

Web Programming

Server-side programming I.

Recall from Last Time

- HTTP requests
 - GET, POST
- HTTP responses

Today

- Making a simple server-side application
 - Serve static content (css files, html files, images, etc.)
 - Generate dynamic content
 - Handle input from URL and forms

Flask

- Framework for server-side web application development
- Included in the Anaconda distribution (i.e., no need to install anything)
- Others `python -m pip install Flask`
- Practical notes
 - Put each application in a separate folder (even if it consists of a single file)
 - This is also how the examples on GitHub are structured (`examples/python/flask`)
 - Don't call your application `flask` (that would conflict with Flask itself)
- <http://flask.pocoo.org/>

A Minimal Web Application

 [examples/python/flask/0_minimal/app.py](https://github.com/realpython/examples/blob/master/python/flask/0_minimal/app.py)

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

@app.route('/')
def hello_world():
    return("Hello, World!")

if __name__ == "__main__":
    app.run()
```

A Minimal Web Application

📄 [examples/python/flask/0_minimal/app.py](#)

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

@app.route('/')
def hello_world():
    return "Hello, World!"

if __name__ == "__main__":
    app.run()
```

Create an instance of the Flask class
(the argument is the name of the module)

A Minimal Web Application

📄 [examples/python/flask/0_minimal/app.py](#)

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

```
@app.route("/")  
def hello_world():  
    return "Hello, world!"
```

The **app.route** decorator tells Flask which URL should trigger this function

```
if __name__ == "__main__":  
    app.run()
```

Exercise #0



[https://github.com/dat310-2023/info/tree/master/](https://github.com/dat310-2023/info/tree/master/exercises/python/flask1)
exercises/python/flask1

Routing

📄 [examples/python/flask/1_routing/app.py](#)

- The **route()** decorator is used to bind a function to a URL
- Add variable parts to the URL by marking them as **<varname>**

```
@app.route("/")
def index():
    return "Index Page"

@app.route("/info")
def hello():
    return "This is a static info page"

@app.route("/user/<username>")
def user(username):
    return "Showing the profile page for user {}".format(username)
```

Variable rules

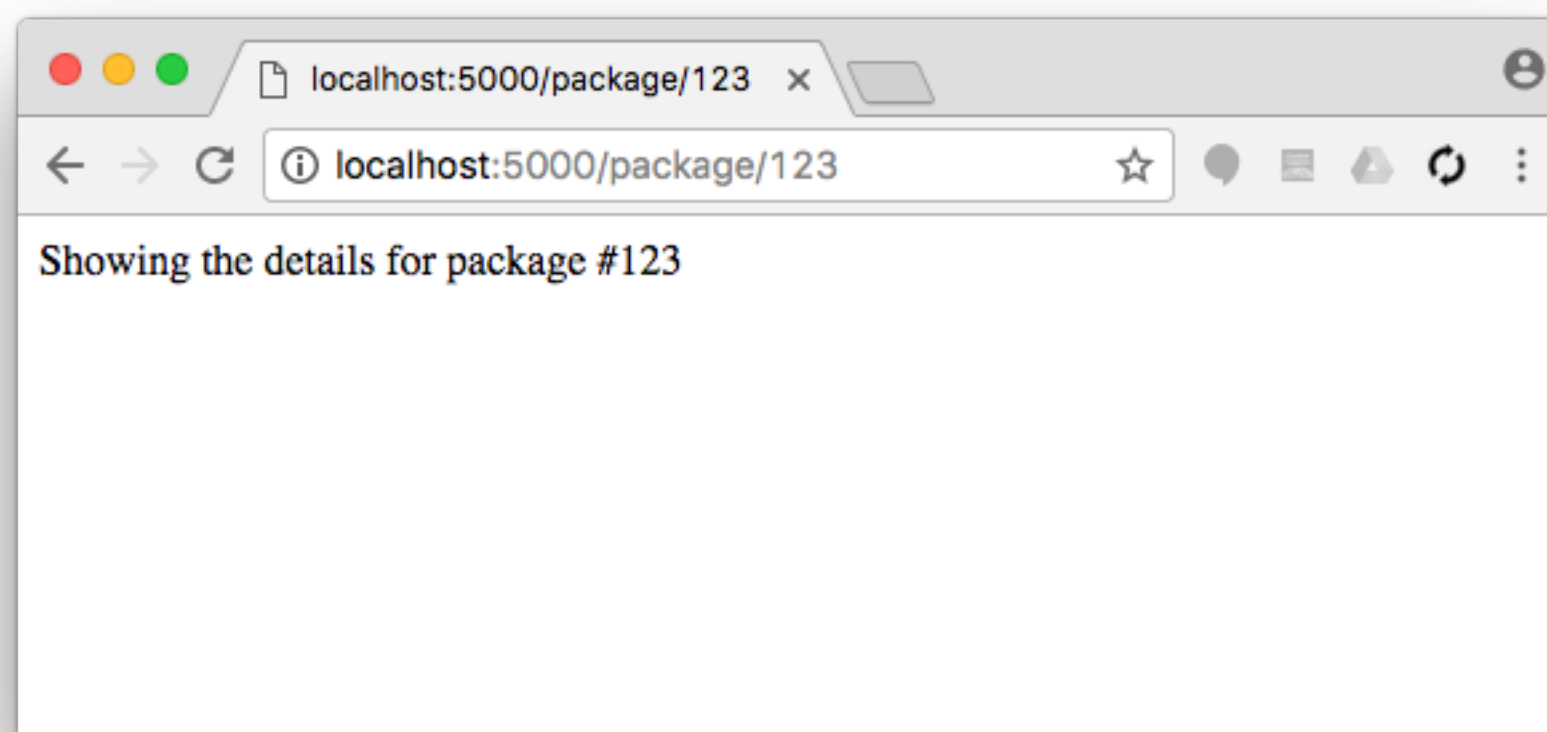
- **<converter:varname>** — optionally, a converter may be used to convert the input variable to the specified format
- Converters:
 - string (default)
 - int
 - float
 - path (same as the default, but also accepts slashes)
 - ...

Example

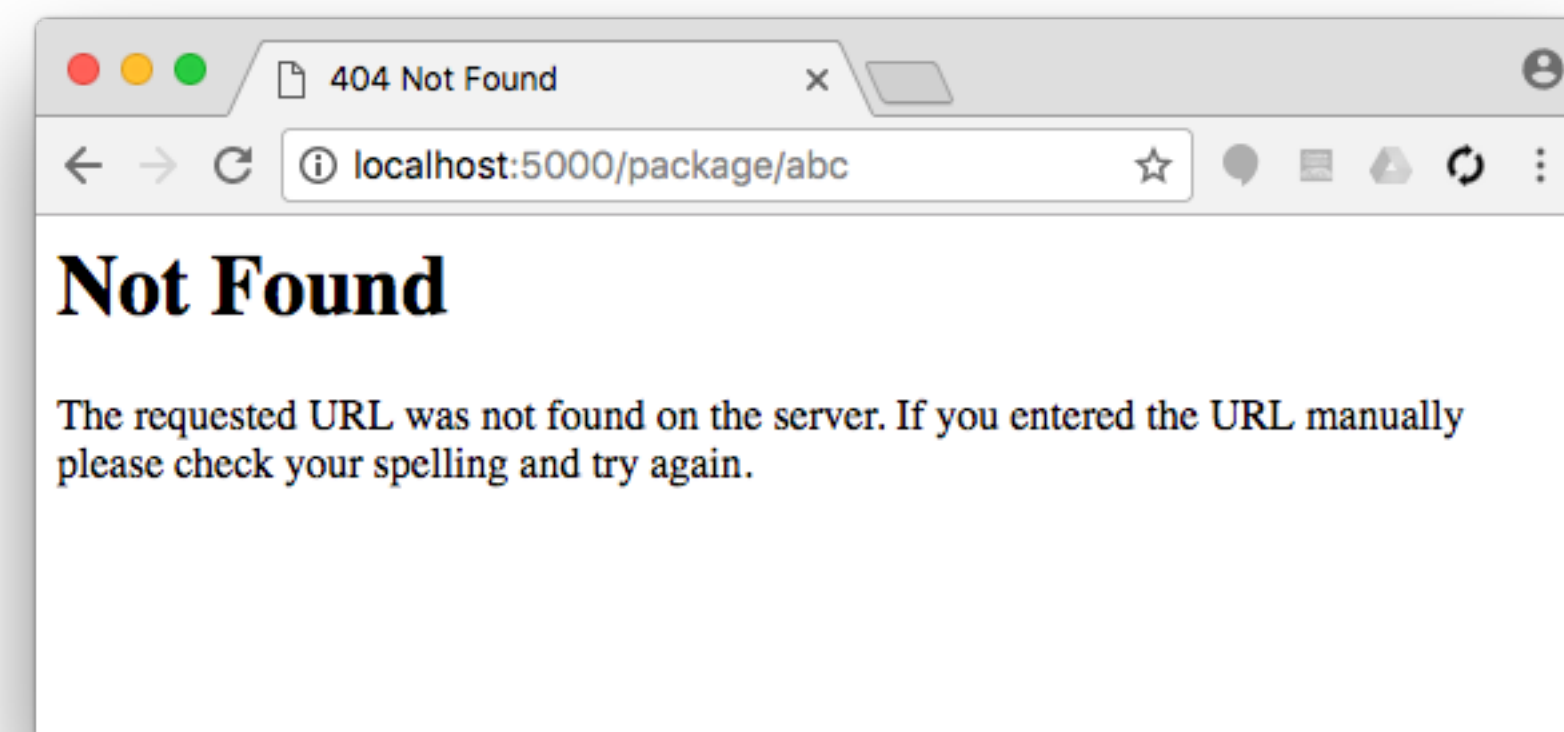
🔗 `examples/flask/1_routing/app.py`

```
@app.route("/package/<int:package_id>")
def package(package_id):
    return "Showing the details for package #{}".format(package_id)
```

<http://localhost:5000/package/123>



<http://localhost:5000/package/abc>



URL Building

- **url_for()** — generates a URL for a specific function
 - first argument is the function name, optionally a number of additional arguments corresponding to the variable part of the URL rule

```
@app.route("/")
def index():
    # ...

@app.route("/user/<username>")
def user(username):
    # ...
```

```
from flask import url_for

print(url_for("index")) # /
print(url_for("user", username="JohnDoe")) # /user/JohnDoe
```

Exercise #1



[https://github.com/dat310-2023/info/tree/master/](https://github.com/dat310-2023/info/tree/master/exercises/python/flask1)
exercises/python/flask1

Serving Static Files

- Dynamic web applications also need static files (css, javascript, images, etc.)
- Keep them under a folder called **static**
- To generate URLs for them use the special **static** endpoint name

```
url_for("static", filename="style.css")
```

Example

🔗 `examples/flask/2_static/app.py`

```
HTML_FRAME_TOP = "<!DOCTYPE HTML>\n<html>\n<head>\n<title>My site</title>\n" \
                  "<link rel=\"stylesheet\" href=\"{css}\"/>\n</head>\n<body>\n"
HTML_FRAME_BOTTOM = "</body>\n</html>\n"
```

```
@app.route("/")
def index():
    html = HTML_FRAME_TOP.replace("{css}", url_for("static", filename="style.css"))
    html += "<h1>Hello world</h1>"
    html += HTML_FRAME_BOTTOM
    return html
```

we replace the string {css} with the URL generated for the static "style.css" file

Accessing Request Data

- In Flask, this information is provided by the global **request** object

```
from flask import request
```

```
@app.route("/login", methods=["POST", "GET"])
def login():
    error = None
    if request.method == "POST":
        if valid_login(request.form["username"],
                        request.form["password"]):
            # do something
        else:
            error = "Invalid username/password"
```


Accessing Request Data

- Accessing parameters submitted in the URL (on GET)
- These are contained in **request.args** (dict)
- Checking if a param has been provided

```
if "name" in request.args:  
    print(request.args["name"])
```

- Getting param with default value

```
print(request.args.get("name", ""))
```

- Iterating all parameters

```
for k, v in request.args.items():  
    print("{:20} : {}".format(k, v))
```

Redirects and Errors

- Redirect the user to another endpoint

```
@app.route('/')  
def index():  
    return redirect(url_for('login'))
```

- Abort a request with an early error code

```
from flask import abort  
  
@app.route('/login')  
def login():  
    abort(401)  
    # this_is_never_executed()
```

Custom Error Pages

- Use the **errorhandler()** decorator

```
from flask import render_template

@app.errorhandler(404)
def page_not_found(error):
    return render_template('page_not_found.html'), 404
```

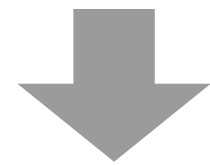
Example

📄 examples/flask/3_forms/app.py

```
from flask import Flask, url_for, redirect, request

@app.route("/")
def index():
    return redirect(url_for("static", filename="form.html"))
```

localhost:5000/



localhost:5000/static/form.html

redirect to static form page from web root

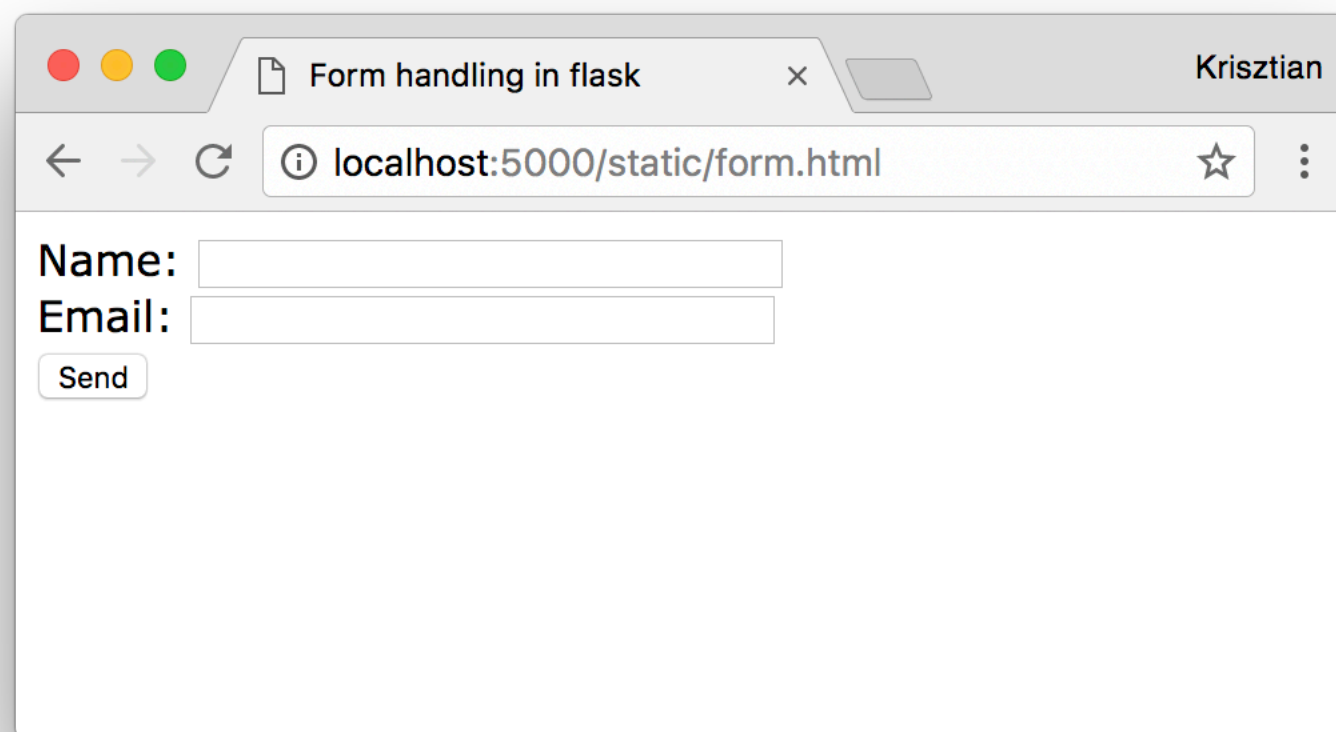
The screenshot shows a web browser window with the title 'Form handling in flask'. The address bar displays 'localhost:5000/static/form.html'. The form contains two input fields: 'Name:' and 'Email:', each followed by a text input box. Below these fields is a 'Send' button. The browser's tab bar shows the current tab and a user profile icon labeled 'Krisztian'.

Example

🔗 `examples/flask/3_forms/app.py`

```
@app.route("/sendform", methods=["POST"])
def sendform():
    html = HTML_FRAME_TOP.replace("{css}", url_for("static", filename="style.css"))
    html += "Name: " + request.form["name"] \
           + "<br />" \
           + "Email: " + request.form["email"]
    html += HTML_FRAME_BOTTOM
    return html
```

localhost:5000/static/form.html



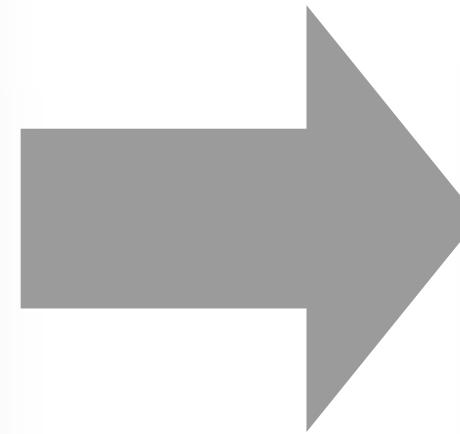
Form handling in flask

localhost:5000/static/form.html

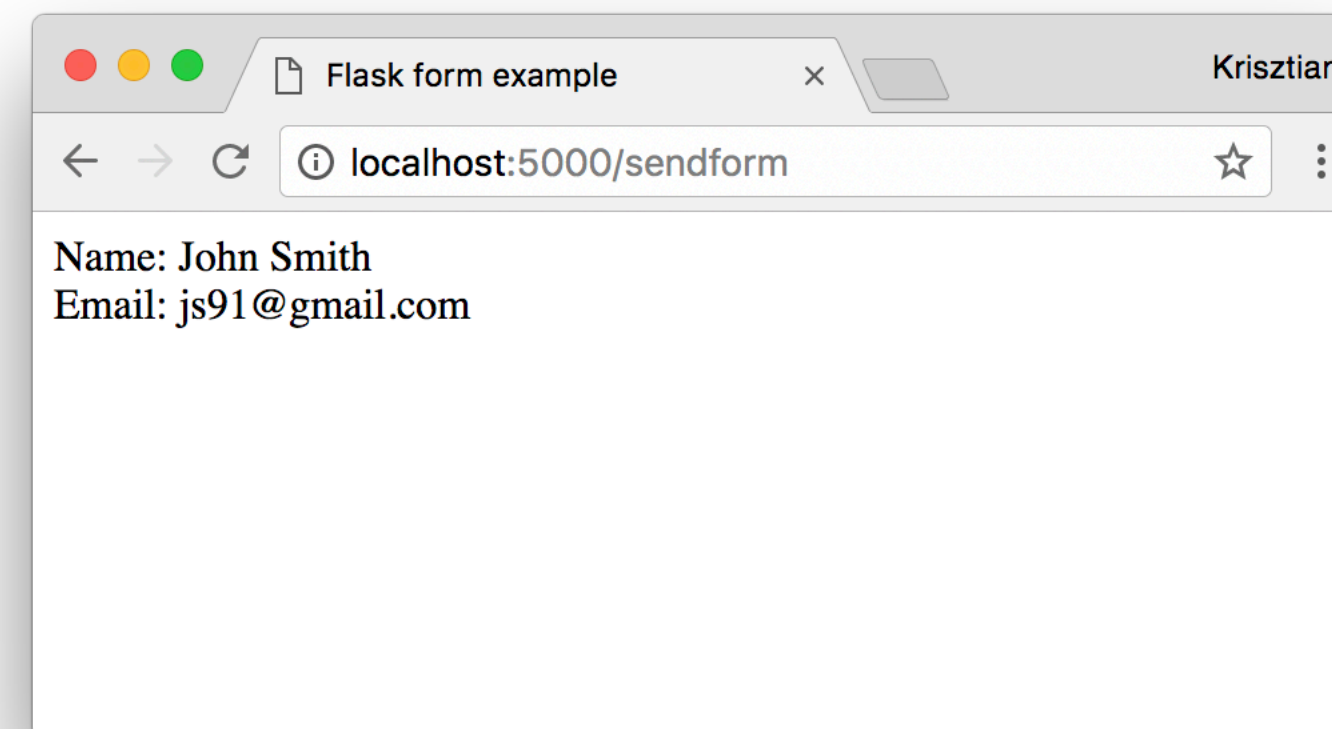
Name:

Email:

Send



<http://localhost:5000/sendform>



Flask form example

localhost:5000/sendform

Name: John Smith
Email: js91@gmail.com

Exercise #2



[https://github.com/dat310-2023/info/tree/master/](https://github.com/dat310-2023/info/tree/master/exercises/python/flask1)
exercises/python/flask1