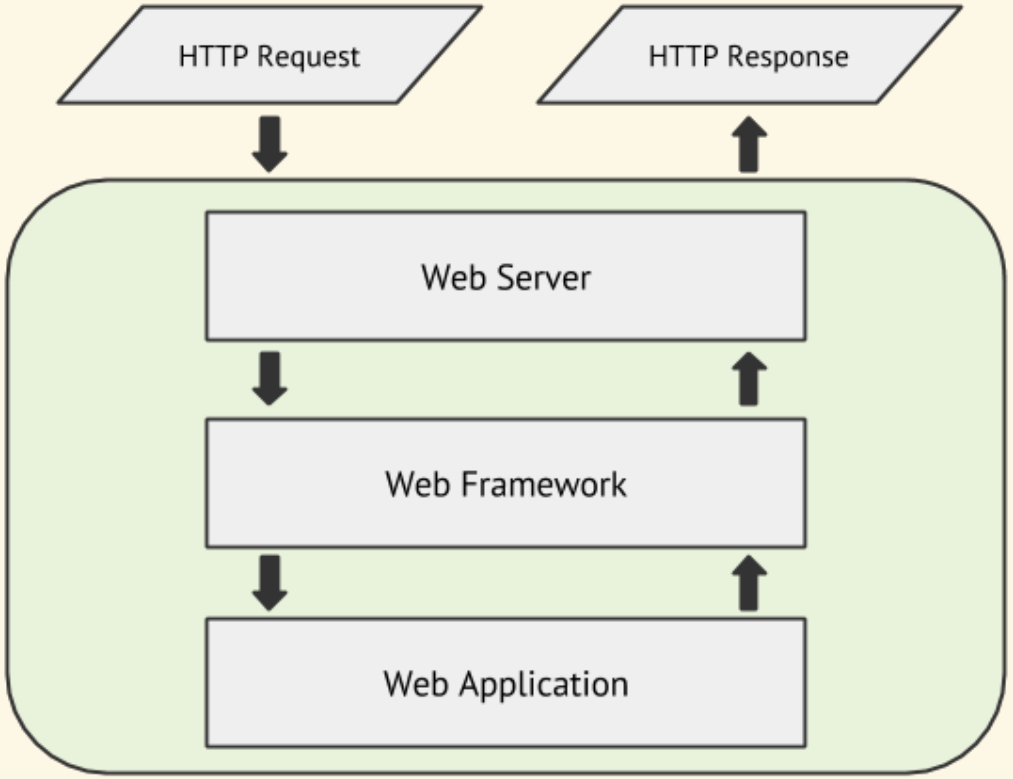
Python Web Frameworks

a collection of packages or modules which allow developers to write web applications or services.

What is a web app?

- receives an HTTP request
- does something
- returns an HTTP response


HTTP Request
HTTP Response
Web Server
Web Framework
Web Application

The Request

```
GET / HTTP/1.1
Host: example.org
User-Agent: Chrome
```

```
print("HTTP/1.1 200 OK"
      "\r\n\r\n"
      "<html><body>Hello</body></html>"
      "\r\n")
```

The Response

```
HTTP/1.1 200 OK

<html><body>Hello</body></html>
```

The Old Way

- common gateway interface (cgi)
- mod_python

WSGI

PEP 333

a standard interface between web servers and Python web applications or frameworks, to promote web application portability across a variety of web servers

WSGI Interface

```
environment = {
    'REQUEST_METHOD': 'GET',
    'HTTP_HOST': 'example.com:80',
    'wsgi.version': (1, 0),
    ...
}
def start_response(status_code, headers):
    ...
```

```
['<html><body>', 'Hello', '</body></html>']
```

WSGI application

```
def simple_app(environment, start_response):
    status = '200 OK'
    response_headers = [('Content-type', 'text/plain')]
    start_response(status, response_headers)
    return ['Hello world!\n']
```

Why Frameworks?

- request routing
- input validation
- data persistence
- response templating
- convention

Routing

/some/restful/url → webapp.views.get_url()

Decorators

frameworks: Flask, Bottle, Pyramid

```
@app.route("/hello/plain")
def hello_view():
    return "Hello World!"
```

Regex / Patterns

frameworks: Django, web.py, web2py

```
urlpatterns = patterns('',
    url(r'^articles/(\d{4})/$',
        'news.views.year_archive'),
    ...
)
```

Traversal / Object Dispatch

frameworks: Twisted, TurboGears, CherryPy

```
class RootController(BaseController):
    movie = MovieController()

class MovieController(BaseController):

    # handles url /movie/list
    @expose()
    def list(self):
        return 'hello'
```

Validation

```
POST /event/34/register HTTP/1.1

name=Daniel&guests=4&date=2013-05-01
```

```
event_id = 34
details = {
    'name': 'Daniel',
    'guests': 4,
    'date': datetime.date(2013, 5, 1)
}
```

Request Objects

libraries: WebOb, Werkzeug

```
request = Request(environ)
request.body
request.headers
request.GET
...
```

Forms

libraries: WTForms, django.forms, web.py

```
class MyForm(Form):
    first_name = TextField(
        u'First Name'
        validators=[validators.required()],
        widget=FirstNameWidget())

    last_name  = TextField(
        u'Last Name',
        validators=[validators.optional()])
```

Schemas

libraries: colander

```
class Person(colander.MappingSchema):

    name = colander.SchemaNode(colander.String())

    age  = colander.SchemaNode(colander.Integer(),
                validator=colander.Range(0, 200))
```

Persistence

- relational databases
- key-value stores
- document stores

ORM

libraries: Django, sqlalchemy, web2py

```
class User(db.Model):
    id       = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    email    = db.Column(db.String(120), unique=True)
```

Templating

```
context = {
    'name': 'Python Dev',
    'location': 'Edmonton'
}
```

```
<div>
  <h3>Python Dev</h3>
  <p><strong>Location:</strong> Edmonton</p>
</div>
```

Mako

```
<table>
    % for row in rows:
        ${makerow(row)}
    % endfor
</table>
<%def name="makerow(row)">
    <tr>
    % for name in row:
        <td>${name}</td>\
    % endfor
    </tr>
</%def>
```

Jinja / Django

```
{% for item in navigation %}
    <li>
      <a href="{{ item.href }}">
          {{ item.caption|lower }}</a>
    </li>
{% endfor %}
```

Mustache

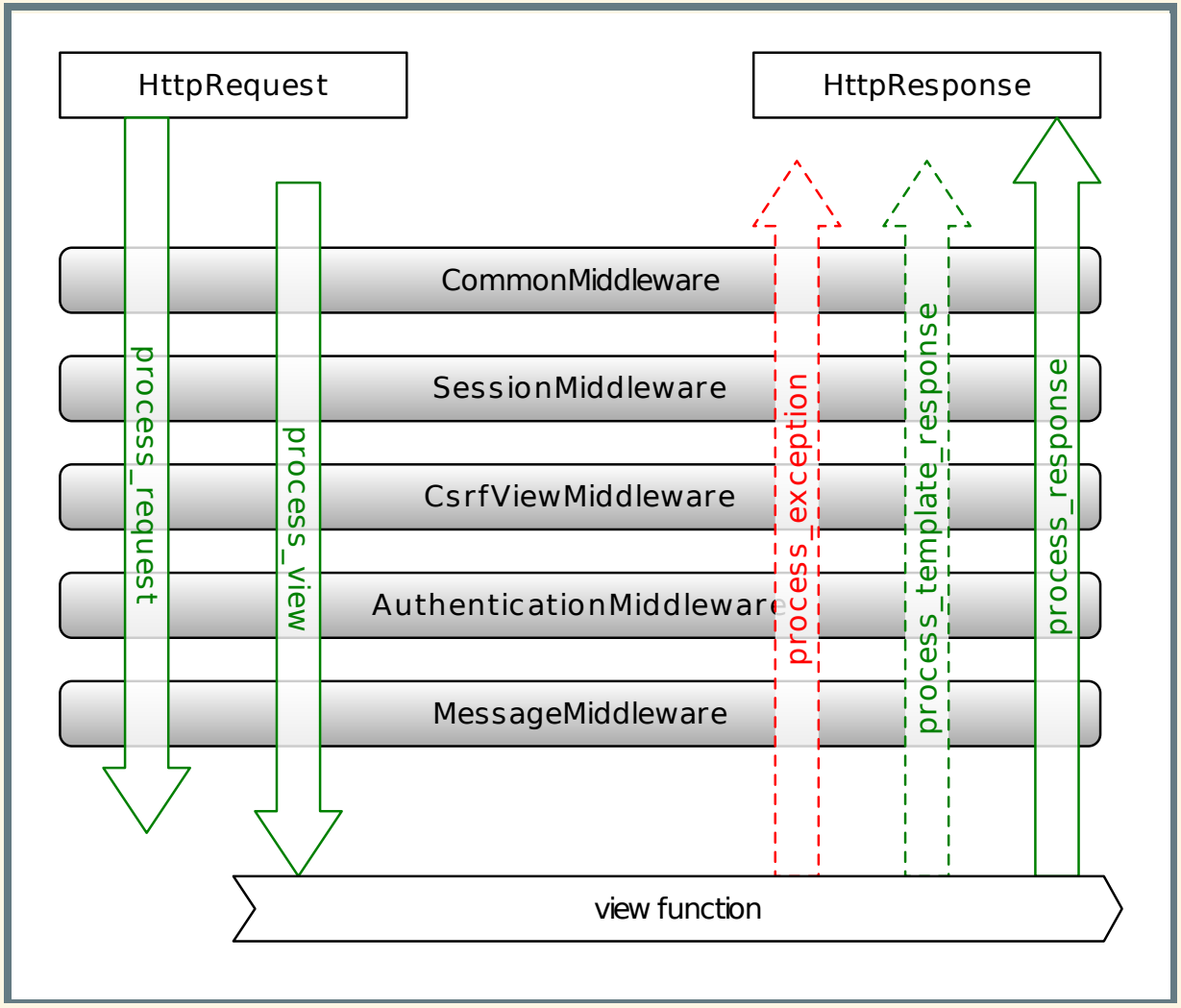
```
{{#items}}
  {{#first}}
    <li><strong>{{name}}</strong></li>
  {{/first}}
  {{#link}}
    <li><a href="{{url}}">{{name}}</a></li>
  {{/link}}
{{/items}}
```

Extensions

Django

Middleware

- Request
  - process_request()
  - process_view()
- Response
  - process_exception()
  - process_template_response()
  - process_response()


HttpRequest
HttpResponse
CommonMiddleware
SessionMiddleware
CsrfViewMiddleware
AuthenticationMiddleware
MessageMiddleware
process_request
process_view
process_exception
process_template_response
process_response
view function

Bottle

Hooks, and Plugins

```
@hook('before_request')
def check_permissions():
    ...

@hook('after_request')
def enable_cors():
    response.headers['X-Served'] = 1
```

Pyramid

- error views hook
- request factory hook
- before render event
- response callbacks
- finished callbacks
- traverser hook
- configuration decorators
- tweens

Why use a web framework?

- conventions
- documentation

The Frameworks

Microframeworks

Flask, Bottle, CherryPy

Full-Stack Frameworks

Django, TurboGears, web2py

Collection of Components

Pyramid

Read More

WSGI
Django
Pyramid
Flask
Bottle
SQLAlchemy
PyStache
Mako