CSE 10001
Server-Side Programming

Warmup Questions

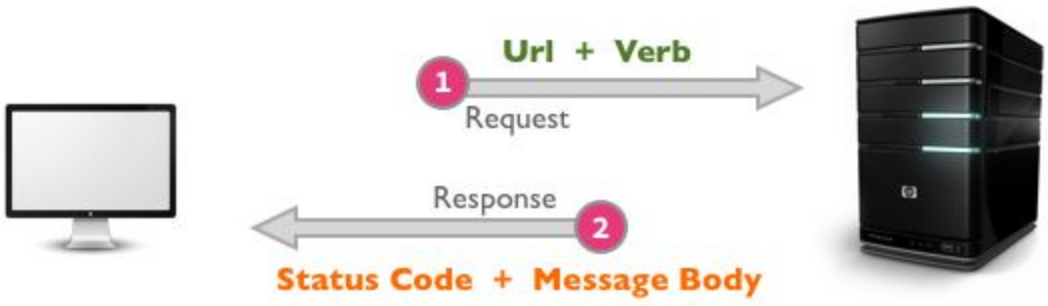
- What happens when you enter a **URL** in your web browser?
- What is a **client** and is a **server**? What is each responsible for in a **HTTP** transaction?
- What exactly is **Tornado** and what does it allow us to do?

HTTP

HTTP: Overview

- **Hypertext Transfer Protocol**
- Just as we have file formats, we must also organize our **communication** for efficient computing
- **Stateless** form of communication
- Language of the **WWW**!

HTTP: Client/Server



- **Clients** (web browser) makes a **request** for a document or resource
- **Server** (web application) receives **request**, processes it, and sends back **response**

HTTP: Request/Response

Request

```
GET / HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Cache-Control: max-age=0
Accept:
text/html,application/xhtml+xml,application/x
ml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/46.0.2490.80 Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
```

Response

```
HTTP/1.0 200 OK
Content-Type: text/html

<html>
...
</html>
```

Tornado

Tornado: Overview

Python **web framework** and asynchronous **networking** library

- Use it to build **web applications**
- Comes with lots of built-in functionality, but is still light-weight

hello

Tornado: Hello, World

This is bare minimum amount of code for a simple "Hello, World" program using Python and Tornado:

```
import tornado.ioloop
import tornado.options
import tornado.web

PORT = 9999

class HelloHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, World!")

Application = tornado.web.Application([
    (r"/", HelloHandler),
], debug=True)
Application.listen(PORT)

tornado.options.parse_command_line()
tornado.ioloop.IOLoop.current().start()
```

Tornado: Format HTML

Instead of responding with plain text, we can also write **HTML** by embedding the code in strings:

```
class HelloHandler(tornado.web.RequestHandler):
    def get(self):
        self.write('<h1>Hello, World!</h1>')
        self.write('<img src="http://yld.me/awX?raw=1">')
```

form

Tornado: Form Processing

To read input from the user, we must create a form where each input element has a name and then retrieve the value of each element by using the `get_argument` method.

```
class FormHandler(tornado.web.RequestHandler):
    def get(self):
        name = self.get_argument('name', '')
        if name:
            self.write('<b>Hello, {}</b>'.format(name))
        self.write('''
 <h1>Enter Your Name:</h1>
 <form>
     <input type="text" name="name" value="{}">
     <input type="submit" value="Echo!">
 </form>
 '''.format(name))
```

templates

Tornado: Templates

Rather than embedding **HTML** inside of strings, we can put our **HTML** code in a separate **template** file and then render it by passing arguments.

Python

```python
class TemplateHandler(tornado.web.RequestHandler):
    def get(self):
        name = self.get_argument('name', '')
        self.render('template.html', name=name)
```

HTML Template

```
<html>
    <body>
         {% if name %}
         <b>Hello, {{ name }}</b>
         {% end %}
         <h1>Enter Your Name</h1>
         <form>
             <input type="text" name="name" value="{{ name }}">
             <input type="submit" value="Echo!">
         </form>
     </body>
</html>
```

routing

Tornado: Routing

We can route different URLs to separate handlers by creating a series of **regular expressions** and associating them to different handler classes. Likewise, we can even specify URL arguments by performing grouping in the URL regular expressions.

```
class IndexHandler(tornado.web.RequestHandler):
    def get(self):
        self.write('Index!')

class PageHandler(tornado.web.RequestHandler):
    def get(self, page_id):
        self.write('Page {}'.format(page_id))

Application = tornado.web.Application([
    (r'/page/(.*)', PageHandler),
    (r'/', IndexHandler),
])
```

Tornado: Static Files

For **static content** such as images, stylesheets, or JavaScript, we can simply use the built-in **StaticFileHandler** to serve that content.

```
class IndexHandler(tornado.web.RequestHandler):
    def get(self):
        self.write('Hello, World')

Application = tornado.web.Application([
    (r'/files/(.*)', tornado.web.StaticFileHandler, {'path': os.curdir}),
    (r'/', IndexHandler),
])
```

Tornado: Examples

Here are some example web applications built using **Tornado**:

- yldme
- notifyd
- dredd

All applications in this presentation can be found here:

Tornado Examples