

Server-Side Web Programming: CGI (Part 3)

Copyright © 2025 by
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
 - A fundamental example
 - Stateful web programming

Agenda

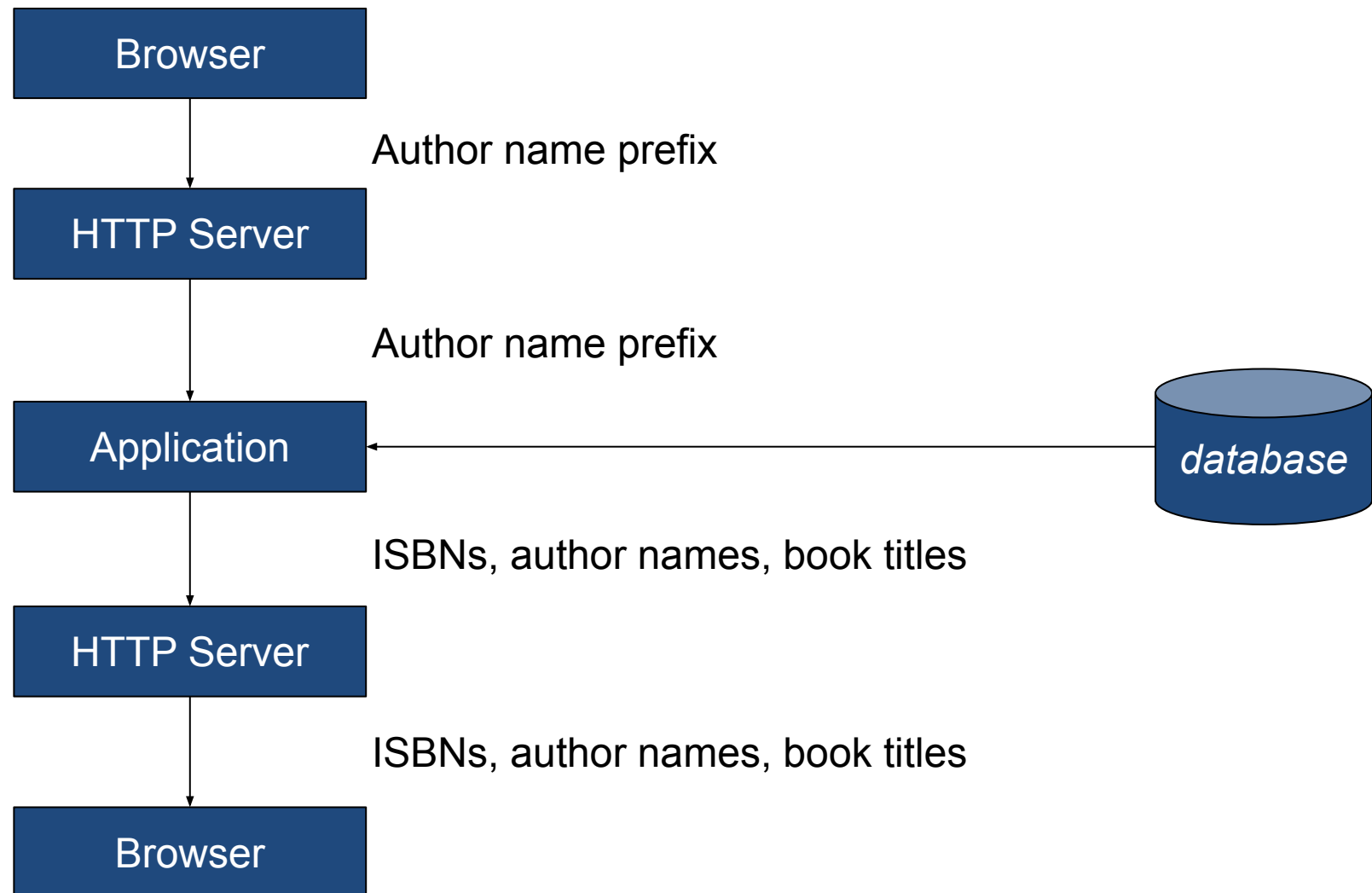
- **Fundamental example**
- Stateful web programming
- Stateful web programming with cookies

Fundamental Example

- **Penny app**
 - Website for a very small bookstore
 - We'll see *many* versions

Fundamental Example

Penny app



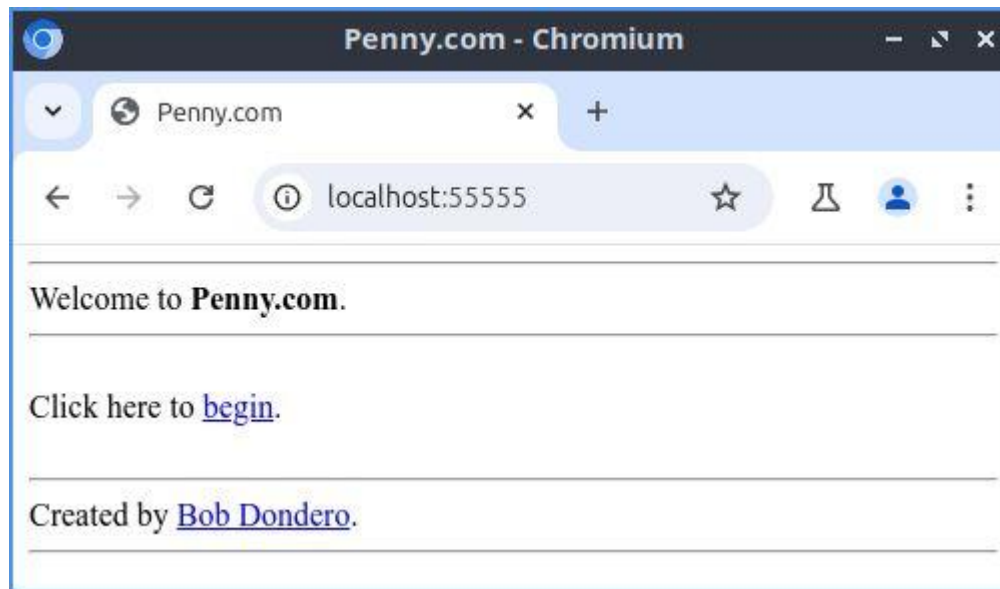
Fundamental Example

- See **PennyCgi** app

```
$ python runserver.py 55555  
Serving HTTP on 0.0.0.0 port 55555  
(http://0.0.0.0:55555/) ...
```

Fundamental Example

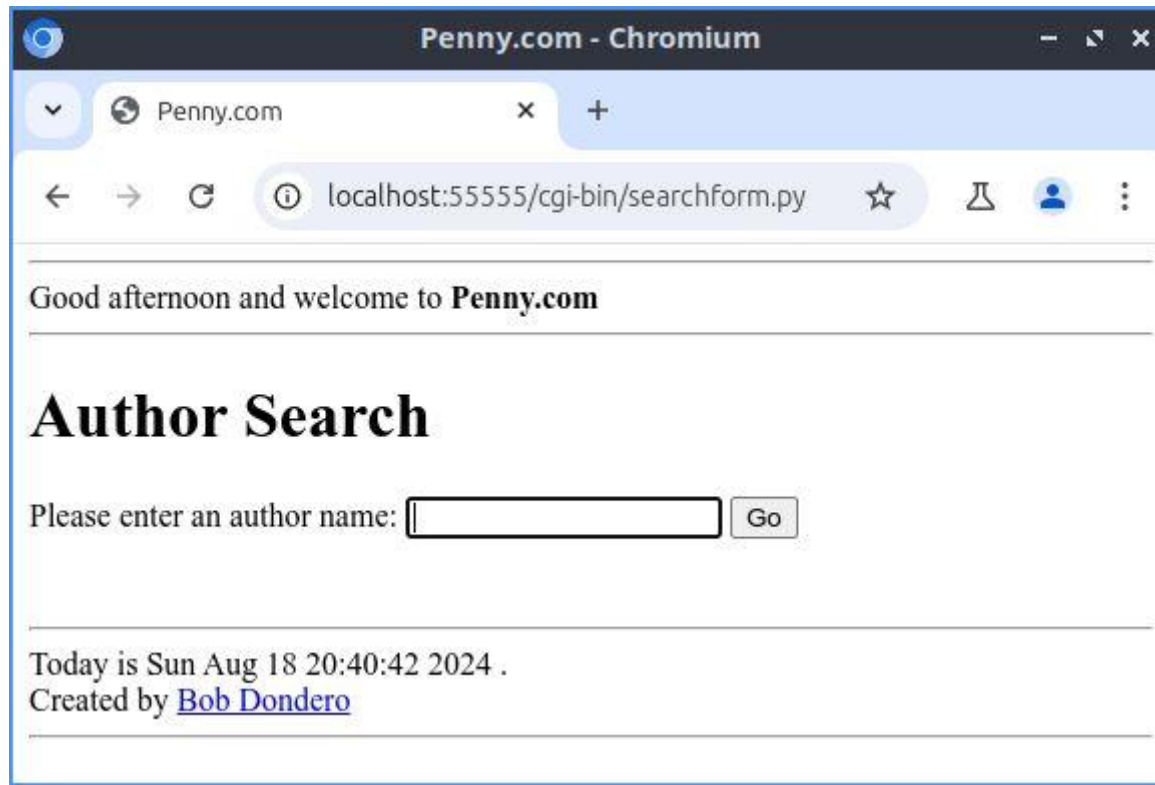
- See **PennyCgi** app (cont.)



The
index
page

Fundamental Example

- See **PennyCgi** app (cont.)



The
searchform
page

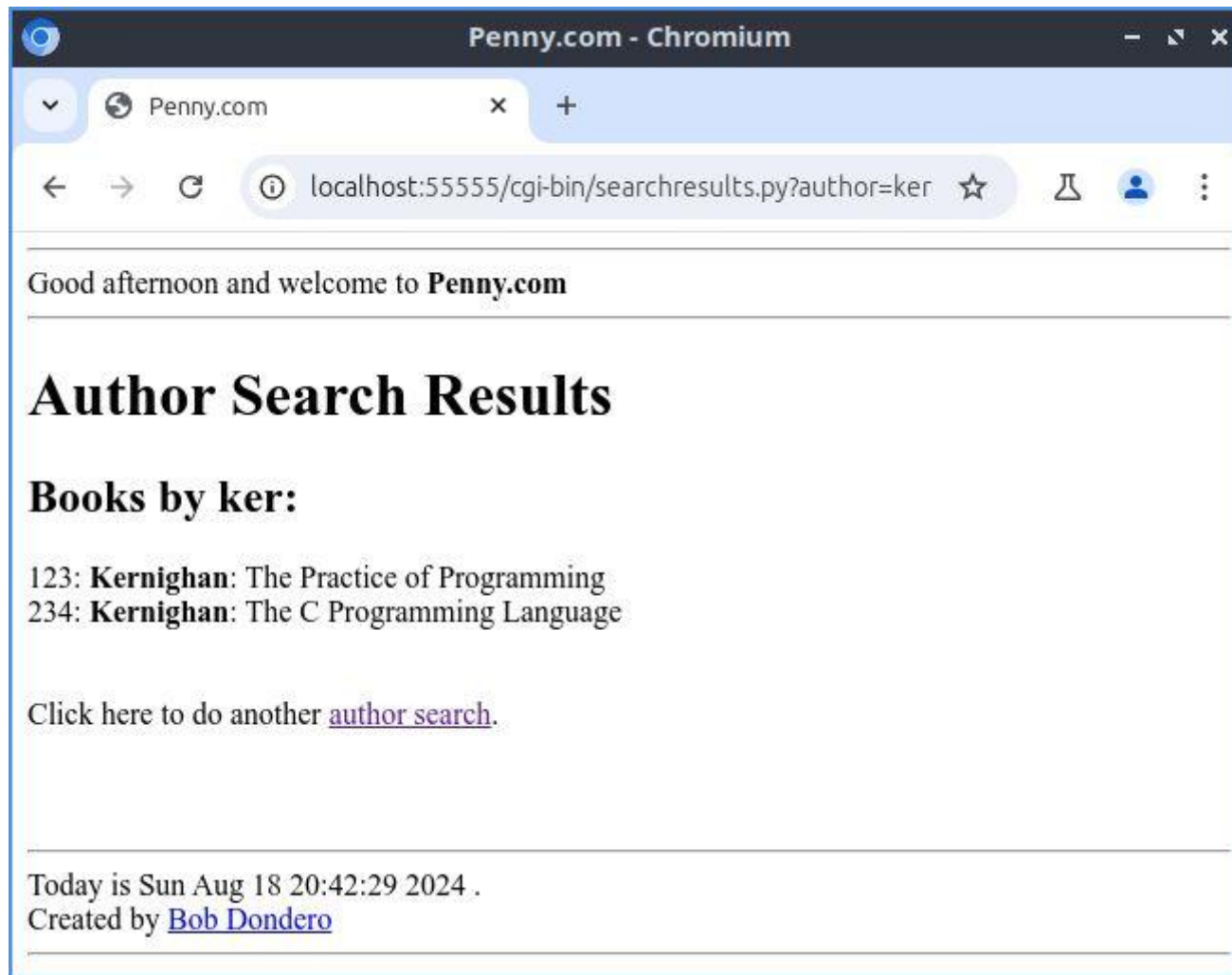
Fundamental Example

- See **PennyCgi** app (cont.)



Fundamental Example

- See PennyCgi app (cont.)



The
searchresults
page

Fundamental Example

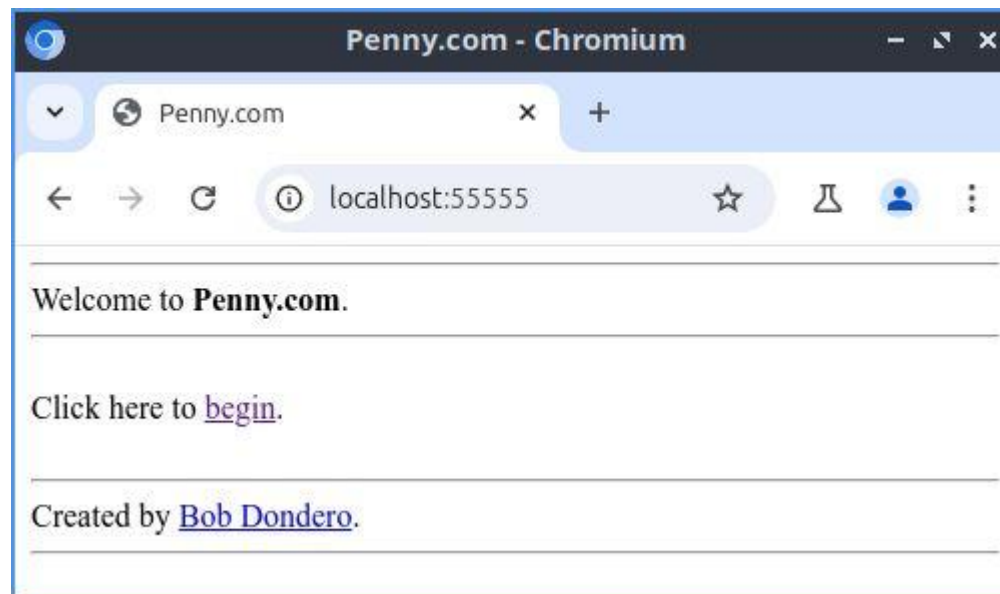
- See **PennyCgi** app (cont.)
 - runserver.py
 - **penny.sql**
 - penny.sqlite
 - **index.html**
 - **cgi-bin/database.py**
 - **cgi-bin/common.py**
 - cgi-bin/parseargs.py
 - **cgi-bin/searchform.py**
 - **cgi-bin/searchresults.py**

Agenda

- Fundamental example
- **Stateful web programming**
- Stateful web programming with cookies

Stateful Web Programming

- See **PennyCgiState** app



Stateful Web Programming

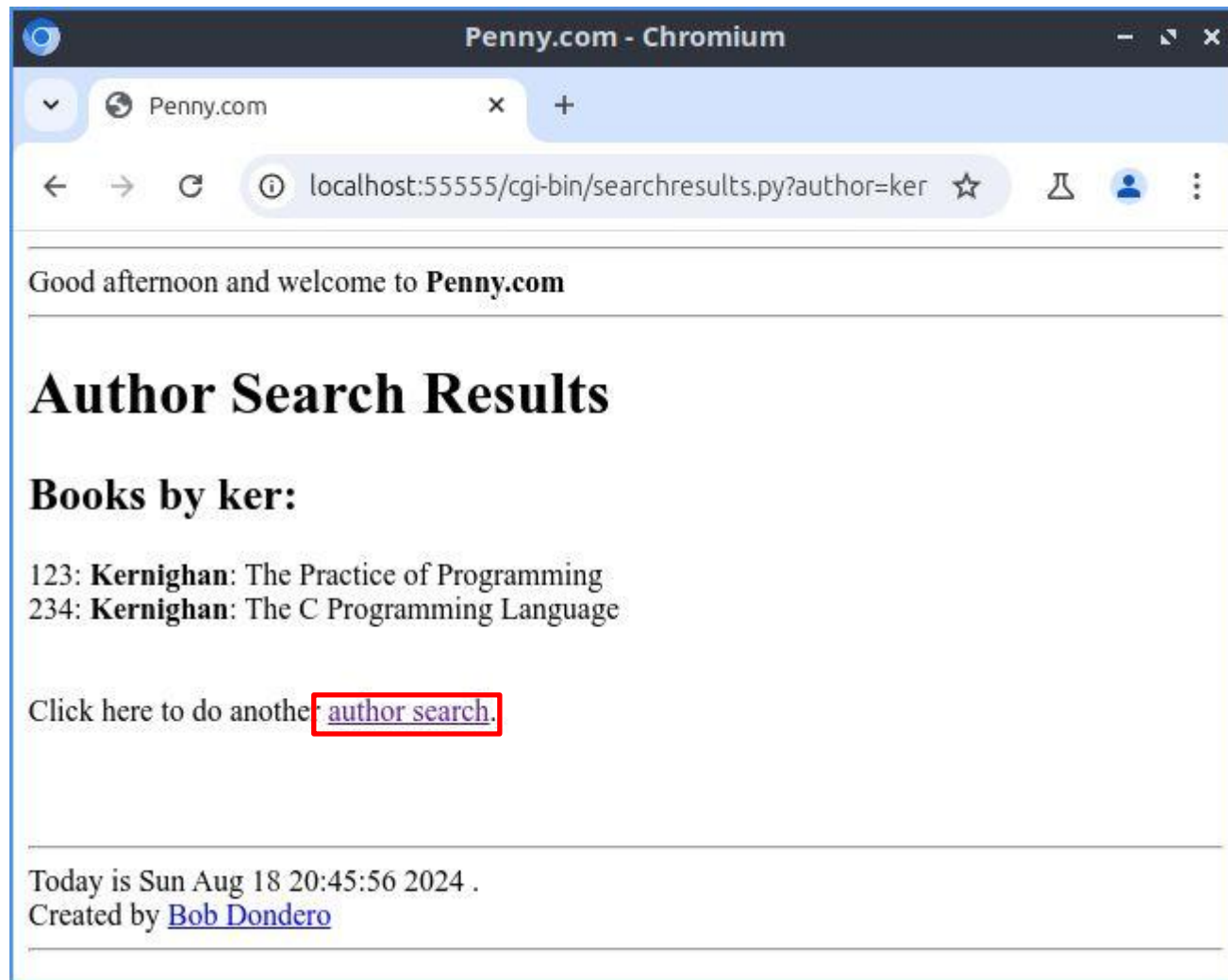
- See **PennyCgiState** app (cont.)



What's
different?

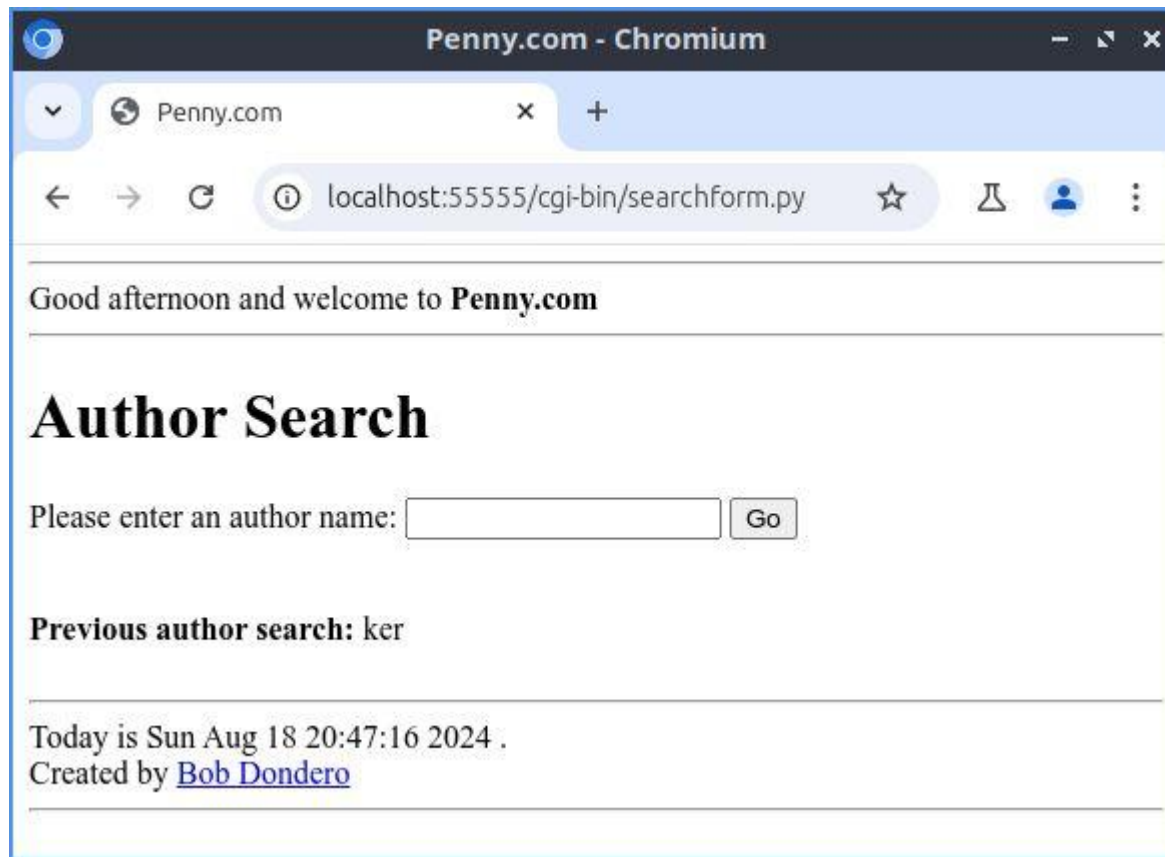
Stateful Web Programming

- See **PennyCgiState** app (cont.)



Stateful Web Programming

- See **PennyCgiState** app (cont.)

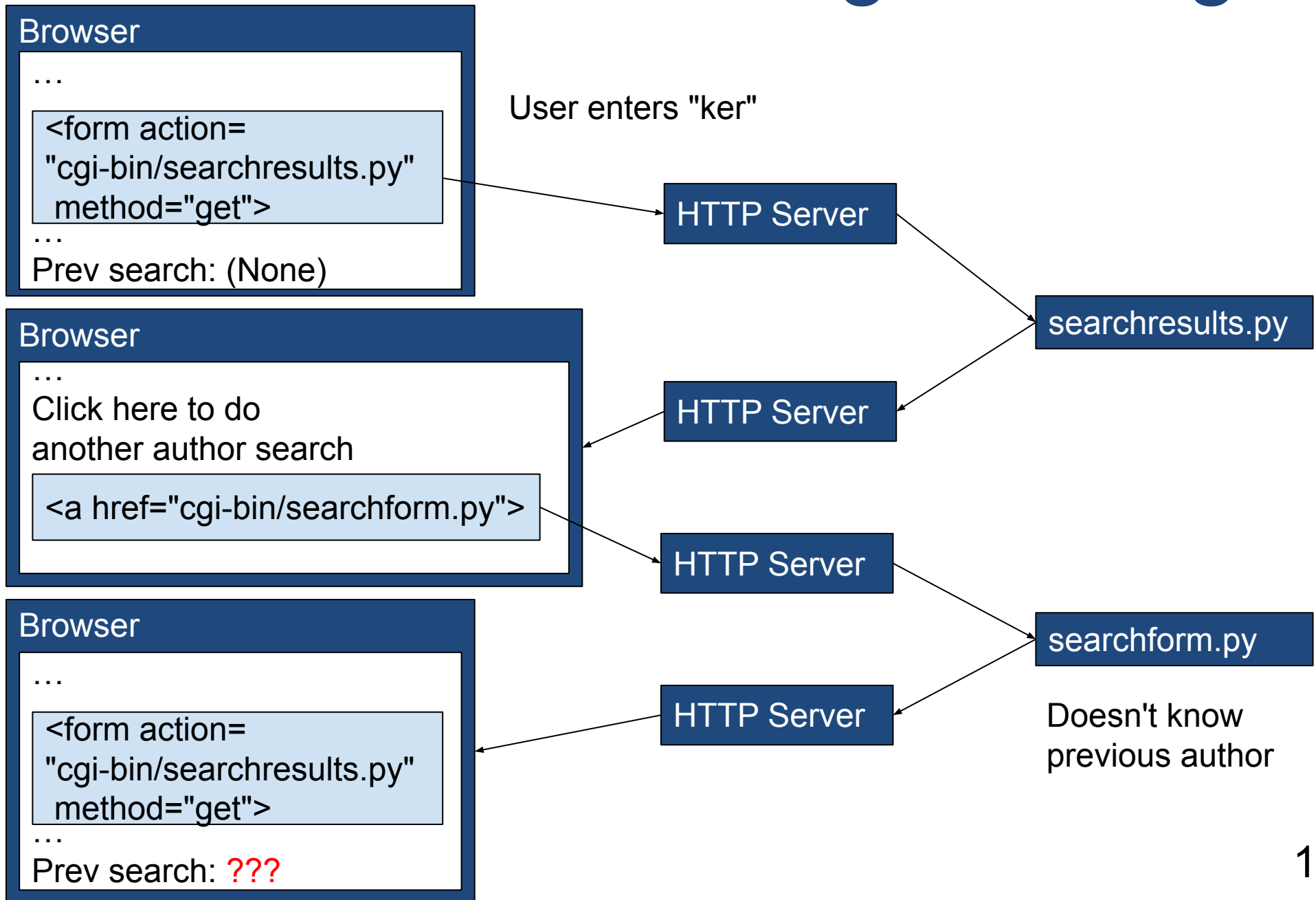


What's
different?

Stateful Web Programming

- See **PennyCgiState** app (cont.)
 - Displays name of previously-searched-for author in searchform page
 - But how???

Stateful Web Programming



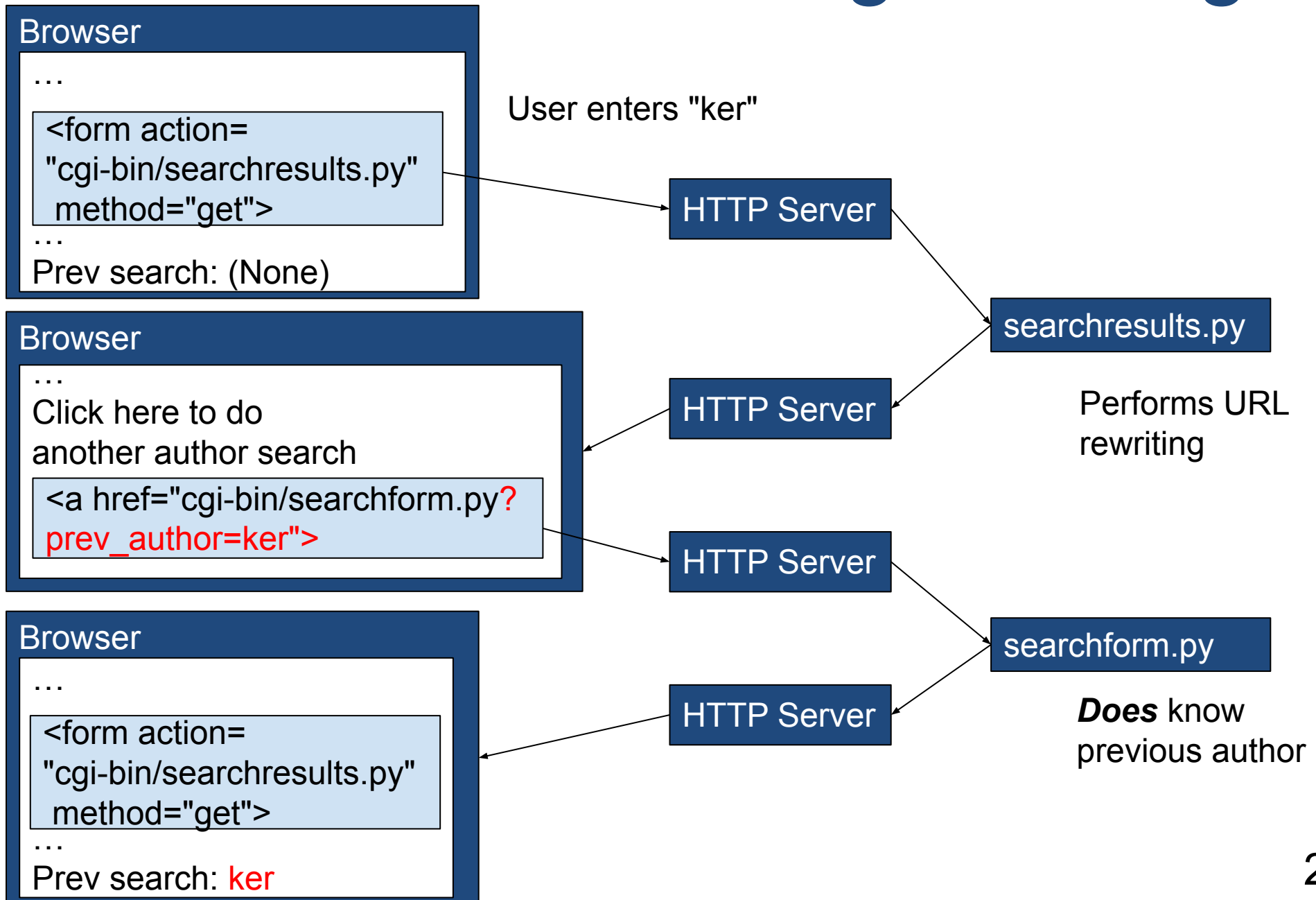
Stateful Web Programming

- Generalizing...
- **Problem:**
 - HTTP is a *stateless* protocol
 - Neither the browser nor the HTTP server remembers previous interactions

Stateful Web Programming

- **Problem:**
 - HTTP is a stateless protocol
- **Solution 1: *URL rewriting***
 - Append state data to end of URL

Stateful Web Programming



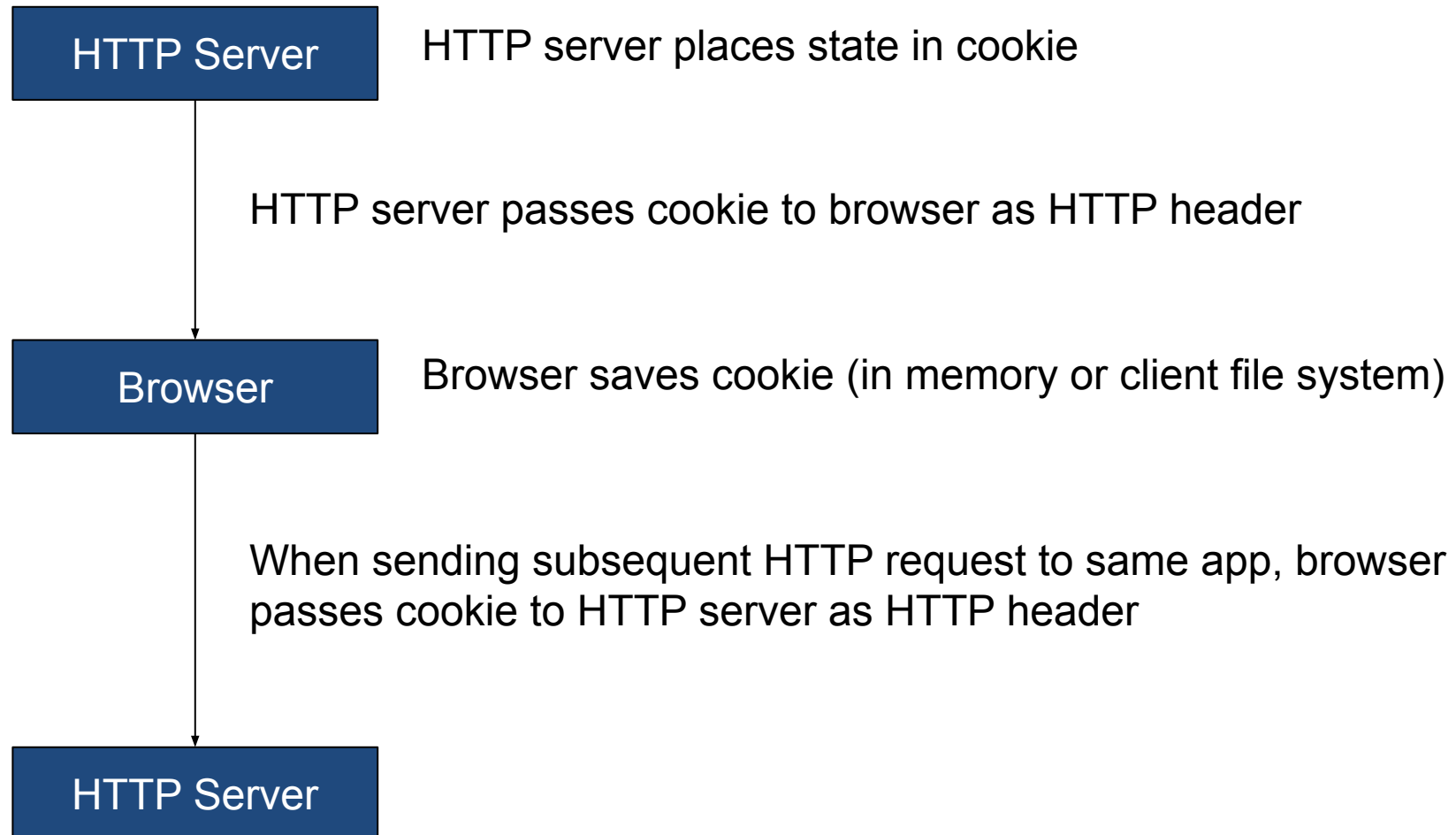
Agenda

- Fundamental example
- Stateful web programming
- **Stateful web programming with cookies**

Stateful Web Pgmming: Cookies

- **Problem:**
 - HTTP is a stateless protocol
- **Solution 1: URL rewriting**
 - Append state data to end of URL
- **Solution 2: *Cookies***

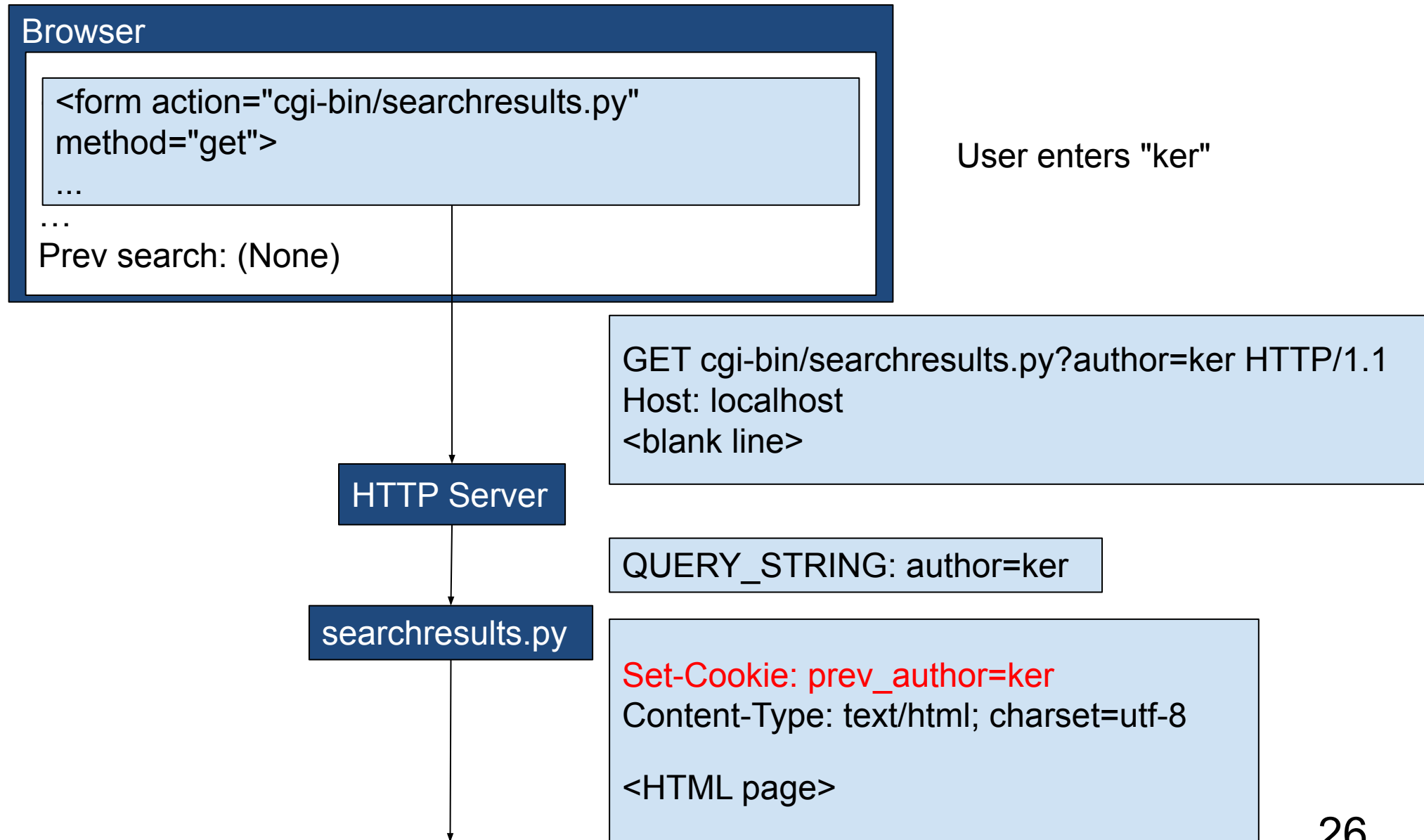
Stateful Web Pgmming: Cookies



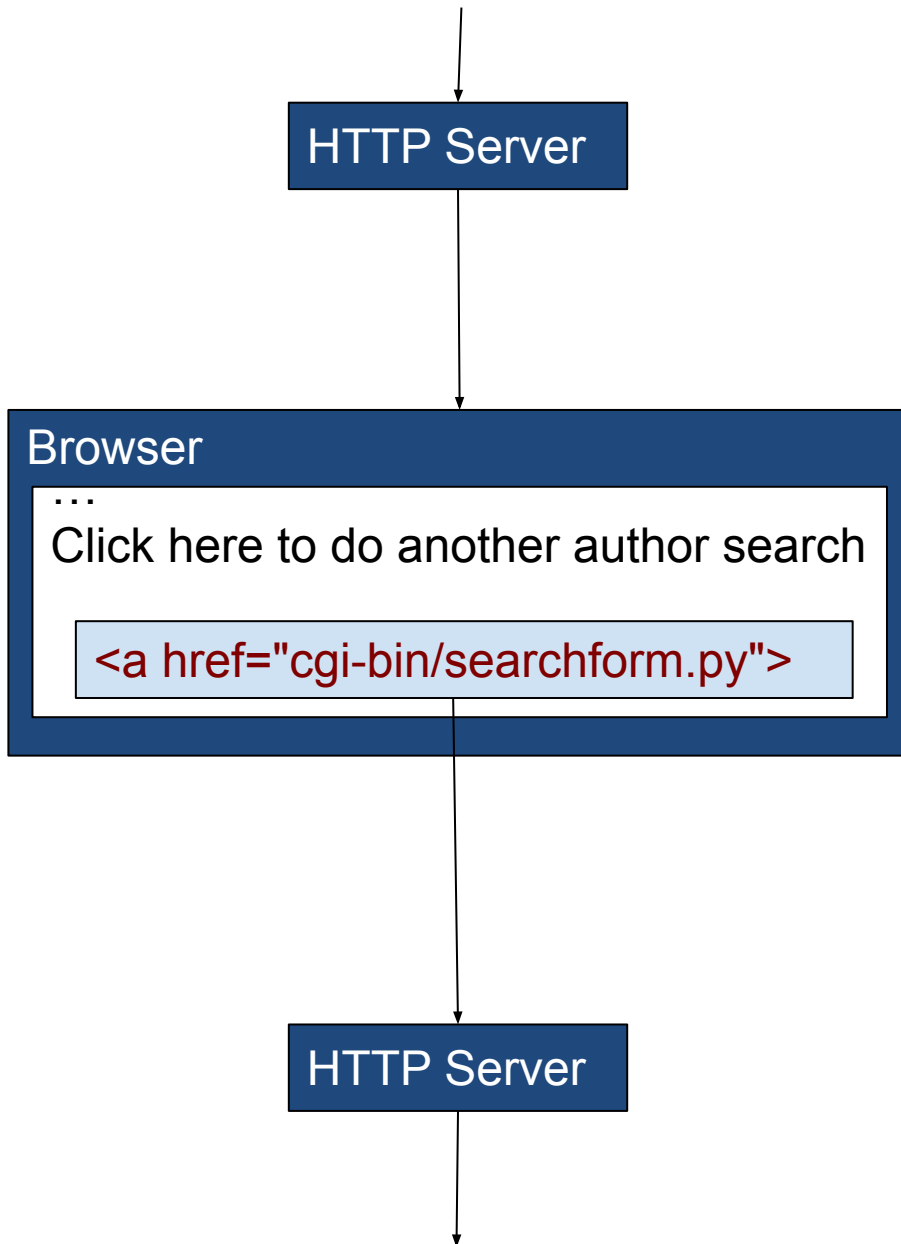
Stateful Web Pgmming: Cookies

- Cookie attributes:
 - Name
 - Content
 - Host & path
 - Expiration date
 - ...

Stateful Web Programming: Cookies



Stateful Web Pgmming: Cookies



```
HTTP/1.1 200 OK
Date: date
Server: localhost
...
Set-Cookie: prev_author=ker
Content-Type: text/html; charset=utf-8

<HTML page>
```

Browser saves
cookie

```
GET cgi-bin/searchform.py HTTP/1.1
Host: localhost
Cookie: prev_author=ker
<blank line>
```

```
HTTP_COOKIE: prev_author=Ker
```

Stateful Web Pgmming: Cookies

searchform.py

Knows previous author via HTTP_COOKIE env var

Content-type: text/html; charset=utf-8

<HTML page containing ker>

HTTP Server

HTTP/1.1 200 OK

Date: *date*

Server: localhost

...

Content-Type: text/html; charset=utf-8

<HTML page containing ker>

Browser

...
<form action="cgi-bin/searchresults.py" method="get">
...

Prev search: ker

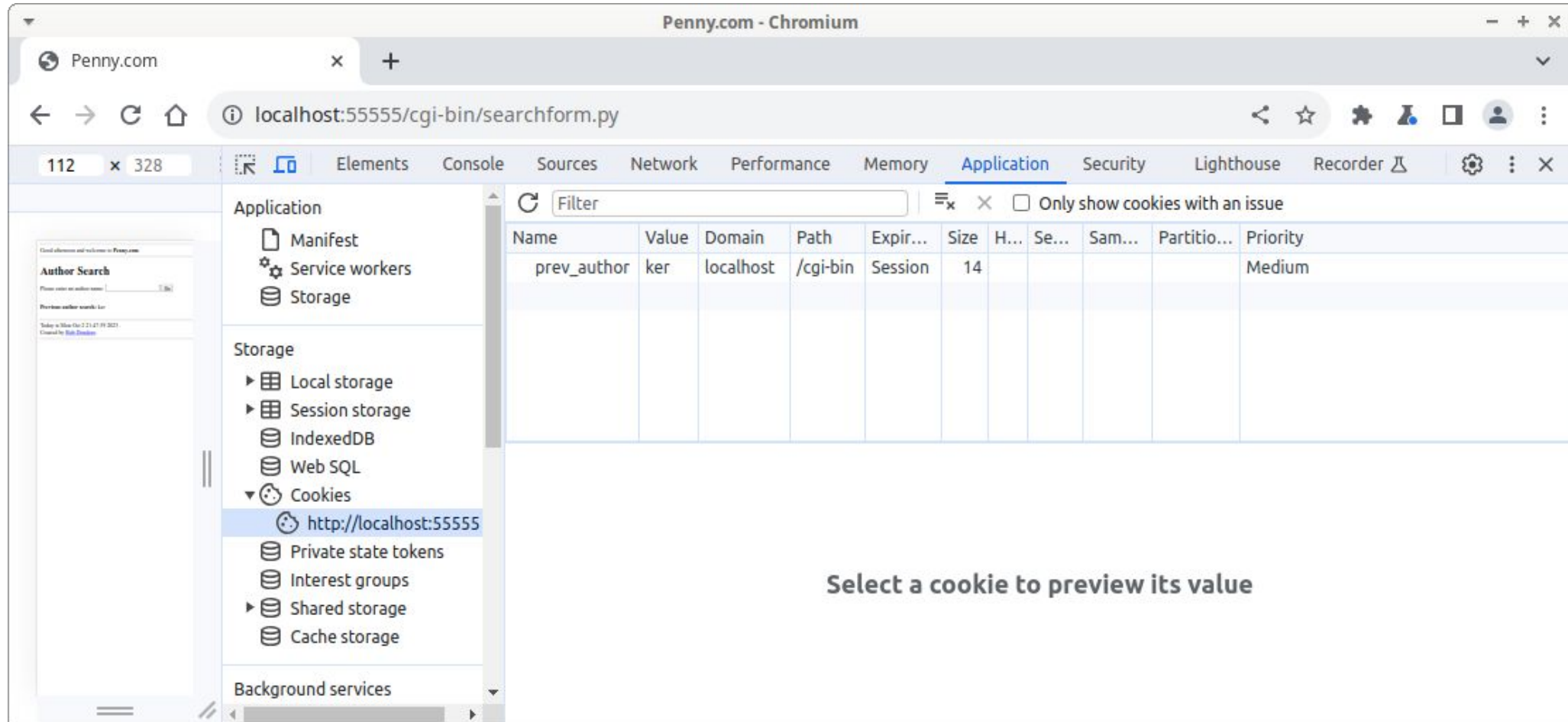
Stateful Web Pgmming: Cookies

- See **PennyCgiState** app
 - runserver.py
 - penny.sql
 - penny.sqlite
 - index.html
 - cgi-bin/database.py
 - cgi-bin/parseargs.py
 - **cgi-bin/searchresults.py**
 - **cgi-bin/searchform.py**

Stateful Web Pgmming: Cookies

- Viewing cookies in Chrome:
 - Browse to a page in the Penny application after the cookie has been created
 - From the menu at the upper right...
 - More tools →
 - Developer tools →
 - Application →
 - Cookies →
 - Select `http://localhost:55555`

Stateful Web Pgmming: Cookies



The screenshot shows a web browser window titled "Penny.com - Chromium" with the address bar displaying "localhost:55555/cgi-bin/searchform.py". The "Application" tab is selected in the developer tools, showing a tree view of storage areas. Under "Storage", the "Cookies" folder is expanded, showing a list of cookies for the domain "http://localhost:55555". The "prev_author" cookie is selected, and its details are shown in the table below.

Name	Value	Domain	Path	Expir...	Size	H...	Se...	Sam...	Partitio...	Priority
prev_author	ker	localhost	/cgi-bin	Session	14					Medium

Select a cookie to preview its value

Stateful Web Pgmming: Cookies

- Viewing cookies in Firefox:
 - Browse to a page in the Penny application after the cookie has been created
 - From the menu at the upper right...
 - More tools →
 - Web Developer Tools →
 - Storage →
 - Cookies →
 - Select `http://localhost:55555`

Stateful Web Pgmming: Cookies

Penny.com — Mozilla Firefox

File Edit View History Bookmarks Tools Help

Penny.com

localhost:5555/cgi-bin/searchform.py

Getting Started CS Department Web ... COS217 COS333 IntroCsBook IndepWork Princeton Health Linux Mint Other Bookmarks

Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name:

Inspector Console Debugger Network Style Editor Performance Memory **Storage** Accessibility Application

Cache Storage Cookies **http://localhost:5555** Indexed DB Local Storage Session Storage

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
prev_auth...	ker	localhost	/cgi-bin	Session	14	false	false

Filter values

prev_author:"ker"
Created:"Tue, 03 Oct 2023 01:53:17 GMT"
Domain:"localhost"
Expires / Max-Age:"Session"
HostOnly:true

Lecture Summary

- In this lecture we covered:
 - A fundamental example
 - Stateful web programming

Lecture Series Summary

- In this lecture series we covered:
 - CGI programming
 - CGI using the HTTP GET method
 - CGI using the HTTP POST method
 - GET vs. POST
 - A fundamental example
 - Stateful web programming
- See also:
 - **Appendix 1: Cookie Problems**
 - **Appendix 2: Python Decorators**

More Information

- The COS 333 *Lectures* web page provides references to supplementary information

Appendix 1: Cookie Problems

Cookie Problems

- **Problem 1:**
 - Cookie size is limited to 4K bytes
- **Solution:**
 - Cookie content stored on server-side (in database), indexed by a unique key
 - Cookie contains key only

Cookie Problems

- **Problem 2:**

- Browser user may block cookies for *all* websites

- Chrome

- Settings → Privacy & Security → Site Settings → Cookies and site data → Block all cookies

- Firefox:

- Settings → Privacy & Security → Custom → Cookies → All Cookies

Cookie Problems

- **Problem 2 (cont.):**
 - Browser user may block cookies for *some* websites
 - Encouraged by the *European General Data Protection Regulation (GDPR)*

Cookie Problems

- **Problem 2 (cont.):**

The **GDPR legislation** requires all multinational companies to provide an opt-in whereby website owners receive a user's permission to use cookies before they can be stored on a user's web browsers...

<https://us.norton.com/blog/privacy/should-i-accept-cookies>

Cookie Problems

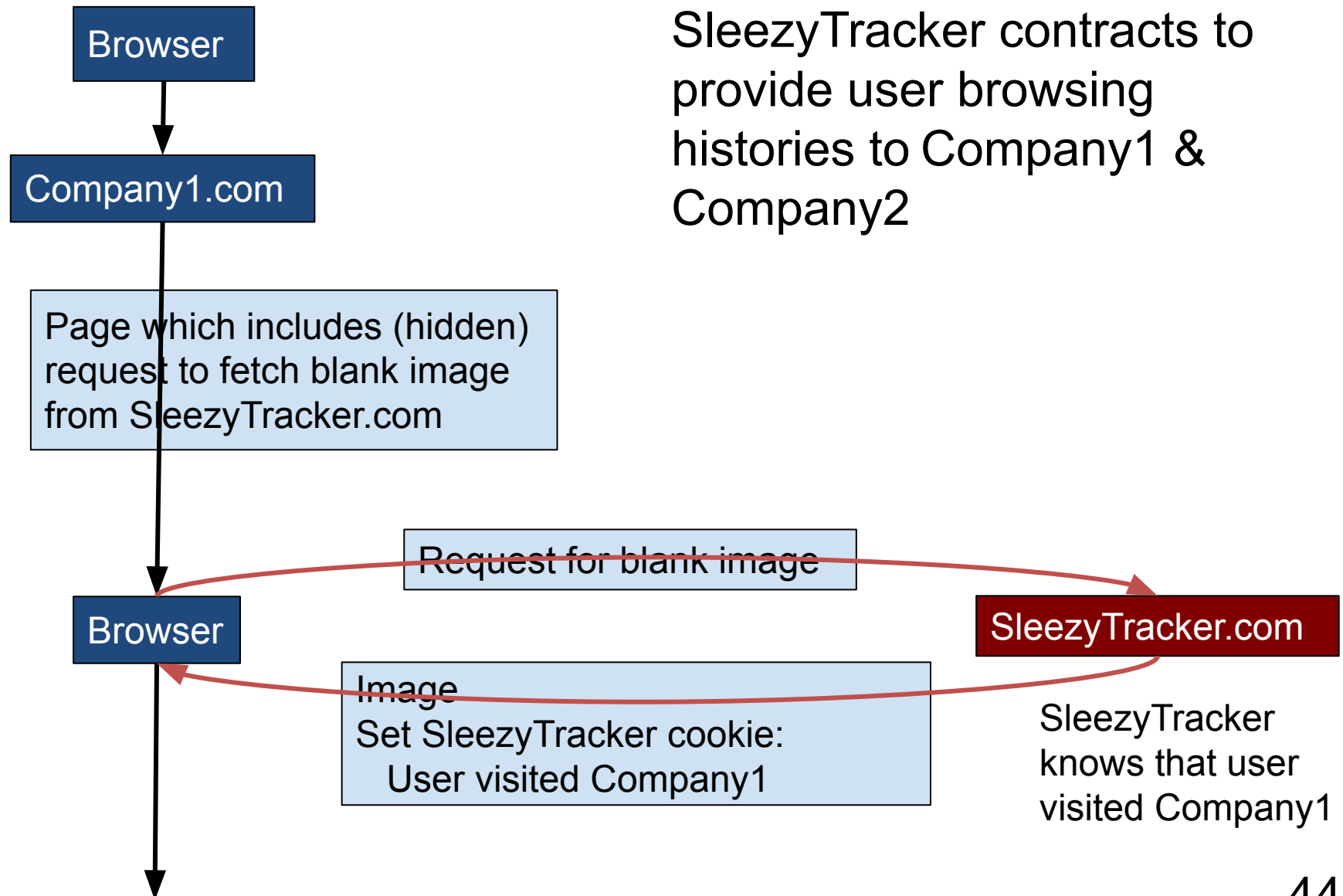
- **Solution:**
 - Ask the user to enable cookies! Or...
 - Add more logic
 - Use URL rewriting
 - Legal?

Cookie Problems

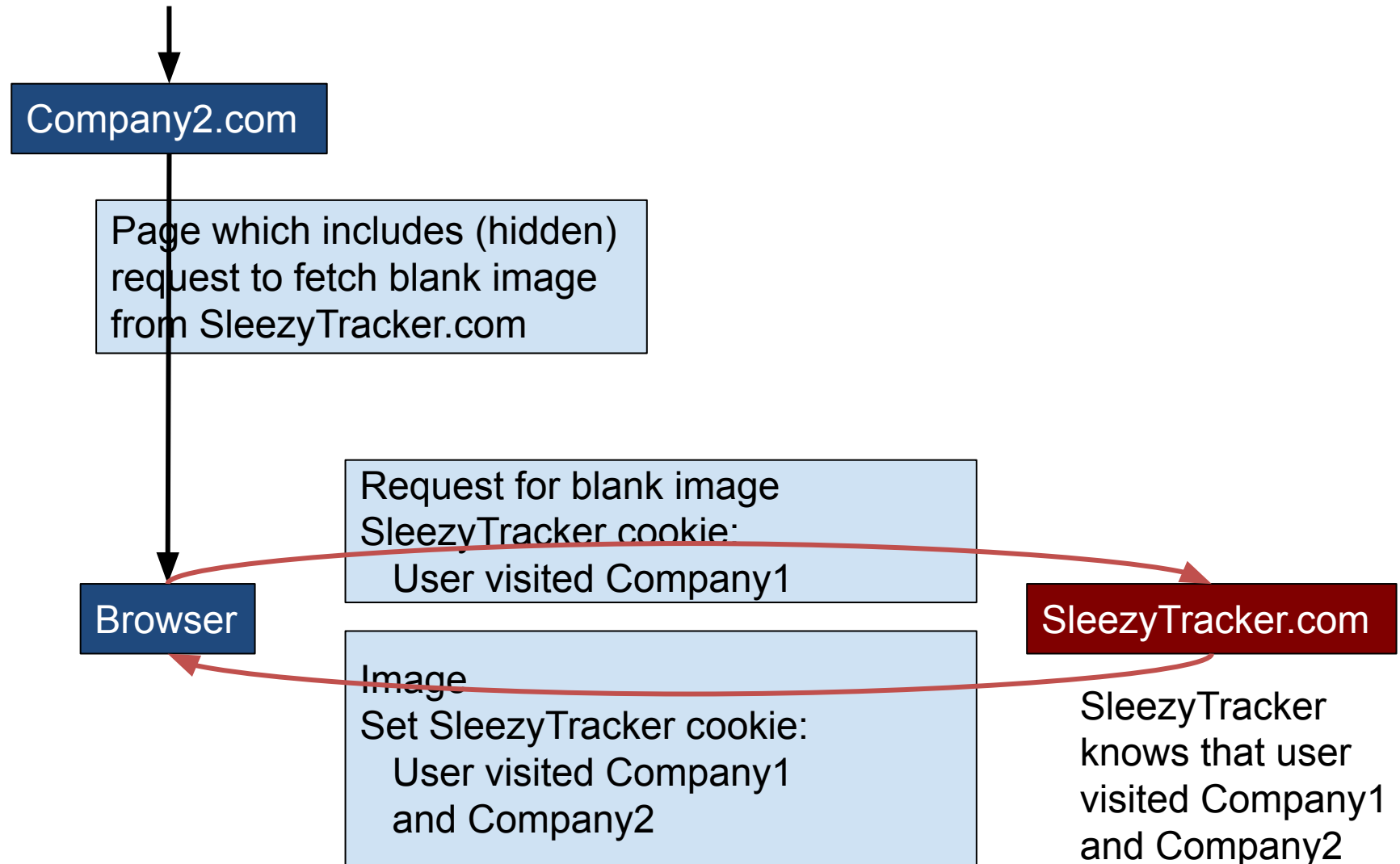
- **Problem 3:**

- *Third-party cookies* can invade privacy
 - See https://en.wikipedia.org/wiki/Cookie_stuffing

Cookie Problems



Cookie Problems



SleezyTracker provides user browsing histories to Company1 & Company2

Cookie Problems

- **Solution:** Tell browser to refuse third-party cookies
 - Chrome
 - Settings → Privacy & Security → Site Settings → Cookies and site data → Block third-party cookies
 - Firefox:
 - Settings → Privacy & Security → Custom → Cookies → Cross-site tracking cookies, and isolate other cross-site cookies

Appendix 2:

Python Decorators

Python Decorators

```
def sqr(i):  
    return i * i  
  
def main():  
    result = sqr(5)  
    print(result)  
  
if __name__ == '__main__':  
    main()
```

Wanted:

sqr() prints “sqr was called” each time it is called

Python Decorators

```
def sqr(i):  
    print('sqr was called')  
    return i * i  
  
def main():  
    result = sqr(5)  
    print(result)  
  
if __name__ == '__main__':  
    main()
```

OK, but...

Requires edit of def of `sqr()`

Python Decorators

One approach

```
def print_name_decorator(f):  
    def fwrapper(i):  
        print(f.__name__, 'was called')  
        return f(i)  
    return fwrapper  
  
def sqr(i):  
    return i * i  
  
sqr = print_name_decorator(sqr)  
# Defines fwrapper as this:  
#     def fwrapper(i):  
#         print('sqr', 'was called')  
#         return sqr(i)  
# and then does this:  
#     sqr = fwrapper  
  
def main():  
    result = sqr(5)  
    print(result)  
  
if __name__ == '__main__':  
    main()
```

Trace:

```
result = sqr(5)  
result = fwrapper(5)  
    fwrapper(5) prints 'sqr was called'  
    fwrapper(5) returns sqr(5)  
result = 25
```

Prints:

```
sqr was called  
25
```

Python Decorators

```
def print_name_decorator(f):  
    def fwrapper(i):  
        print(f.__name__, 'was called')  
        return f(i)  
    return fwrapper
```

```
@print_name_decorator
```

```
def sqr(i):  
    return i * i
```

```
def main():  
    result = sqr(5)  
    print(result)
```

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

Using a
decorator

Prints:

sqr was called
25