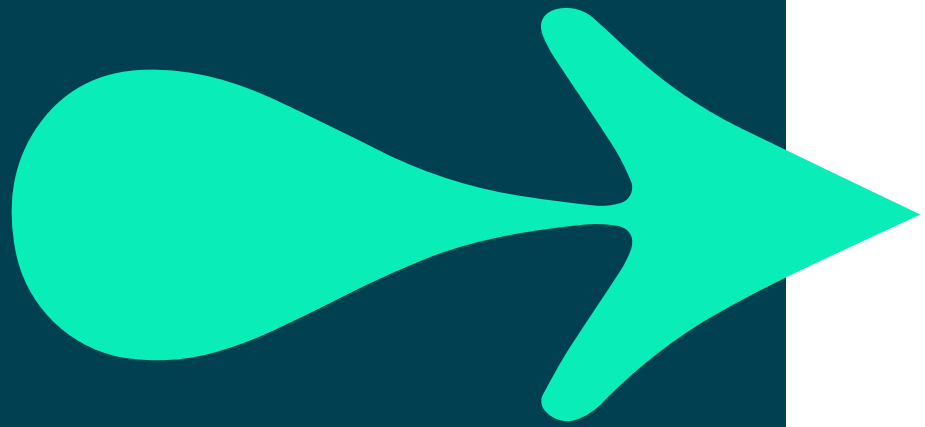
POWERING POTENTIAL

Flask Part 2

Module 21

MODULE OVERVIEW



Contents

- Jinja 2 Templates
- HTML Forms
- Database queries

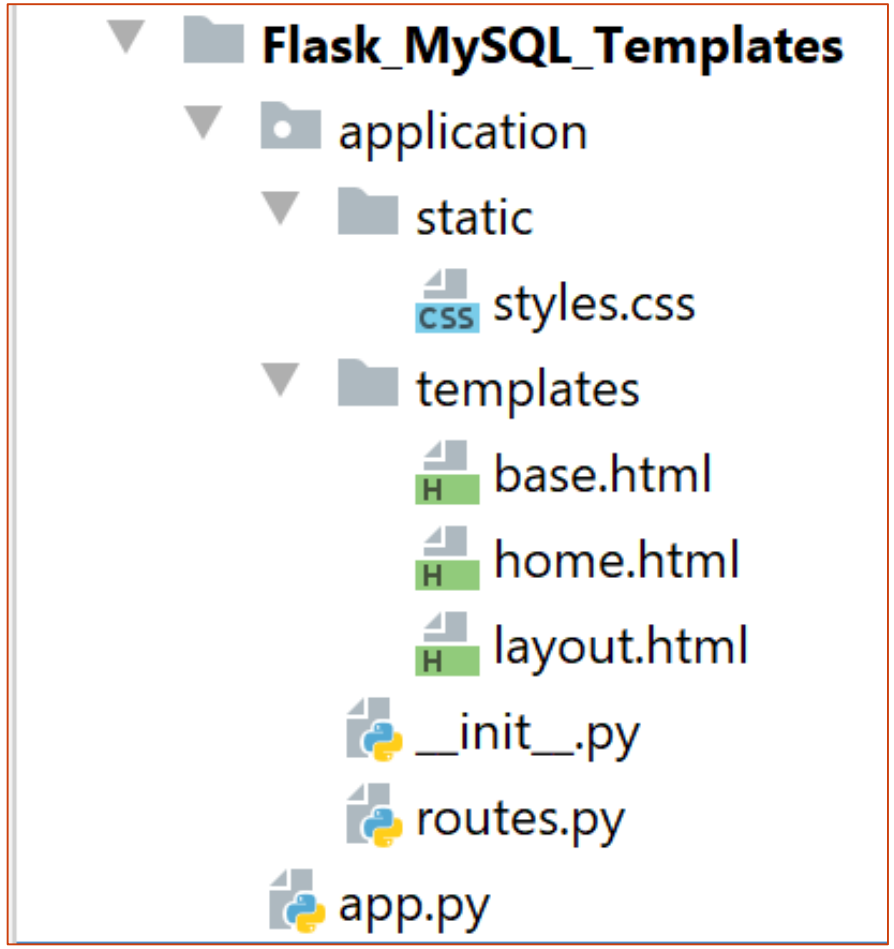
render_template() function

Instead of constructing HTML as a large string to return as a response you can create HTML templates and return these using the **render_template()** function

You can include code within the HTML to evaluate variables, repeat logic and render output conditionally

```
1   from flask import render_template
2
3   from application import app
4
5   @app.route('/')
6   @app.route('/home')
7   def home():
8       return render_template('home.html', title='Home')
9
```

Flask Project Structure



Example Files

app.py

```
from application import app

if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

routes.py

```
from flask import render_template

from application import app

@app.route('/')
@app.route('/home')
def home():
    return render_template('home.html', title='Home')

# use base html template
@app.route("/welcome/<name>/")
def template_base(name):
    return render_template('base.html', name=name, group="Everyone")
```

__init__.py

```
# import Flask class from the flask module
from flask import Flask

# create a new instance of Flask and store it in app
app = Flask(__name__)

# import the ./application/routes.py file
from application import routes
```

Template Examples

A Simple Jinja Template:

base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Templates</title>
</head>
<body>
    <h1>Welcome to Flask and Jinja</h1>
    <p>Hello {{name}}</p>
    <p>Welcome {{group}}</p>
</body>
</html>

```

```
# use base html template
@app.route("/welcome/<name>/")
def template_base(name):
    return render_template('base.html', name=name, group="Everyone")
```

Inheriting Templates

layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{{title}} - Page</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <h1>{{ title }} Page</h1>
{% block body_content %}{% endblock %}
</body>
</html>
```

home.html

```
{% extends "layout.html" %}
{% block body_content %}
    <p> Welcome to my first inherited template</p>
{% endblock %}

```

```
from flask import render_template

from application import app

@app.route('/')
@app.route('/home')
def home():
    return render_template('home.html', title='Home')
```

Activity:

Restructure your application files to the recommended structure

Create a **layout.html** file in the templates folder

Create an inherited template called **about.html** that extends that layout template

Change your **About** route to render this template

Jinja IF Statements

```
{% if 3 == 4 %}

<h1>3 is equal to 4</h1>

{% else %}

<h1>3 is not equal to 4</h1>

{% endif %}
```

Jinja FOR Loops

```
<p>Loop through the list:</p>
 <ul>
   {% for item in my_list %}
   <li>{{ item }}</li>
   {% endfor %}
 </ul>
```

Activity:

Create an inherited template called **favoutrites.html** that extends that **layout** template

Create a favourite **route** and pass a *list* of your favourite things to the **render_template** function to be displayed within the favourites.html template

Flask Forms

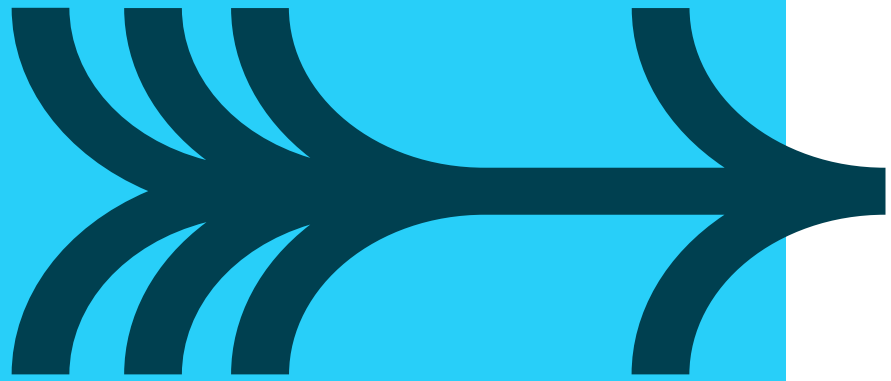
Form Field Types

Field types	Description	Optional Arguments	Example
StringField	A field which takes in string input.	size=(), maxlength=()	name = StringField('Name', maxlength=20)
IntegerField	A field which takes in integer input.	size=(), maxlength=()	number = IntegerField('', size=20)
BooleanField	A field which takes in true or false input.	false_values=None	bool = BooleanField()
DateField	A field which takes in date inputs.	format='%Y-%m-%d'	date = DateField()
DateTimeField	A field which takes in date and time inputs	format='%Y-%m-%d %H:%M:%S'	datetime = DateTimeField()
DecimalField	A field which takes in decimal input.	places=2, rounding=None	decimal = DecimalField()
SubmitField	A field which allows for checking of a given submit button being pressed.	none	input type="submit"
SelectField	A field which allows us to make use of the choices parameter which is simply a list of value and label pairs. This allows us to give users a list of options to interact with on the webpage.	choices=[], validate_choice=True or False	language = SelectField('Programming Language', choices=[('cpp', 'C++'), ('py', 'Python'), ('text', 'Plain Text')])

Form Field Attributes

Attribute	Example	Description
data	firstname.data	The data submitted for a field named firstname.
label	firstname.label	The label of a field named firstname.
errors	firstname.errors	Any errors thrown at submission.
type	firstname.type	The type of field firstname belongs to, such as StringField.

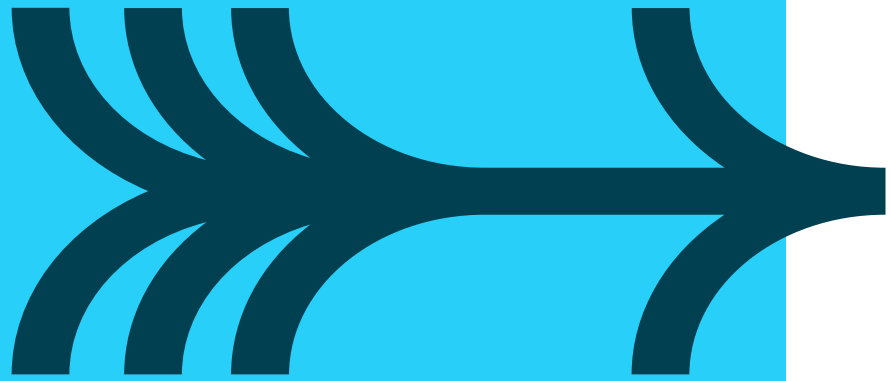
INSTRUCTOR WALKTHROUGH



home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Simple Form</title>
</head>
<body>
<div class='form'>
            <form method='POST' action=''>
                {{ form.hidden_tag() }}
                {{ form.first_name.label }} {{ form.first_name }}
                <br>
                {{ form.last_name.label }} {{ form.last_name }}
                <br>
                {{ form.submit }}
            {{ message }}
            </form>
        </div>
</body>
</html>
```

FORM AND ROUTE



forms.py

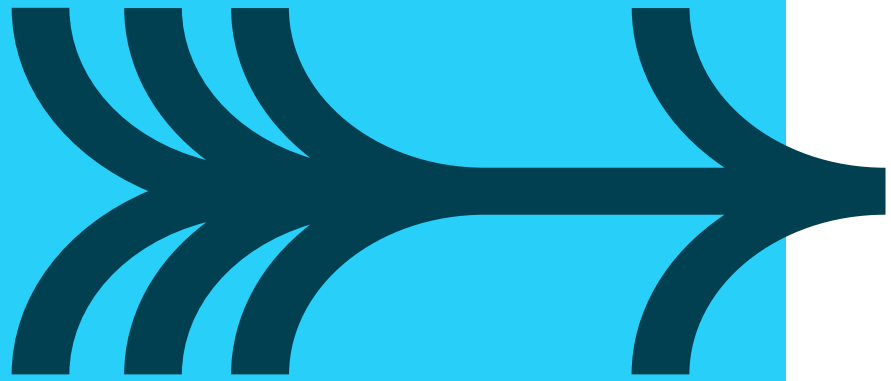
```python
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField


class BasicForm(FlaskForm):
    first_name = StringField('First Name')
    last_name = StringField('Last Name')
    submit = SubmitField('Add Name')
```

routes.py

```python
from flask import render_template, request
from application import app
from application.forms import BasicForm


@app.route('/', methods=['GET', 'POST'])
@app.route('/home', methods=['GET', 'POST'])
def register():
    error = ""
    form = BasicForm()

    if request.method == 'POST':
        first_name = form.first_name.data
        last_name = form.last_name.data

        if len(first_name) == 0 or len(last_name) == 0:
            error = "Please supply both first and last name"
        else:
            return 'Thank you!'

    return render_template('home.html', form=form, message=error)
```

Database Queries

INSTRUCTOR WALKTHROUGH

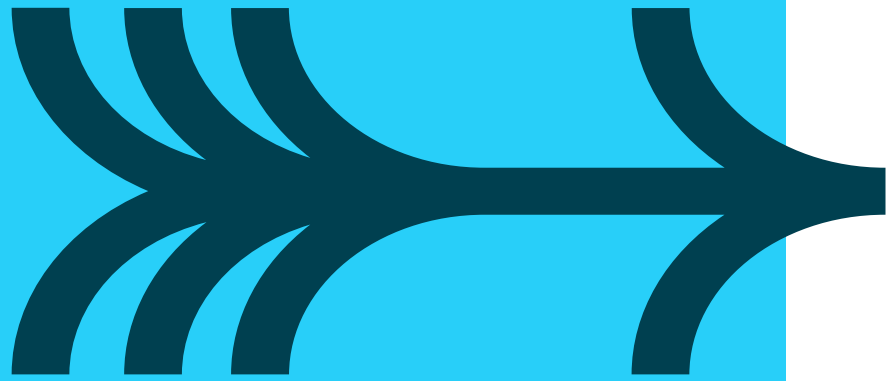


- Create
- Read
  - Queries: all(), first(), filter_by(), get(), order_by(), limit(), count()
- Update
- Delete

- Models
- SQLAlchemy configuration
- db.create_all()
- db.drop_all()
- Model / Table Relationships

INSTRUCTOR WALKTHROUGH

- validate_on_submit()
- db.session.add()
- db.session.commit()
- db.session.delete()
- Combining GET and POST on the same route
- redirect()



Any questions?