters. Documentation here.

Other string methods that may be useful can be found in the Python Docs here.

time.strptime(string, format) -- Changes a string to a time using the given format. Documentation here.