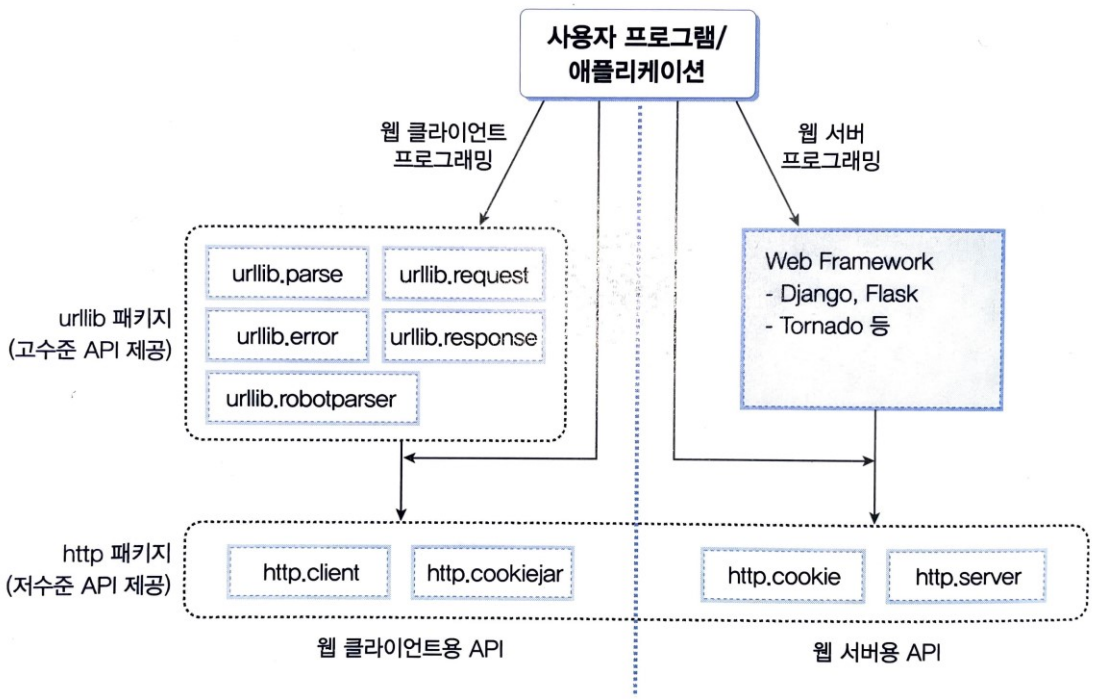
02장 파이썬 웹 표준 라이브러리

2.1 웹 라이브러리 구성

■ 파이썬의 웹 관련 라이브러리 3.x 버전에서는 관련된 모듈들을 모아서 패키지를 만들었고, 모듈명을 통해 서버쪽 라이브러리와 클라이언트쪽 라이브러리를 좀 더 확실히 구분하였다.
- urllib 패키지: 웹 클라이언트를 작성하는 데 사용되는 모듈들이 있으며, 가장 빈번하게 사용하는 모듈들이다.
- http 패키지: 크게 서버용과 클라이언트용 라이브러리로 나누어 모듈을 담고 있으며, 쿠키 관련 라이브러리도 http 패키지 내에서 서버용과 클라이언트용으로 모듈이 구분되어 있다.



2.2 웹 클라이언트 라이브러리

2.2.1 urlib.parse 모듈

■ 이 모듈은 URL의 분해, 조립, 변경 및 URL 문자 인코딩, 디코딩 등을 처리하는 함수를 제공한다.

■ urlparse() 함수는 URL을 파싱한 결과로 ParseResult 인스턴스를 반환한다. ParseResult 클래스 속성의 의미는 다음과 같다.
- scheme: URL에 사용된 프로토콜을 의미한다.
- netloc: 네트워크 위치. user:password@host:port 형식으로 표현되며, HTTP 프로토콜인 경우는 host:port 형식이다.
- path: 파일이나 애플리케이션 경로를 의미한다.
- params: 애플리케이션에 전달될 매개변수이다.
- query: 질의 문자열 또는 매개변수로 앰퍼샌드(&)로 구분된 '이름=값' 쌍 형식으로 표시

한다.

- fragment: 문서 내의 앵커 등 조각을 지정한다.

```
# 1. 가상 환경(v3PyBook)으로 진입한다.
D:\dev\workspace\django1>conda activate v3PyBook

# 2. bash 쉘을 실행하기전에 시스템 환경변수(path)에 bash.exe 파일이 있는
# "D:\Program Files\Git\bin" 경로를 지정한다.
(v3PyBook) D:\dev\workspace\django1>bash
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1:> vi ~/.bashrc
export PS1="\[\e[36;1m\]\u@\[\e[32;1m\]\h:\[\e[31;1m\]\w:> \[\e[0m\]"
export PYTHONHOME="/D/prod/Anaconda3/envs/v3PyBook"
~
~
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1:> source ~/.bashrc
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1:> echo $PYTHONHOME
/D/prod/Anaconda3/envs/v3PyBook

# 3. urllib.parse 모듈 사용
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1> python
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from urllib.parse import urlparse
>>> result = urlparse("http://www.python.org:80/guido/python.html;philosophy?overal=3#n10")
>>> result
ParseResult(scheme='http', netloc='www.python.org:80', path='/guido/python.html', params='philosophy',
query='overal=3', fragment='n10')
```

[꿀팁] 가상 환경 만들기

■ 가상 환경이 필요한 주된 이유는 여러 파이썬 프로그램이 버전만 다른 라이브러리를 사용할 경우 발생하는 충돌을 방지하기 위함이다.

```
D:\dev\workspace\django1>python -V
Python 3.8.5

D:\dev\workspace\django1>conda info --envs
# conda environments:
#
base                  *  D:\prod\Anaconda3

# Check if conda is installed in your path.
D:\dev\workspace\django1>conda -V
conda 4.10.3

# Update the conda environment
D:\dev\workspace\django1>conda update conda

# 파이썬 3.6을 사용하는 가상 환경 v3PyBook을 만든다.
D:\dev\workspace\django1>conda create -n v3PyBook python=3.6 anaconda

# v3PyBook 가상 환경으로 진입한다.
D:\dev\workspace\django1>conda activate v3PyBook
(v3PyBook) D:\dev\workspace\django1>

# 가상 환경에서 빠져나온다.
(v3PyBook) D:\dev\workspace\django1>conda deactivate
D:\dev\workspace\django1>
```

```
# Deletion of virtual environment
D:\dev\workspace\django1>conda remove -n v3PyBook --all
```

2.2.2 urlib.request 모듈

- 주어진 URL에서 데이터를 가져오는 기본 기능을 제공한다.
- urlopen() 함수 - GET 방식 요청

[ch2/2-1.py]

```
01  from urllib.request import urlopen
02
03  f = urlopen("http://www.example.com")
04
05  print(f.read(500).decode('utf-8'))
```

```
# 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python 2-1.py
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue",
Helvetica, Arial, sans-serif;
```

- urlopen() 함수 - POST 방식 요청
  - 04: urlopen() 함수를 호출 시 data 인자를 지정해주면, 함수는 자동으로 POST 방식으로 요청을 보낸다.

[ch2/2-2.py]

```
01  from urllib.request import urlopen
02
03  data = "language=python&framework=django"
04  f = urlopen("http://127.0.0.1:8000", bytes(data, encoding='utf-8'))
05
06  print(f.read(500).decode('utf-8'))
```

```
# 1. 실습 서버를 먼저 실행한다.
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2-test-server:> python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
```

```
December 03, 2021 - 22:56:36
Django version 2.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.


# 2. 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python 2-2.py
<!DOCTYPE html>
<html lang="ko">




<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>ch2-test-server</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css">

    <!-- my css -->
    <link rel="shortcut
```

■ urlopen() 함수 - Request 클래스로 요청 헤더 지정
- 18: url 인자에 문자열 대신에 Request 객체를 지정한다.

[ch2/2-3.py]

```
01  from urllib.request import urlopen, Request
02  from urllib.parse import urlencode
03
04
05  url = 'http://127.0.0.1:8000'
06
07  data = {
08      'name': '김석훈',
09      'email': 'shkim@naver.com',
10      'url': 'http://www.naver.com',
11  }
12  encData = urlencode(data)
13  postData = bytes(encData, encoding='utf-8')
14
15  req = Request(url, data=postData)
16  req.add_header('Content-Type', 'application/x-www-form-urlencoded')
17
18  f = urlopen(req)
19
20  print(f.info())
21  print(f.read(500).decode('utf-8'))
```

```
# 실행 결과
D:\dev\workspace\django>python d:/dev/workspace/django/beginner/ch2/2-3.py
Date: Wed, 10 Nov 2021 16:18:12 GMT
Server: WSGIServer/0.2 CPython/3.8.5
```

```
Content-Type: text/html; charset=utf-8
X-Frame-Options: DENY
Content-Length: 1758
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin


<!DOCTYPE html>
<html lang="ko">




<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>ch2-test-server</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css">

    <!-- my css -->
    <link rel="shortcut
```

2.2.3 urlib.request 모듈 예제

■ 다음 예제는 특정 웹 사이트에서 이미지만을 검색하여 그 리스트를 보여주는 코드이다.

- 16: HTML 문장이 주어지면 HTMLParser 클래스를 사용해서 이미지를 찾고, 그 리스트를 출력해주는 함수이다.

[ch2/parse_image.py]

```
01  from urllib.request import urlopen
02  from html.parser import HTMLParser
03
04
05  class ImageParser(HTMLParser):
06      def handle_starttag(self, tag, attrs):
07          if tag != 'img':
08              return
09          if not hasattr(self, 'result'):
10              self.result = []
11          for name, value in attrs:
12              if name == 'src':
13                  self.result.append(value)
14
15
16  def parse_image(data):
17      parser = ImageParser()
18      parser.feed(data)
19      dataSet = set(x for x in parser.result)
20      return dataSet
21
22
23  def main():
24      url = "http://www.google.co.kr"
```

```
25
26      with urlopen(url) as f:
27          charset = f.info().get_param('charset')
28          data = f.read().decode(charset)
29
30      dataSet = parse_image(data)
31      print("\n>>>>>>>>> Fetch Images from", url)
32      print('\n'.join(sorted(dataSet)))
33
34
35  if __name__ == '__main__':
36      main()
```

```
# 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python parse_image.py

>>>>>>>>> Fetch Images from http://www.google.co.kr
/logos/doodles/2021/seasonal-holidays-2021-6753651837109324-6752733080595605-cst.gif
/textinputassistant/tia.png
```

2.2.4 http.client 모듈

- 대부분의 웹 클라이언트 프로그램은 urllib.requet 모듈에 정의된 기능만으로도 작성이 가능하다. 그러나 GET, POST 이외의 방식으로 요청을 보내거나, 요청 헤더와 바디 사이에 타이머를 두어 시간을 지연시키는 등 urllib.request 모듈로는 쉽게 처리할 수 없는 경우, http.client 모듈을 사용한다.

- http.client 모듈 사용 - GET 방식 요청

[ch2/2-8.py]

```
01  from http.client import HTTPConnection
02
03  host = 'www.example.com'
04  conn = HTTPConnection(host)
05
06  conn.request('GET', '/')
07
08  r1 = conn.getresponse()
09  print(r1.status, r1.reason)
10
11  data1 = r1.read()
12  # 일부만 읽는 경우
13  # data1 = r1.read(100)
14
15  # 두번째 요청에 대한 테스트
16  conn.request('GET', '/')
17
18  r2 = conn.getresponse()
19  print(r2.status, r2.reason)
20
21  data2 = r2.read()
22  print(data2.decode())
23
24  conn.close()
```

```
# 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python 2-8.py
200 OK
200 OK
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue",
Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

■ http.client 모듈 사용 - HEAD 방식 요청

[ch2/2-9.py]

```
01  from http.client import HTTPConnection
02
03  conn = HTTPConnection('www.example.com')
04  conn.request('HEAD', '/')
05
06  resp = conn.getresponse()
```

```
07  print(resp.status, resp.reason)
08
09  data = resp.read()
10  print(len(data))
11  print(data == b'')
```

```
# 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python 2-9.py
200 OK
0
True
```

■ http.client 모듈 사용 - POST 방식 요청

[ch2/2-10.py]

```
01  from http.client import HTTPConnection
02  from urllib.parse import urlencode
03
04
05  host = '127.0.0.1:8000'
06  params = urlencode({
07      'language': 'python',
08      'name': '김석훈',
09      'email': 'shkim@naver.com',
10  })
11  headers = {
12      'Content-Type': 'application/x-www-form-urlencoded',
13      'Accept': 'text/plain',
14  }
15
16  conn = HTTPConnection(host)
17  conn.request('POST', '', params, headers)
18  resp = conn.getresponse()
19  print(resp.status, resp.reason)
20
21  data = resp.read()
22  print(data.decode('utf-8'))
23
24  conn.close()
```

```
# 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python 2-10.py
200 OK
<!DOCTYPE html>
<html lang="ko">



<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>ch2-test-server</title>
```

```
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap-
theme.min.css">

    <!-- my css -->
    <link rel="shortcut icon" href="/static/img/favicon.ico">

    <style type="text/css">
    .homewindow {
        background-size: 100%;
        height: 500px;
        border-top: 10px solid #ccc;
        border-bottom: 10px solid #ccc;
        padding: 20px 100px;
    }
    .title {
        color: #c80;
        font-weight: bold;
    }
    .homewindow ul li {
        font-weight: bold;
        line-height: 30px;
    }
    </style>

</head>

<body style="padding-top:90px;">

    <div class="homewindow">
        <h2 class="title">Django - Python Web Programming</h2>
        <br>
        <h3 style="margin-left: 25px;">ch2-test-server 를 사용
하는 예제 리스트</h3>
        <ul>
            <li>예제 2-2 urlopen() 함수 - POST 방식 요청</li>
            <li>예제 2-3 urlopen() 함수 - Request 클래스로 요청
 헤더 지정</li>
            <li>예제 2-4 urlopen() 함수 - HTTPBasicAuthHandler
클래스로 인증 요청</li>
            <li>예제 2-5 urlopen() 함수 - HTTPCookieProcessor
클래스로 쿠키 데이터를 포함하여 요청</li>
            <li>예제 2-10 http.client 모듈 사용 - POST 방식 요
청</li>
            <li>예제 2-11 http.client 모듈 사용 - PUT 방식 요청
</li>
        </ul>
    </div>

</body>
</html>
```

2.2.5 http.client 모듈 예제

■ 다음 예제는 특정 웹 사이트에서 이미지만을 검색하여 그 이미지를 다운로드하는 코드이다.

[ch2/download-image.py]

```
  01  import os
  02  from http.client import HTTPConnection
  03  from urllib.parse import urljoin, urlunparse
```

```
04  from urllib.request import urlretrieve
05  from html.parser import HTMLParser
06
07
08  class ImageParser(HTMLParser):
09      def handle_starttag(self, tag, attrs):
10          if tag != 'img':
11              return
12          if not hasattr(self, 'result'):
13              self.result = []
14          for name, value in attrs:
15              if name == 'src':
16                  self.result.append(value)
17
18
19  def download_image(url, data):
20
21      if not os.path.exists('DOWNLOAD'):
22          os.makedirs('DOWNLOAD')
23
24      parser = ImageParser()
25      parser.feed(data)
26      dataSet = set(x for x in parser.result)
27
28      for x in sorted(dataSet) :
29          imageUrl = urljoin(url, x)
30          basename = os.path.basename(imageUrl)
31          targetFile = os.path.join('DOWNLOAD', basename)
32
33          print("Downloading...", imageUrl)
34          urlretrieve(imageUrl, targetFile)
35
36
37  def main():
38      host = "www.google.co.kr"
39
40      conn = HTTPConnection(host)
41      conn.request("GET", '')
42      resp = conn.getresponse()
43
44      charset = resp.msg.get_param('charset')
45      data = resp.read().decode(charset)
46      conn.close()
47
48      print("\n>>>>>>>>> Download Images from", host)
49      url = urlunparse(('http', host, '', '', '', ''))
50      download_image(url, data)
51
52
53  if __name__ == '__main__':
54      main()
```

2.3 웹 서버 라이브러리

- 프레임워크는 개발자가 웹 서버 프로그램을 개발하기 쉽도록 저수준의 기능을 이미 만들어 놓은 기반 프로그램으로, 웹 서버 프로그램 개발자는 프레임워크를 활용하여 응용 로직만 개발하면 되기 때문에 훨씬 효율적이다.

- 좀더 경험이 쌓이고 중급, 고급 전문가가 되기 위해서는 웹 프레임워크가 어떻게 동작하는지, 파이썬의 웹 서버 라이브러리가 웹 프레임워크에 어떻게 사용되는지 등에 대한 기술을

파악할 필요가 있다.

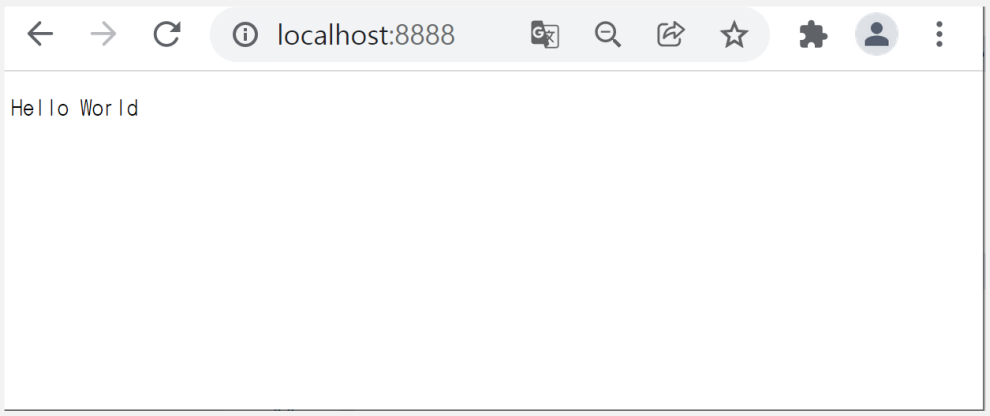
2.3.1 간단한 웹 서버

■ 아래 예제는 웹 클라이언트로부터 요청을 받고 “Hello World”라는 문장을 되돌려주는 아주 간단한 웹 서버이다.

[ch2/my_httpserver.py]

```
01  from http.server import HTTPServer, BaseHTTPRequestHandler
02
03
04  class MyHandler(BaseHTTPRequestHandler):
05      def do_GET(self):
06          self.send_response_only(200, 'OK')
07          self.send_header('Content-Type', 'text/plain')
08          self.end_headers()
09          self.wfile.write(b"Hello World")
10
11
12  if __name__ == '__main__':
13      server = HTTPServer(('', 8888), MyHandler)
14      print("Started WebServer on port 8888...")
15      print("Press ^C to quit WebServer.")
16      server.serve_forever()
```

```
# 실행 결과
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python my_httpserver.py
Started WebServer on port 8888...
Press ^C to quit WebServer.
```

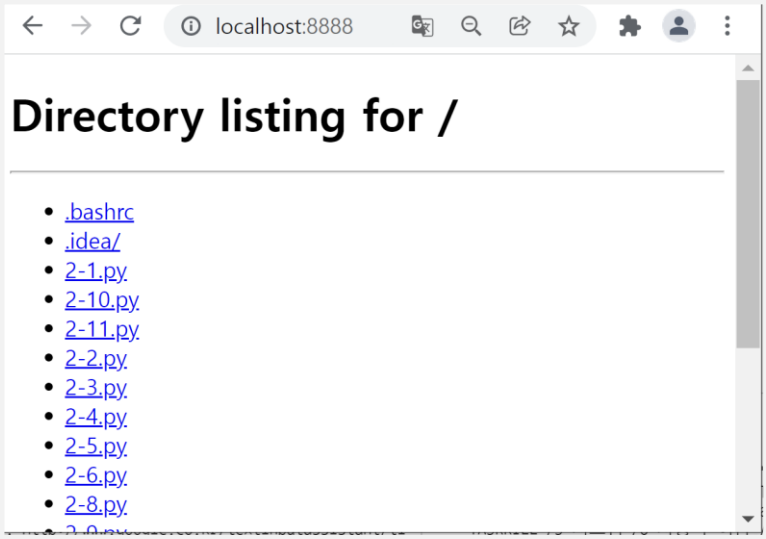


2.3.2 HTTPServer 및 BaseHTTPRequestHandler 클래스

■ 우리가 원하는 웹 서버를 만들기 위해서는 기반 클래스를 임포트하거나 상속받아야 한다. 이처럼 기반이 되는 클래스가 바로 HTTPServer 및 BaseHTTPRequestHandler 클래스이다.

```
# Simple 웹 서버 실행
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python http.server 8888
```

```
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```




```
# 2. 웹 서버 프로세스 종료
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2-test-server:> netstat -nao ¦ findstr 8888
  TCP    0.0.0.0:8888           0.0.0.0:0              LISTENING       2500
  TCP    127.0.0.1:8888         127.0.0.1:55416        ESTABLISHED     2500
  TCP    127.0.0.1:55416        127.0.0.1:8888         ESTABLISHED     9848
C:\Users\kshelp>taskkill /f /pid 2500
성공: 프로세스(PID 2500)가 종료되었습니다.
```

2.3.3 SimpleHTTPRequestHandler

- 별도의 코딩 없이도 필요할 때 즉시 웹 서버를 실행할 수 있도록 SimpleHTTPRequestHandler 클래스가 정의되어 있다.

2.3.4 CGIHTTPRequestHandler 클래스

- CGIHTTPRequestHandler 클래스에는 do_POST() 메소드가 정의되어 있어서 POST 방식을 처리할 수 있다. 물론 SimpleHTTPRequestHandler 클래스를 상속받고 있어서, GET 및 HEAD 방식도 처리한다. 다만 CGIHTTPServer 클래스의 메소드는 CGI 처리 기능만 구현되어 있어서 모든 POST 방식을 처리할 수 있는 것은 아니다.

```
# CGI 웹 서버 실행
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2/cgi-server:> python -m http.server 8888 --cgi
Serving HTTP on 0.0.0.0 port 8888 (http://0.0.0.0:8888/) ...
```

- CGI 웹 서버 시험용 CGI 스크립트

[ch2/cgi-server/cgi-bin/script.py]

```
01  import cgi
02
03
04  form = cgi.FieldStorage()
```

```
05  name = form.getvalue('name')
06  email = form.getvalue('email')
07  url = form.getvalue('url')
08
09  print("Content-Type: text/plain")
10  print()
11
12  print("Welcome... CGI Scripts")
13  print("name is", name)
14  print("email is", email)
15  print("url is", url)
```

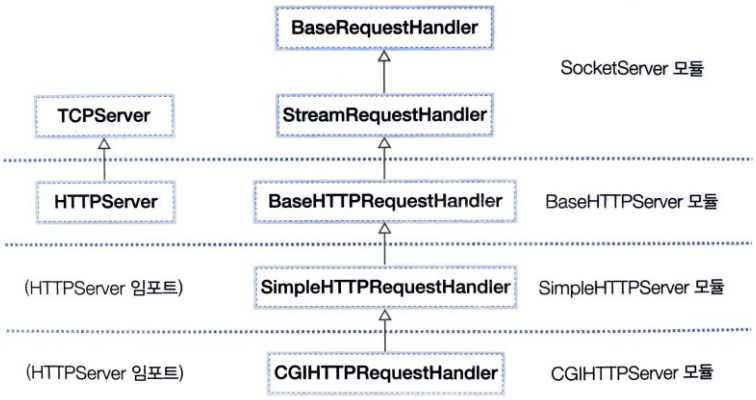
■ CGI 웹 서버 시험용 클라이언트

[ch2/cgi_client.py]

```
01  from urllib.request import urlopen
02  from urllib.parse import urlencode
03
04
05  url = "http://127.0.0.1:8888/cgi-bin/script.py"
06  data = {
07      'name': '김석훈',
08      'email': 'shkim@naver.com',
09      'url': 'http://www.naver.com',
10  }
11  encData = urlencode(data)
12  postData = encData.encode('ascii')
13
14  f = urlopen(url, postData)   # POST
15  print(f.read().decode('cp949'))
```

```
# CGI 웹 서버 실행
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2:> python cgi_client.py
Welcome... CGI Scripts
name is 김석훈
email is shkim@naver.com
url is http://www.naver.com
```

~~2.3.5 xxxHTTPServer 모듈 간의 관계(파이썬 2.x 버전만 해당)~~



2.4 CGI/WSGI 라이브러리

- 파이썬에서는 WSGI(Web Server Gateway Interface) 규격이 정의되어 있다. 파이썬 애플리케이션을 실행하고자 하는 웹 서버는 이 규격을 준수해야 한다. WSGI는 웹 서버와 웹 애플리케이션을 연결해주는 규격으로, 장고(Django)와 같은 파이썬 웹 프레임워크를 개발하거나, 이런 웹 프레임워크를 아파치(Apache Httpd)와 같은 웹 서버와 연동할 때 사용한다.

2.4.1 CGI 관련 모듈

- 웹 서버가 사용자의 요청을 애플리케이션에 전달하고 애플리케이션의 처리 결과를 애플리케이션으로부터 되돌려받기 위한, 즉 웹 서버와 애플리케이션 간에 데이터를 주고받기 위한 규격을 CGI(Common Gateway Interface)라고 한다.

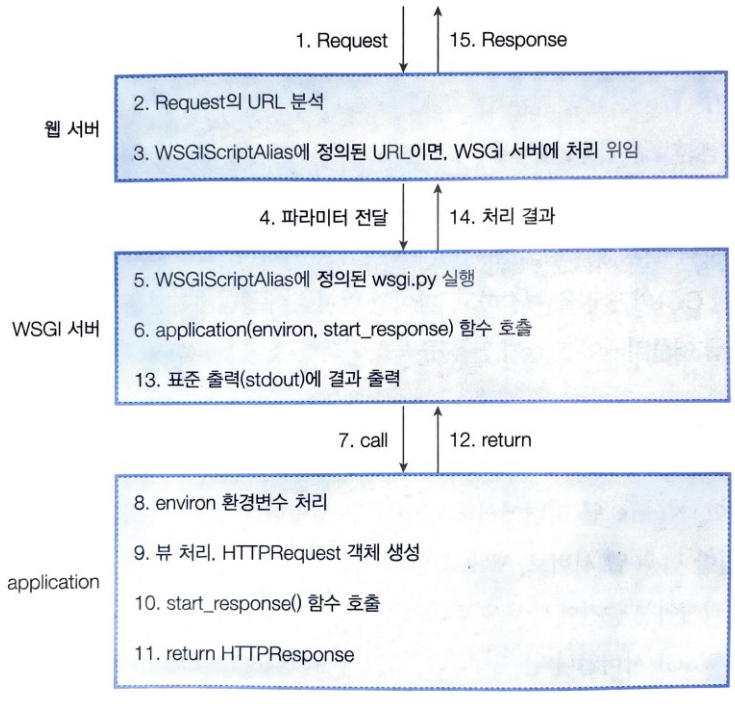
2.4.2 WSGI 개요

- CGI 방식은 요청이 들어올 때마다 처리를 위한 프로세스가 생성되는 방식이라서, 짧은 시간에 수천, 수만의 다량 요청을 받으면 서버의 부하가 높아져서 프로세스가 멈추거나 다운될 수도 있다. 이러한 CGI의 단점을 해결하고 파이썬 언어로 애플리케이션을 좀 더 쉽게 작성할 수 있도록 웹 서버와 웹 애플리케이션 간에 연동 규격을 정의한 것이 WSGI 규격이다.

- 그래서 파이썬에서는 WSGI 규격만 맞추면 어떤 웹 서버에서도 파이썬 애플리케이션을 실행할 수 있는 것이다.

2.4.3 WSGI 서버의 애플리케이션 처리 과정

- WSGI 애플리케이션의 처리 순서



■ WSGI 규격에 따라 애플리케이션을 개발할 때 중요한 사항은 세 가지이다.

```
# 1. 개발이 필요한 애플리케이션을 함수 또는 클래스의 메소드로 정의하고 애플리케이션 함수의 인자는 다음과 같이 정의한다.
def application_name(environ, start_response):

#2. start_response 함수의 인자 역시 다음과 같이 정해져 있으므로 그대로 따른다.
start_response(status, headers)

# 3. 애플리케이션 함수의 리턴값은 응답 바디에 해당하는 내용으로 리스트나 제네레이터와 같은 iterable 타입이어야 한다.
```

2.4.4 wsgiref.simple_server 모듈

■ 파이썬 표준 라이브러리에서는 웹 프레임워크 개발자가 웹 서버와의 연동 기능을 개발할 수 있도록 wsgiref 패키지의 하위 모듈로 wsgiref.simple_server 모듈을 제공하고 있다.

■ wgsiref.simple_server 모듈을 사용해서 WSGI 서버를 만든다.
- 01,17,18: WSGI 규격을 준수하여 WSGI 서버를 작성할 수 있도록 API를 제공하고 있으며 make_server() 및 server_forever() 메소드가 그런 API의 일부이다.
- 04: 애플리케이션 로직을 호출 가능한(callable) 함수나 메소드로 정의한다.

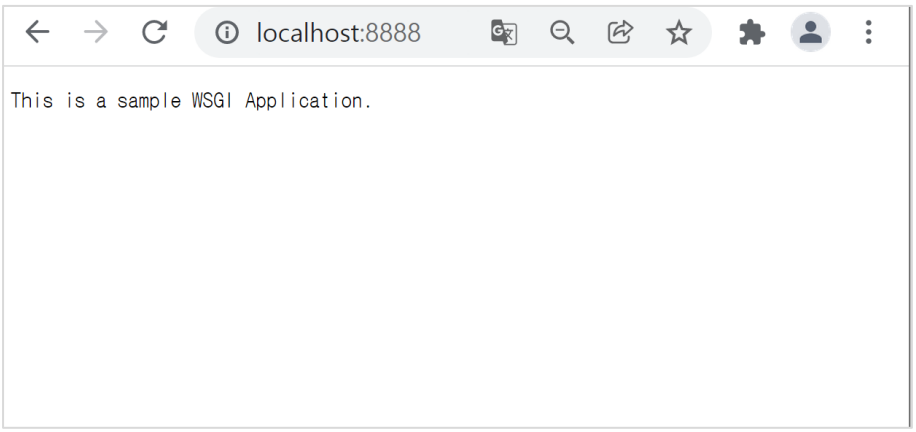
[ch2/wsgi-server/my_wsgiserver.py]

```
01  from wsgiref.simple_server import make_server
02
03
04  def my_app(environ, start_response):
05
06      status = '200 OK'
07      headers = [('Content-Type', 'text/plain')]
08      start_response(status, headers)
09
10      response = [b"This is a sample WSGI Application."]
11
12      return response
13
14
15  if __name__ == '__main__':
16      print("Started WSGI Server on port 8888...")
17      server = make_server('', 8888, my_app)
18      server.serve_forever()
```

2.4.5 WSGI 서버 동작 확인

■ WSGI 서버를 실행하기 위해서 다음 명령을 입력한다.

```
kshelp@DESKTOP-92Q7HNB:/d/dev/workspace/django1/ch2/wsgi-server:> python my_wsgiserver.py
Started WSGI Server on port 8888...
```


localhost:8888
This is a sample WSGI Application.