

EchoWsgiGet/runserver.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # runserver.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import wsgiref.simple_server
10: import echo
11:
12: def main():
13:
14:     if len(sys.argv) != 2:
15:         print('Usage: ' + sys.argv[0] + ' port', file=sys.stderr)
16:         sys.exit(1)
17:
18:     try:
19:         port = int(sys.argv[1])
20:     except Exception:
21:         print('Port must be an integer.', file=sys.stderr)
22:         sys.exit(1)
23:
24:     try:
25:         httpd = wsgiref.simple_server.make_server(
26:             '0.0.0.0', port, echo.app)
27:         print('Listening on port ' + str(port))
28:         httpd.serve_forever()
29:     except Exception as ex:
30:         print(ex, file=sys.stderr)
31:         sys.exit(1)
32:
33: if __name__ == '__main__':
34:     main()

```

blank (Page 1 of 1)

1: This page is intentionally blank.

EchoWsgiGet/echo.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # echo.py
5: # Author: Bob Dondero
6: #-----
7:
8: import html
9: import urllib.parse
10:
11: #-----
12:
13: def index(envIRON, start_response):
14:
15:     html_code = '''
16:         <!DOCTYPE html>
17:         <html>
18:             <body>
19:                 <form action="/echoresult" method="get">
20:                     Enter a line to be echoed:<br>
21:                     <input type="text" name="line"
22:                         value="Enter the line here">
23:                     <br>
24:                     <input type="submit">
25:                 </form>
26:             </body>
27:         </html>
28:     '''
29:
30:     content_header = ('content-type', 'text/html; charset=utf-8')
31:     headers = [content_header]
32:     start_response('200 OK', headers)
33:     return [html_code.encode('utf-8')]
34:
35: #-----
36:
37: def parse_args_get(envIRON, key):
38:
39:     args_str = envIRON.get('QUERY_STRING', None)
40:     if args_str is None:
41:         return None
42:     args = urllib.parse.parse_qs(args_str)
43:     values = args.get(key, None)
44:     if values is None or len(values) == 0:
45:         return None
46:     return values[0]
47:
48: #-----
49:
50: def echo_result(envIRON, start_response):
51:
52:     line_to_be_echoed = parse_args_get(envIRON, 'line')
53:     if line_to_be_echoed is None:
54:         line_to_be_echoed = '(You did not enter a line.)'
55:
56:     html_code = f'''
57:         <!DOCTYPE html>
58:         <html>
59:             <body>
60:                 <p>This is the line that you entered:</p>
61:                 <p>{html.escape(line_to_be_echoed)}</p>
62:             </body>
63:         </html>
64:     '''
65:

```

EchoWsgiGet/echo.py (Page 2 of 2)

```

66:     content_header = ('content-type', 'text/html; charset=utf-8')
67:     headers = [content_header]
68:     start_response('200 OK', headers)
69:     return [html_code.encode('utf-8')]
70:
71: #-----
72:
73: def not_found(envIRON, start_response):
74:
75:     html_code = '''
76:         <!DOCTYPE html>
77:         <html>
78:             <head>
79:                 <title>404 Not Found</title>
80:             </head>
81:             <body>
82:                 <h1>Not Found</h1>
83:                 <p>The requested URL was not found on the server.
84:                 If you entered the URL manually please check your
85:                 spelling and try again.</p>
86:             </body>
87:         </html>
88:     '''
89:
90:     content_header = ('content-type', 'text/html; charset=utf-8')
91:     headers = [content_header]
92:     start_response('404 Not Found', headers)
93:     return [html_code.encode('utf-8')]
94:
95: #-----
96:
97: def app(envIRON, start_response):
98:
99:     path = envIRON.get('PATH_INFO', '').strip('/')
100:
101:     if path in ('', 'index'):
102:         return index(envIRON, start_response)
103:     if path == 'echoresult':
104:         return echo_result(envIRON, start_response)
105:
106:     return not_found(envIRON, start_response)

```

EchoWsgiPost/echo.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # echo.py
5: # Author: Bob Dondero
6: #-----
7:
8: import html
9: import urllib.parse
10:
11: #-----
12:
13: def index(envIRON, start_response):
14:
15:     html_code = '''
16:         <!DOCTYPE html>
17:         <html>
18:             <body>
19:                 <form action="/echoresult" method="post">
20:                     Enter a line to be echoed:<br>
21:                     <input type="text" name="line"
22:                         value="Enter the line here">
23:                     <br>
24:                     <input type="submit">
25:                 </form>
26:             </body>
27:         </html>
28:     '''
29:
30:     content_header = ('content-type', 'text/html; charset=utf-8')
31:     headers = [content_header]
32:     start_response('200 OK', headers)
33:     return [html_code.encode('utf-8')]
34:
35: #-----
36:
37: def parse_args_post(envIRON, key):
38:
39:     content_length_str = envIRON.get('CONTENT_LENGTH', '')
40:     if content_length_str == '':
41:         return None
42:     content_length = int(content_length_str)
43:     request_body_bytes = envIRON['wsgi.input'].read(content_length)
44:     request_body_str = request_body_bytes.decode("utf-8")
45:     args = urllib.parse.parse_qs(request_body_str)
46:     values = args.get(key, None)
47:     if values is None or len(values) == 0:
48:         return None
49:     return values[0]
50:
51: #-----
52:
53: def echo_result(envIRON, start_response):
54:
55:     line_to_be_echoed = parse_args_post(envIRON, 'line')
56:     if line_to_be_echoed is None:
57:         line_to_be_echoed = '(You did not enter a line.)'
58:
59:     html_code = f'''
60:         <!DOCTYPE html>
61:         <html>
62:             <body>
63:                 <p>This is the line that you entered:</p>
64:                 <p>{html.escape(line_to_be_echoed)}</p>
65:             </body>

```

EchoWsgiPost/echo.py (Page 2 of 2)

```

66:         </html>
67:     '''
68:
69:     content_header = ('content-type', 'text/html; charset=utf-8')
70:     headers = [content_header]
71:     start_response('200 OK', headers)
72:     return [html_code.encode('utf-8')]
73:
74: #-----
75:
76: def not_found(envIRON, start_response):
77:
78:     html_code = '''
79:         <!DOCTYPE html>
80:         <html>
81:             <head>
82:                 <title>404 Not Found</title>
83:             </head>
84:             <body>
85:                 <h1>Not Found</h1>
86:                 <p>The requested URL was not found on the server.
87:                 If you entered the URL manually please check your
88:                 spelling and try again.</p>
89:             </body>
90:         </html>
91:     '''
92:
93:     content_header = ('content-type', 'text/html; charset=utf-8')
94:     headers = [content_header]
95:     start_response('404 Not Found', headers)
96:     return [html_code.encode('utf-8')]
97:
98: #-----
99:
100: def app(envIRON, start_response):
101:
102:     path = envIRON.get('PATH_INFO', '').strip('/')
103:
104:     if path in ('', 'index'):
105:         return index(envIRON, start_response)
106:     if path == 'echoresult':
107:         return echo_result(envIRON, start_response)
108:
109:     return not_found(envIRON, start_response)

```

PennyWsgi/runserver.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # runserver.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import wsgiref.simple_server
10: import penny
11:
12: def main():
13:
14:     if len(sys.argv) != 2:
15:         print(f'usage: {sys.argv[0]} port', file=sys.stderr)
16:         sys.exit(1)
17:
18:     try:
19:         port = int(sys.argv[1])
20:     except Exception:
21:         print(f'{sys.argv[0]}: Port must be an integer.',
22:               file=sys.stderr)
23:         sys.exit(1)
24:
25:     try:
26:         httpd = wsgiref.simple_server.make_server(
27:             '0.0.0.0', port, penny.app)
28:         print('Listening on port ' + str(port))
29:         httpd.serve_forever()
30:     except Exception as ex:
31:         print(f'{sys.argv[0]}: {ex}', file=sys.stderr)
32:         sys.exit(1)
33:
34: if __name__ == '__main__':
35:     main()

```

blank (Page 1 of 1)

```

1: This page is intentionally blank.

```

PennyWsgi/penny.sql (Page 1 of 1)

```

1: DROP TABLE IF EXISTS books;
2:
3: CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
4:
5: INSERT INTO books (isbn, author, title)
6:   VALUES ('123', 'Kernighan', 'The Practice of Programming');
7: INSERT INTO books (isbn, author, title)
8:   VALUES ('234', 'Kernighan', 'The C Programming Language');
9: INSERT INTO books (isbn, author, title)
10:  VALUES ('345', 'Sedgewick', 'Algorithms in C');

```

PennyWsgi/database.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # database.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sqlite3
9: import contextlib
10:
11: #-----
12:
13: _DATABASE_URL = 'file:penny.sqlite'
14:
15: #-----
16:
17: def get_books(author):
18:
19:     books = []
20:
21:     with contextlib.closing(
22:         sqlite3.connect(_DATABASE_URL + '?mode=ro',
23:             isolation_level=None, uri=True)) as connection:
24:
25:         with contextlib.closing(connection.cursor()) as cursor:
26:
27:             query_str = '''
28:                 SELECT isbn, author, title FROM books
29:                 WHERE author LIKE ?
30:             '''
31:             cursor.execute(query_str, [author+'%'])
32:
33:             table = cursor.fetchall()
34:             for row in table:
35:                 book = {'isbn': row[0], 'author': row[1],
36:                     'title': row[2]}
37:                 books.append(book)
38:
39:     return books
40:
41: #-----
42:
43: def _test():
44:
45:     books = get_books('ker')
46:     for book in books:
47:         print(book['isbn'])
48:         print(book['author'])
49:         print(book['title'])
50:         print()
51:
52: if __name__ == '__main__':
53:     _test()

```

PennyWsgi/penny.py (Page 1 of 4)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import html
9: import urllib.parse
10: import time
11: import database
12:
13: #-----
14:
15: def get_header():
16:
17:     if time.strftime('%p') == 'AM':
18:         ampm = 'morning'
19:     else:
20:         ampm = 'afternoon'
21:
22:     html_str = f'''
23:     <hr>
24:     Good {ampm} and welcome to <strong>Penny.com</strong>
25:     <hr>
26:     '''
27:
28:     return html_str
29:
30: #-----
31:
32: def get_footer():
33:
34:     current_time = time.asctime(time.localtime())
35:
36:     html_str = f'''
37:     <hr>
38:     Today is {current_time}<br>
39:     Created by <a href="https://www.cs.princeton.edu/~rdondero">
40:     Bob Dondero</a>
41:     <hr>
42:     '''
43:
44:     return html_str
45:
46: #-----
47:
48: def index(environ, start_response):
49:
50:     html_code = f'''
51:     <!DOCTYPE html>
52:     <html>
53:         <head>
54:             <title>Penny.com</title>
55:         </head>
56:         <body>
57:             {get_header()}
58:             <br>
59:             Click to <a href="/searchform">begin</a>.<br>
60:             <br>
61:             {get_footer()}
62:         </body>
63:     </html>
64:     '''
65:

```

PennyWsgi/penny.py (Page 2 of 4)

```

66:     content_header = ('content-type', 'text/html; charset=utf-8')
67:     headers = [content_header]
68:     start_response('200 OK', headers)
69:     return [html_code.encode('utf-8')]
70:
71: #-----
72:
73: def search_form(environ, start_response):
74:
75:     html_code = f'''
76:     <!DOCTYPE html>
77:     <html>
78:         <head>
79:             <title>Penny.com</title>
80:         </head>
81:         <body>
82:             {get_header()}
83:             <h1>Author Search</h1>
84:             <form action="/searchresults" method="get">
85:                 Please enter an author name:
86:                 <input type="text" name="author" autofocus>
87:                 <input type="submit" value="Go">
88:             </form>
89:             <br>
90:             <br>
91:             {get_footer()}
92:         </body>
93:     </html>
94:     '''
95:
96:     content_header = ('content-type', 'text/html; charset=utf-8')
97:     headers = [content_header]
98:     start_response('200 OK', headers)
99:     return [html_code.encode('utf-8')]
100:
101: #-----
102:
103: def parse_args_get(environ, key):
104:
105:     args_str = environ.get('QUERY_STRING', None)
106:     if args_str is None:
107:         return None
108:     args = urllib.parse.parse_qs(args_str)
109:     values = args.get(key, None)
110:     if values is None or len(values) == 0:
111:         return None
112:     return values[0]
113:
114: #-----
115:
116: def convert_to_html(books):
117:
118:     if len(books) == 0:
119:         return '(None)'
120:     html_code = ''
121:     for book in books:
122:         html_code += f'''
123:         <strong>{html.escape(book['isbn'])}</strong>:
124:         {html.escape(book['author'])}:
125:         {html.escape(book['title'])}<br>
126:         '''
127:     return html_code
128:
129: #-----
130:

```

PennyWsgi/penny.py (Page 3 of 4)

```

131: def search_results(envIRON, start_response):
132:
133:     author = parse_args_get(envIRON, 'author')
134:     if author is None:
135:         author = ''
136:     author = author.strip()
137:
138:     if author == '':
139:         author = '(None)'
140:         books = []
141:     else:
142:         books = database.get_books(author) # Exception handling omitted
143:
144:     html_code = f'''
145:         <!DOCTYPE html>
146:         <html>
147:             <head>
148:                 <title>Penny.com</title>
149:             </head>
150:             <body>
151:                 {get_header()}
152:                 <h1>Author Search Results</h1>
153:                 <h2>Books by {html.escape(author)}:</h2>
154:                 {convert_to_html(books)}
155:                 <br>
156:                 <br>
157:                 Click here to do another
158:                 <a href="/searchform">author search</a>.
159:                 <br>
160:                 <br>
161:                 {get_footer()}
162:             </body>
163:         </html>
164:     '''
165:
166:     content_header = ('content-type', 'text/html; charset=utf-8')
167:     headers = [content_header]
168:     start_response('200 OK', headers)
169:     return [html_code.encode('utf-8')]
170:
171: #-----
172:
173: def not_found(envIRON, start_response):
174:
175:     html_code = f'''
176:         <!DOCTYPE html>
177:         <html>
178:             <head>
179:                 <title>404 Not Found</title>
180:             </head>
181:             <body>
182:                 <h1>Not Found</h1>
183:                 <p>The requested URL was not found on the server.
184:                 If you entered the URL manually please check your
185:                 spelling and try again.</p>
186:             </body>
187:         </html>
188:     '''
189:
190:     content_header = ('content-type', 'text/html; charset=utf-8')
191:     headers = [content_header]
192:     start_response('404 Not Found', headers)
193:     return [html_code.encode('utf-8')]
194:
195: #-----

```

PennyWsgi/penny.py (Page 4 of 4)

```

196:
197: def app(envIRON, start_response):
198:
199:     path = envIRON.get('PATH_INFO', '').strip('/')
200:
201:     if path in ('', 'index'):
202:         return index(envIRON, start_response)
203:     if path == 'searchform':
204:         return search_form(envIRON, start_response)
205:     if path == 'searchresults':
206:         return search_results(envIRON, start_response)
207:
208:     return not_found(envIRON, start_response)

```