SEO Tech Developer

Web Frameworks and Forms

What you will be able to do:

- Define web framework
- Build a simple Flask webpage
- Create a simple template
- Create a website form
- Validate user input with clear error messages

SEO Tech Developer

SEO Tech Developer

Web Framework

Web Framework

- As we learned, a framework has many related libraries and APIs

- A **web framework** is a framework for web development
  - Examples: Django, Flask, Bottle, etc.

- We will be using Flask because it is lightweight with lots of integrations
  - This makes it quick to learn and highly customizable!

SEO Tech Developer

Hello World in Flask (in .py file)

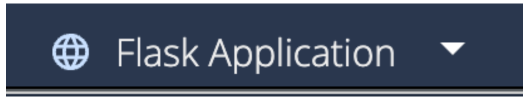
```
from flask import Flask
# gets name of the .py file so Flask knows it's name
app = Flask(__name__)

# tells you the URL the method below is related to
@app.route("/")
def hello_world():
    # prints HTML to the webpage
    return "<p>Hello, World!</p>"
```

SEO Tech Developer

Running Flask in Codio (option 1)

- Set an environment variable:
  - `export FLASK_APP=file_name_without_extension`
  - e.g. if your file is `hello.py`, run `export FLASK_APP=hello`
- Start the Flask server by running:
  - `flask run --host=0.0.0.0`
  - Stop server using CTRL + C
- Click "Flask Application" in the top menu bar to see the webpage
  - If you see 404, check that your server is running

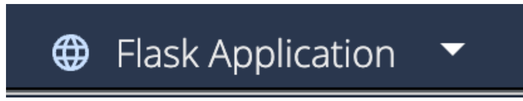


SEO Tech Developer

Running Flask in Codio (option 2)

- Add the following to the end of your python file:

  -
    ```
    if __name__ == '__main__':
        app.run(host="0.0.0.0")
    ```
- Start the Flask server using normal python command:
  - `python3 file_name.py`
  - Stop server using CTRL + C
- Click "Flask Application" in the top menu bar to see the webpage
  - If you see 404, check that your server is running



SEO Tech Developer

Making Changes

- If you make a change with the server running and re-load the webpage, you'll notice the change isn't propagated. Either:
  - Re-start the server - stop it using CTRL+C and re-start it
  - Run your server in DEBUG mode

- To run your server in DEBUG, change the last line of code to:
  - `app.run(debug=True, host="0.0.0.0")`

SEO Tech Developer

Adding a page to your website

- Set the URL – for example website.com/about is normal:
  - `@app.route("/about")`

- Make function that does something on that page:

  -
    ```
    def about():
        return "<p>About us!</p>"
    ```

- View your new page by:
  - Starting your Flask server
  - Open your website
  - Add `/about` to the URL

SEO Tech Developer

SEO Tech Developer

Templates

HTML template files

- Instead of having to squeeze a bunch of HTML into your python file, you can use templates which are HTML files

- Create a `templates` folder/directory.
  - Inside it, create 3 files:
    - `layout.html`
    - `home.html`
    - `about.html`

SEO Tech Developer

Jinja

- Jinja is what powers templates in flask -- allowing us to do things like leave placeholders called Blocks to be filled later.

- Jinja also helps inject python into HTML pages. Some basic and useful things:
  - Variables
  - if statements
  - for loops

SEO Tech Developer

layout.html

- Layout holds the general layout so you don’t need to copy-paste to every other page

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
    <h1>This shows up on every page</h1>

    {% block content %}{% endblock %}
</body>
</html>
```

Jinja block

home.html and about.html

```
{% extends "layout.html" %}
{% block content %}
    <h2>{{ subtitle }}</h2>
{% endblock content %}
```

Notes about the above code:

1. `{% extends "layout.html" %}` - says this inherits from our layout.html file
2. `{% block content %}...{% endblock content %}` - specifies the part of the layout.html template we are filling in
3. `<h2>{{ subtitle }}</h2>` - says that we are expecting something called subtitle we want to print

SEO Tech Developer

Rendering templates

Update your python file in the following ways:

1. Import `render_template`
2. Call `render_template` instead of returning raw HTML

```
from flask import Flask, render_template
app = Flask(__name__)

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

SEO Tech Developer

SEO Tech Developer

Forms

WTForms

- WTForms is a python library for forms validation and rendering
- Forms are made up of Fields with Validators
    - Fields include basic form field types
        - `StringField` – field to input text
        - `PasswordField` – like StringField but value is not rendered back to browser
        - `SubmitField` - allows checking if a given submit button has been pressed
    - Validators check user input
        - `DataRequired` - sets the required flag on fields it is used on
        - `Length` - Validates the length of a string
        - `Email` - Validates an email address using email_validator package
        - `EqualTo` – Compares the values of two fields - used to facilitate the password change form
- Flask-WTF is a library that integrated WTForms with Flask

SEO Tech Developer

Secrets

To protect our website form from bad cookies, we need to set a secret key.

1. Pop open a python interpreter and generate a 16 byte token -- copy it without the quotes
```
import secrets
secrets.token_hex(16)
```

2. In `hello.py`, after `app = Flask(__name__)` add...
```
app.config['SECRET_KEY'] = 'the key you generated'
```

SEO Tech Developer

Create Forms.py

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired, Length, Email, EqualTo

class RegistrationForm(FlaskForm):
        username = StringField('Username’,
                          validators=[DataRequired(), Length(min=2, max=20)])
        email = StringField('Email’, validators=[DataRequired(), Email()])
        password = PasswordField('Password', validators=[DataRequired()])
        confirm_password = PasswordField('Confirm Password’,
                          validators=[DataRequired(), EqualTo('password')])
        submit = SubmitField('Sign Up')
```

SEO Tech Developer

Add Form to Website (hello.py)

- Make sure your Flask app can see your form by importing it in `hello.py`:

```
from forms import RegistrationForm
```

- Add a new page to your site:

```
@app.route("/register", methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    return render_template('register.html', title='Register', form=form)
```

SEO Tech Developer

Simple register.html

Sends information to form

Passes your SECRET key

Adds username field

Adds email field

Adds password field

Adds confirm password field

Adds Button

```
{% extends "layout.html" %}

{% block content %}

    <div class="content-section">

        <form method="POST" action="">

            {{ form.hidden_tag() }}

            <fieldset class="form-group">

                <legend class="border-bottom mb-4">Join Today</legend>

                <div class="form-group">

                    {{ form.username.label(class="form-control-label") }}

                    {{ form.username(class="form-control form-control-lg") }}

                </div>

                <div class="form-group">

                    {{ form.email.label(class="form-control-label") }}

                    {{ form.email(class="form-control form-control-lg") }}

                </div>

                <div class="form-group">

                    {{ form.password.label(class="form-control-label") }}

                    {{ form.password(class="form-control form-control-lg") }}

                </div>

                <div class="form-group">

                    {{ form.confirm_password.label(class="form-control-label") }}

                    {{ form.confirm_password(class="form-control form-control-lg") }}

                </div>

            </fieldset>

         <div class="form-group">{{ form.submit(class="btn btn-outline-info") }} </div>

        </form></div>{% endblock content %}
```

SEO Tech Developer

Passing Success Message (hello.py)

- After you instantiate form (`form = RegistrationForm()`) in `hello.py`:

```
  if form.validate_on_submit():
          flash(f'Account created for {form.username.data}!', 'success')
          return redirect(url_for('home'))
```

- The f-string allows us to insert a variable -- in this case the data filled in by the user in the username field.
- The \`success\` part tells bootstrap the type of message we are sending so it style accordingly.
- Finally, because it was validated and the account was "created" we can send the user to the home screen.

- To get the re-direct to work properly in Codio, we need to add a little code.

```
from flask import Flask, render_template, url_for, flash, redirect
from flask_behind_proxy import FlaskBehindProxy
    app = Flask(__name__)
    proxied = FlaskBehindProxy(app)
```

SEO Tech Developer

Passing Success Message (layout.html)

```
<h1>This shows up on every page</h1>
{% with messages = get_flashed_messages(with_categories=true) %}
  {% if messages %}
    {% for category, message in messages %}
       <div class="alert alert-{{ category }}">
         {{ message }}
      </div>
    {% endfor %}
  {% endif %}
{% endwith %}

{% block content %}{% endblock %}
```

SEO Tech Developer

Showing Validation Errors (register.html)

```
{% extends "layout.html" %}
{% block content %}
    <div class="content-section">
        <form method="POST" action="">
            {{ form.hidden_tag() }}
            <fieldset class="form-group">
              <div class="form-group">
                  {{ form.username.label(class="form-control-label") }}
                  {% if form.username.errors %}
                    {{ form.username(class="form-control form-control-lg is-invalid") }}
                    <div class="invalid-feedback">
                      {% for error in form.username.errors%}
                          <span>{{error}}</span>
                      {% endfor %}
                    </div>
                  {% else %}
                    {{ form.username(class="form-control form-control-lg") }}
                  {% endif %}
              </div>
```

Passes data validation errors to user

SEO Tech Developer

What questions do you have about…

- Defining web framework
- Building a simple Flask webpage
- Creating a simple template
- Creating a website form
- Validating user input with clear error messages

SEO Tech Developer

SEO Tech Developer

Thank you!