

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()

```

PennyWsgi/common.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # common.py
5: # Author: Bob Dondero
6: #-----
7:
8: import time
9:
10: #-----
11:
12: def get_header():
13:
14:     if time.strftime('%p') == 'AM':
15:         ampm = 'morning'
16:     else:
17:         ampm = 'afternoon'
18:
19:     html_str = f'''
20:     <hr>
21:     Good {ampm} and welcome to <strong>Penny.com</strong>
22:     <hr>
23:     '''
24:
25:     return html_str
26:
27: #-----
28:
29: def get_footer():
30:
31:     current_time = time.asctime(time.localtime())
32:
33:     html_str = f'''
34:     <hr>
35:     Today is {current_time}<br>
36:     Created by <a href="https://www.cs.princeton.edu/~rdondero">
37:     Bob Dondero</a>
38:     <hr>
39:     '''
40:
41:     return html_str
42:
43: #-----
44:
45: # For testing:
46:
47: def _test():
48:
49:     print(get_header())
50:     print()
51:     print()
52:     print(get_footer())
53:
54: if __name__ == '__main__':
55:     _test()

```

PennyWsgi/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import http.cookies
9: import html
10: import common
11: import parseargs
12: import database
13:
14: #-----
15:
16: def index(environ, start_response):
17:
18:     html_code = f'''
19:     <!DOCTYPE html>
20:     <html>
21:         <head>
22:             <title>Penny.com</title>
23:         </head>
24:         <body>
25:             {common.get_header()}
26:             <br>
27:             Click to <a href="/searchform">begin</a>. <br>
28:             <br>
29:             {common.get_footer()}
30:         </body>
31:     </html>
32:     '''
33:
34:     content_header = ('content-type', 'text/html; charset=utf-8')
35:     headers = [content_header]
36:     start_response('200 OK', headers)
37:     return [html_code.encode('utf-8')]
38:
39: #-----
40:
41: def search_form(environ, start_response):
42:
43:     prev_author = '(None)'
44:     if 'HTTP_COOKIE' in environ:
45:         cookie = http.cookies.SimpleCookie(environ['HTTP_COOKIE'])
46:         if 'prev_author' in cookie:
47:             prev_author_morsel = cookie['prev_author']
48:             if prev_author_morsel:
49:                 prev_author = prev_author_morsel.value
50:
51:     html_code = f'''
52:     <!DOCTYPE html>
53:     <html>
54:         <head>
55:             <title>Penny.com</title>
56:         </head>
57:         <body>
58:             {common.get_header()}
59:             <h1>Author Search</h1>
60:             <form action="/searchresults" method="get">
61:                 Please enter an author name:
62:                 <input type="text" name="author" autofocus>
63:                 <input type="submit" value="Go">
64:             </form>
65:             <br>

```

PennyWsgi/penny.py (Page 2 of 3)

```

66:                 <br>
67:                 <strong>Previous author search:</strong>
68:                 {html.escape(prev_author)}
69:             <br>
70:             <br>
71:             {common.get_footer()}
72:         </body>
73:     </html>
74:     '''
75:
76:     content_header = ('content-type', 'text/html; charset=utf-8')
77:     headers = [content_header]
78:     start_response('200 OK', headers)
79:     return [html_code.encode('utf-8')]
80:
81: #-----
82:
83: def convert_to_html(books):
84:
85:     if len(books) == 0:
86:         return '(None)'
87:     html_code = ''
88:     for book in books:
89:         html_code += f'''
90:         <strong>{html.escape(book['isbn'])}</strong>:
91:         {html.escape(book['author'])}:
92:         {html.escape(book['title'])}<br>
93:         '''
94:     return html_code
95:
96: #-----
97:
98: def search_results(environ, start_response):
99:
100:    args_str = environ.get('QUERY_STRING', '')
101:    args = parseargs.parse(args_str)
102:    author = args.get('author', '')
103:
104:    author = author.strip()
105:
106:    if author == '':
107:        prev_author = '(None)'
108:        books = []
109:    else:
110:        prev_author = author
111:        books = database.get_books(author) # Exception handling omitted
112:
113:    html_code = f'''
114:    <!DOCTYPE html>
115:    <html>
116:        <head>
117:            <title>Penny.com</title>
118:        </head>
119:        <body>
120:            {common.get_header()}
121:            <h1>Author Search Results</h1>
122:            <h2>Books by {html.escape(prev_author)}:</h2>
123:            {convert_to_html(books)}
124:            <br>
125:            <br>
126:            Click here to do another
127:            <a href="/searchform">author search</a>.
128:            <br>
129:            <br>
130:            {common.get_footer()}

```

PennyWsgi/penny.py (Page 3 of 3)

```

131:         </body>
132:     </html>
133:     '''
134:
135:     content_header = ('content-type', 'text/html; charset=utf-8')
136:     cookie = http.cookies.SimpleCookie()
137:     cookie['prev_author'] = prev_author
138:     cookie_header = ('Set-Cookie', cookie['prev_author'].OutputString())
139:     headers = [content_header, cookie_header]
140:     start_response('200 OK', headers)
141:     return [html_code.encode('utf-8')]
142:
143: #-----
144:
145: def not_found(environ, start_response):
146:
147:     html_code = '''
148:     <!DOCTYPE html>
149:     <html>
150:         <head>
151:             <title>404 Not Found</title>
152:         </head>
153:         <body>
154:             <h1>Not Found</h1>
155:             <p>The requested URL was not found on the server.
156:             If you entered the URL manually please check your
157:             spelling and try again.</p>
158:         </body>
159:     </html>
160:     '''
161:
162:     content_header = ('content-type', 'text/html; charset=utf-8')
163:     headers = [content_header]
164:     start_response('404 Not Found', headers)
165:     return [html_code.encode('utf-8')]
166:
167: #-----
168:
169: def app(environ, start_response):
170:
171:     path = environ.get('PATH_INFO', '').strip('/')
172:
173:     if path in ('', 'index'):
174:         return index(environ, start_response)
175:     if path == 'searchform':
176:         return search_form(environ, start_response)
177:     if path == 'searchresults':
178:         return search_results(environ, start_response)
179:
180:     return not_found(environ, start_response)

```