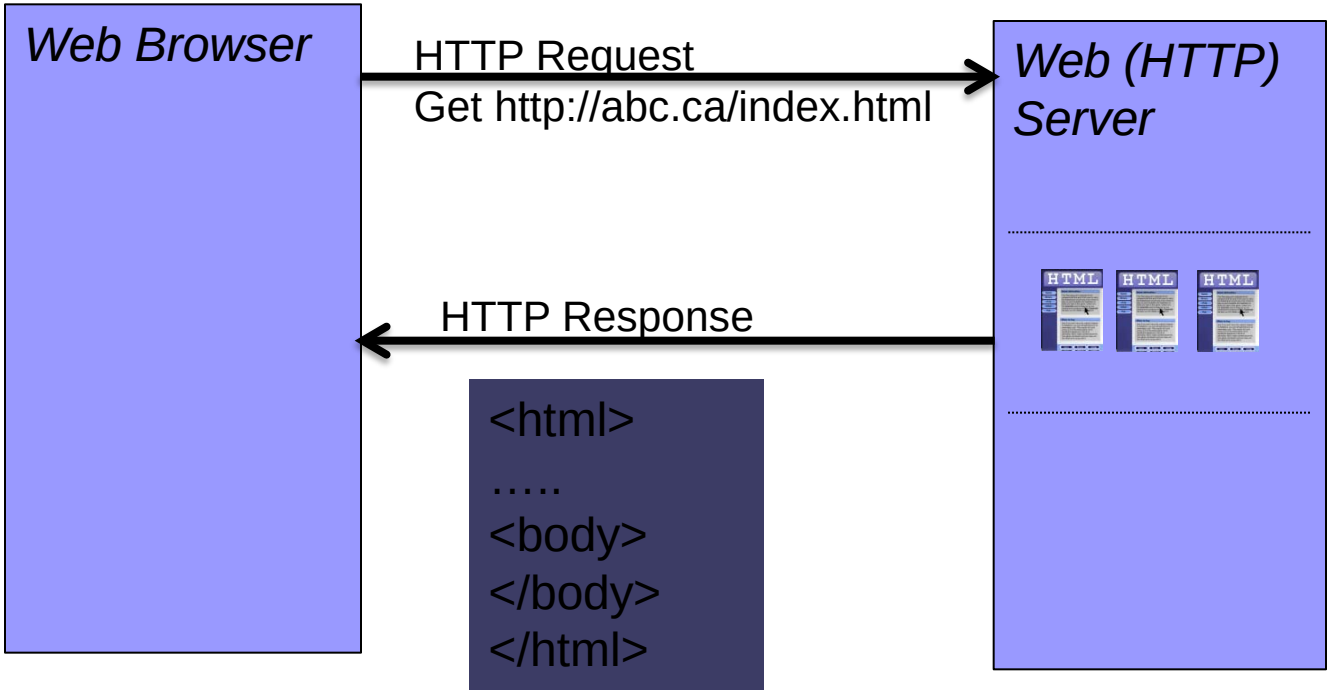
CSC309: Introduction to Web Programming

Lecture 8

Wael Aboulsaadat

Front Layer



How can we create an interactive experience?

Front Layer: what is an interactive experience?

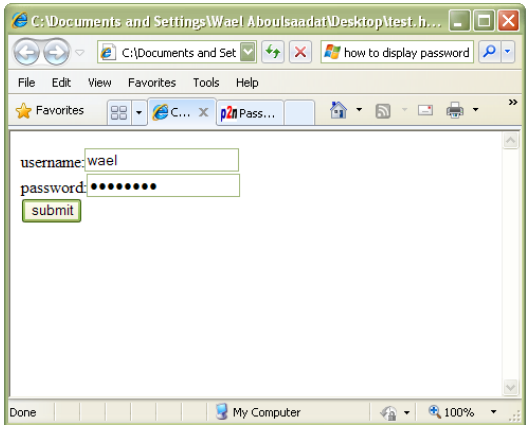
- Program output is a result of user actions
  - Can a web server respond differently based on user actions?
- Each user interacts with an application independently
  - How can we track a user ?

How can we send data from browser to web-server?

- Form

1) Get

```
<FORM ACTION="login" METHOD="get">
    username:<input type ="text" name ="username" value = "">
     password:<input type ="" name ="password" value = "">
                    <input type="submit" value="submit">
</FORM>
```



GET /login?username=
wael&password=1223
HTTP/1.1
Host: localhost

<html>……</html>

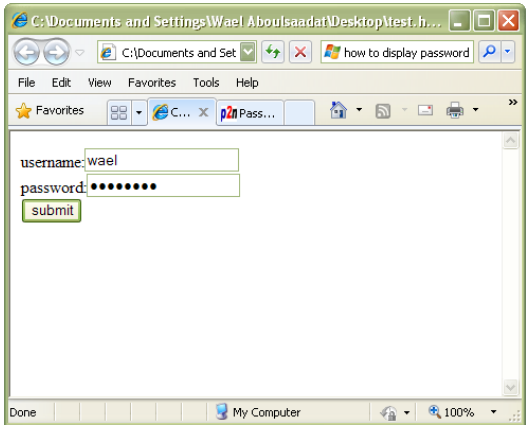
Web Server

How can we send data from browser to web-server?

- Form

1) Post

```
<FORM ACTION="login" METHOD="post">
    username:<input type ="text" name ="username" value = "">
     password:<input type ="" name ="password" value = "">
                <input type="submit" value="submit">
</FORM>
```



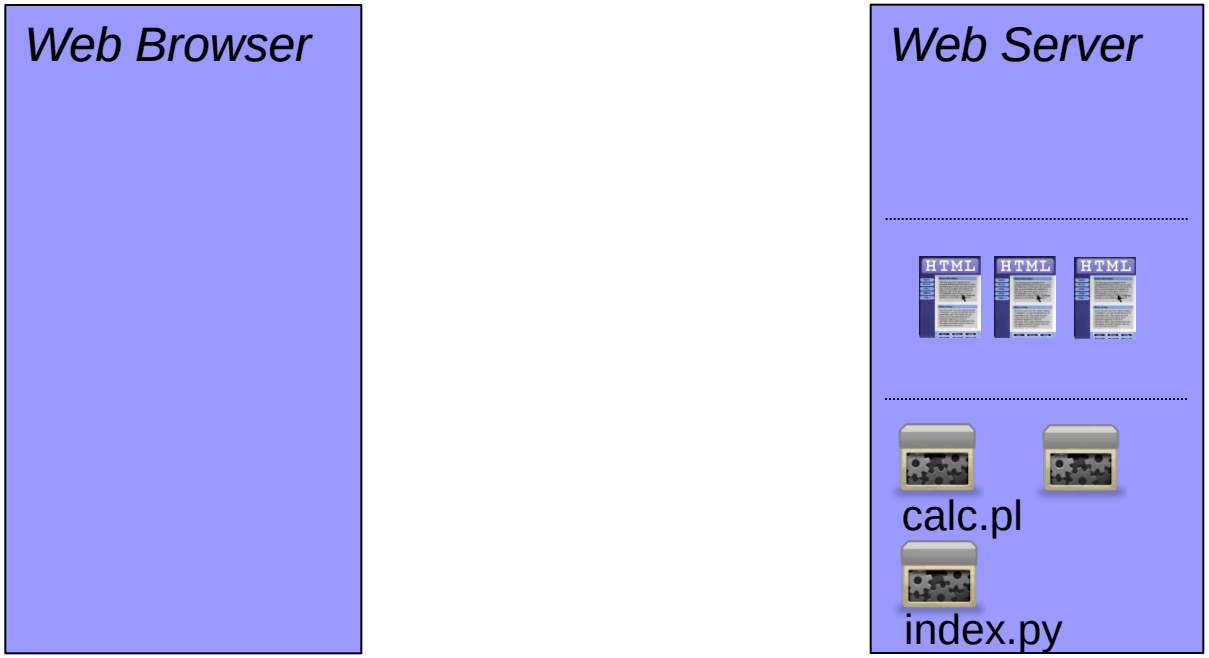
POST /login
HTTP/1.1
Host: localhost
……
username=wael&password
=1233

<html>……</html>

Web Server

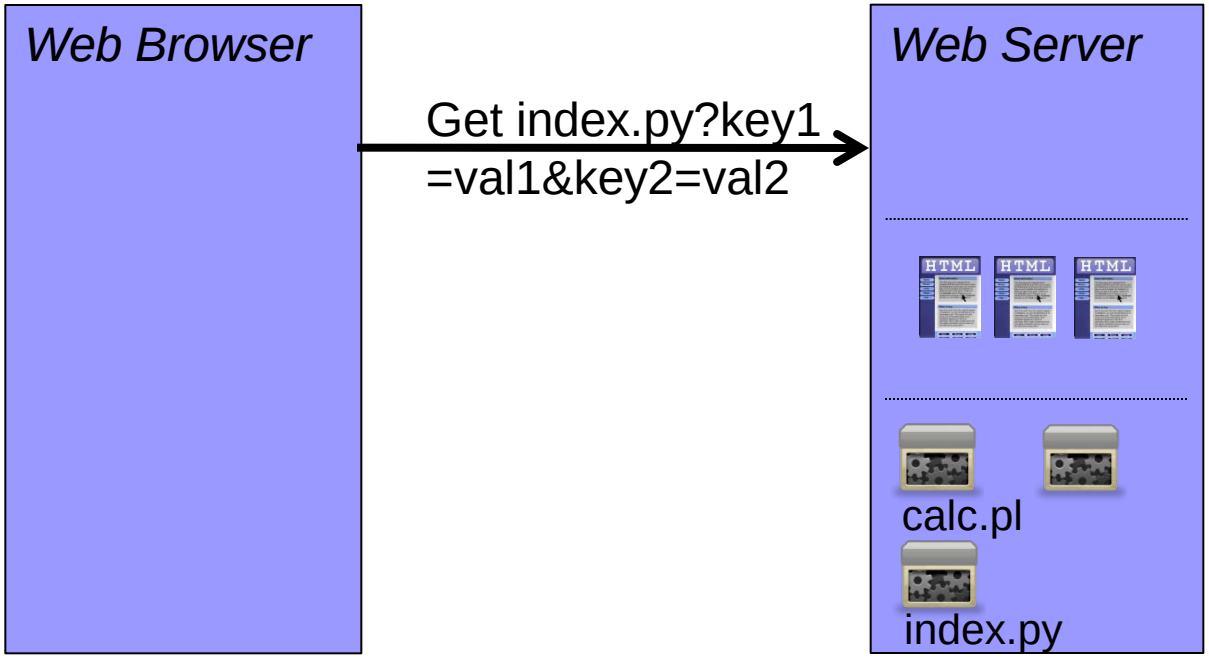
WebServer - WebApp Communication

1. CGI



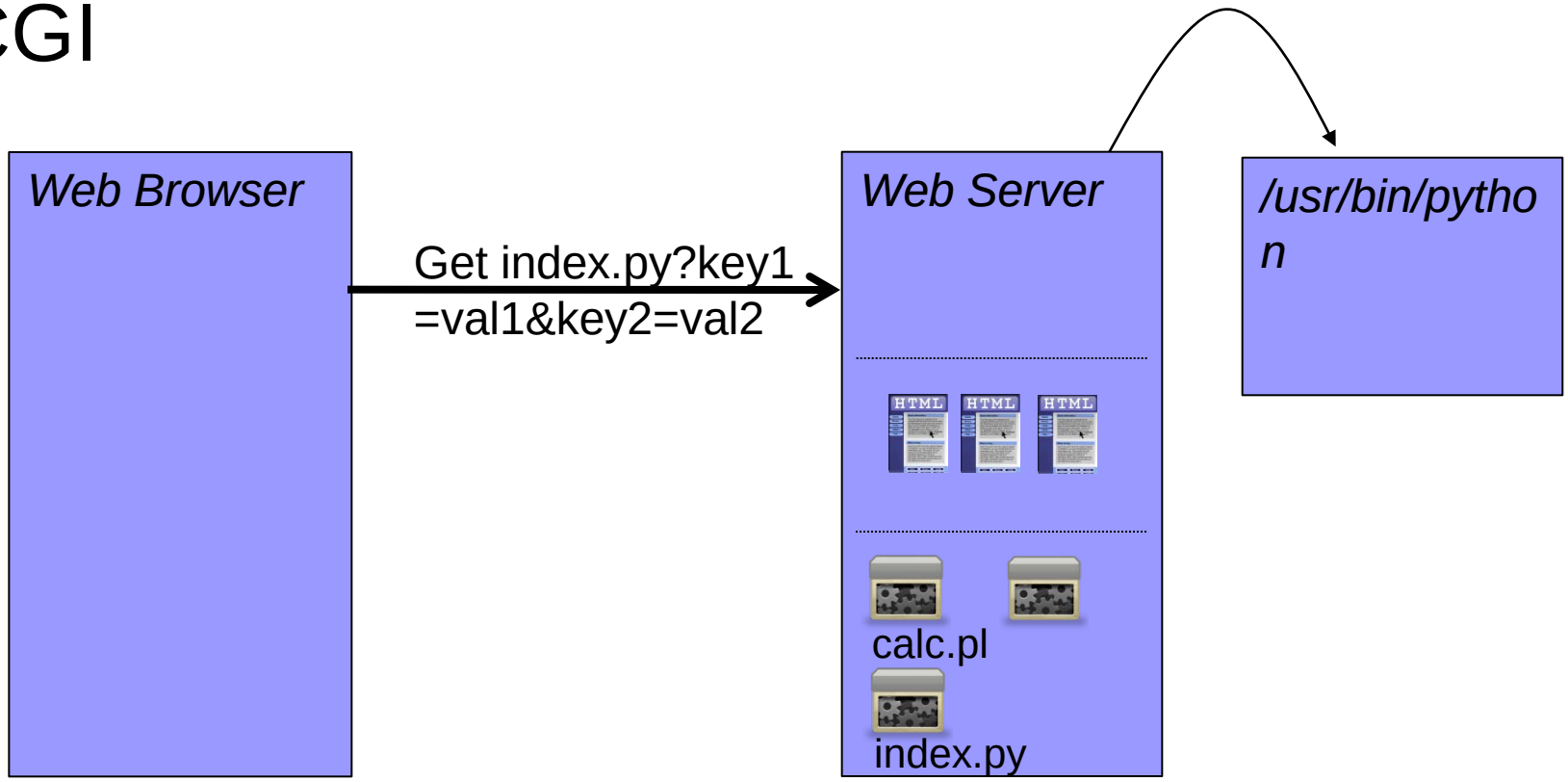
WebServer - WebApp Communication

1. CGI



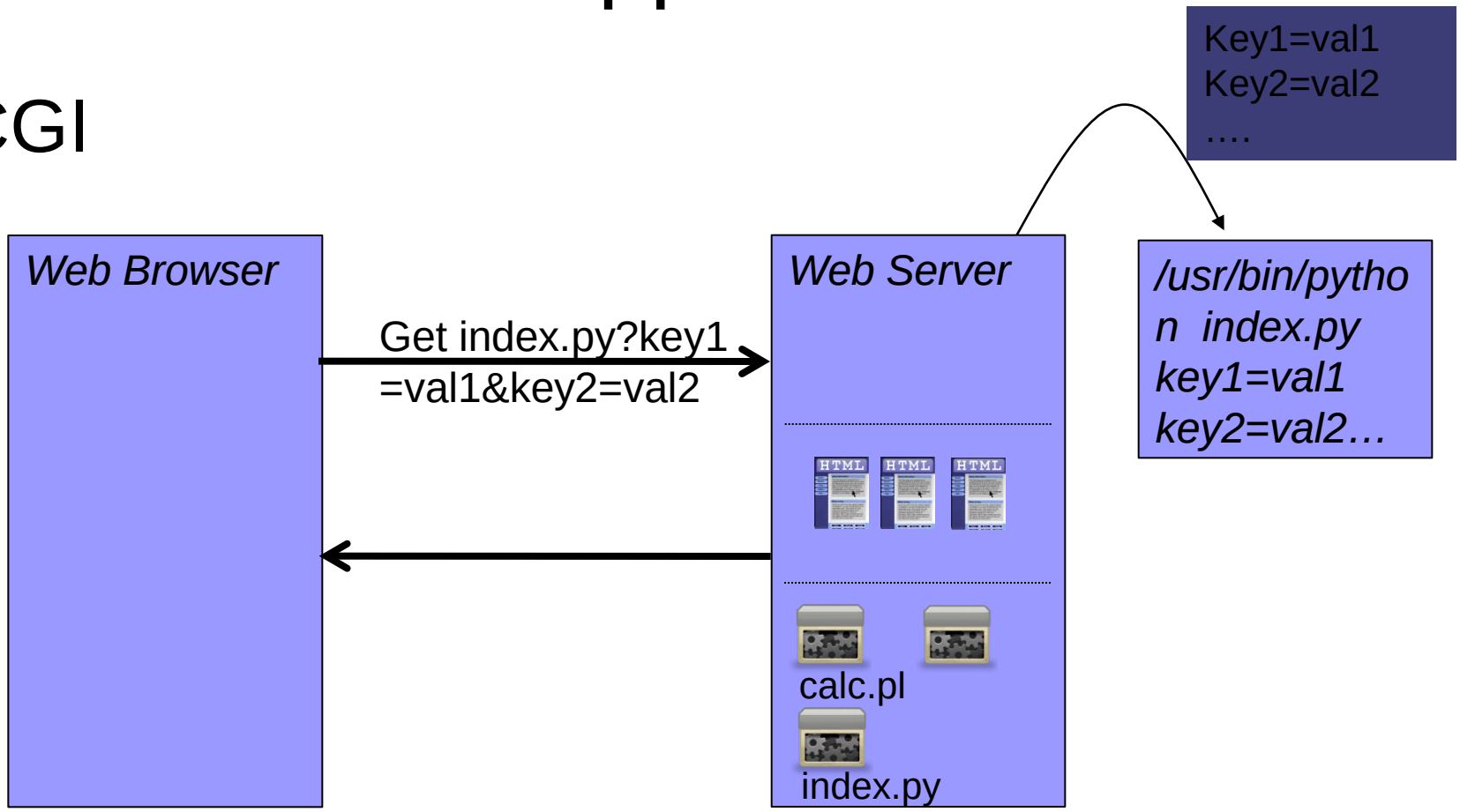
WebServer - WebApp Communication

1. CGI



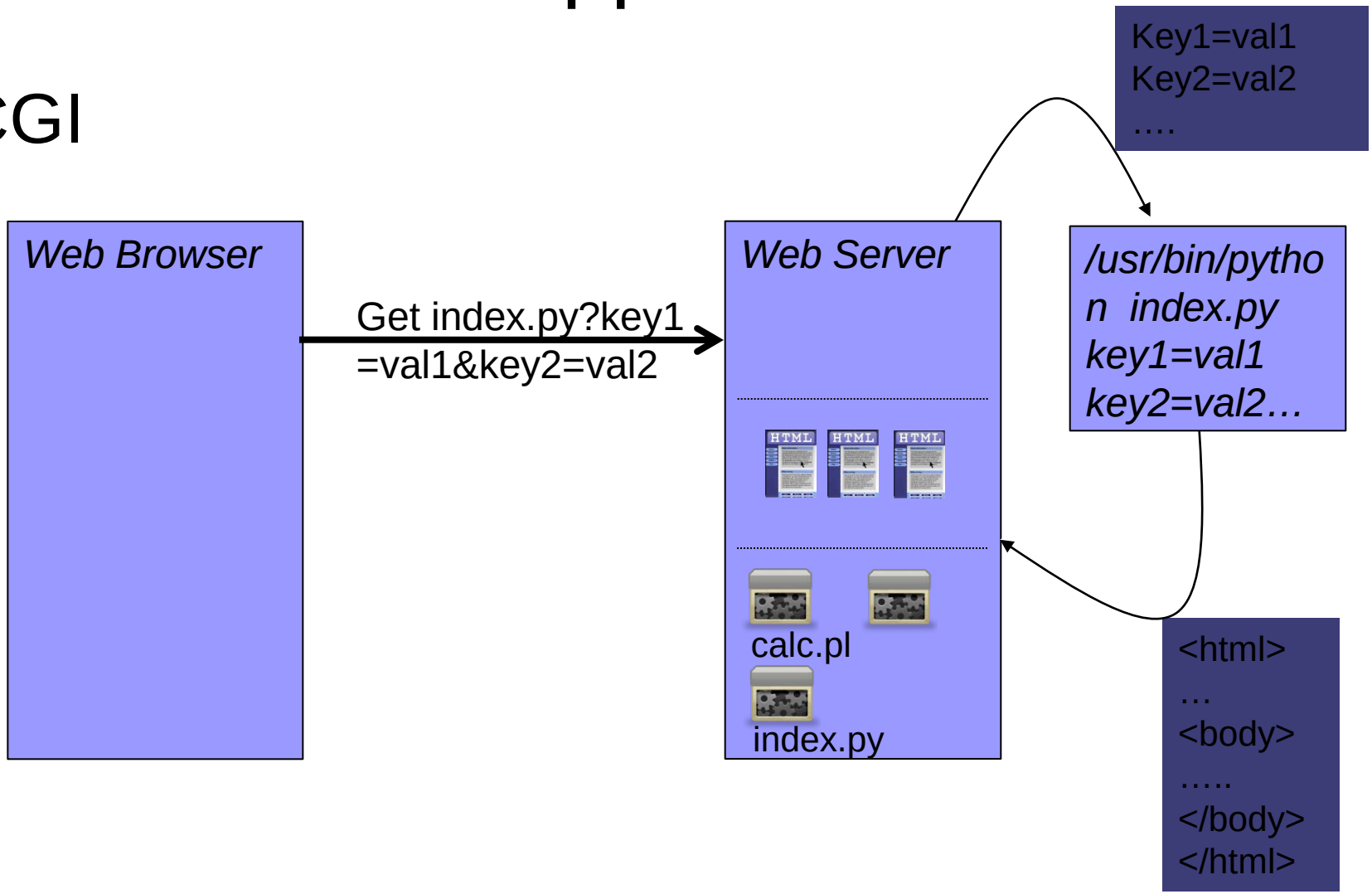
WebServer - WebApp Communication

1. CGI



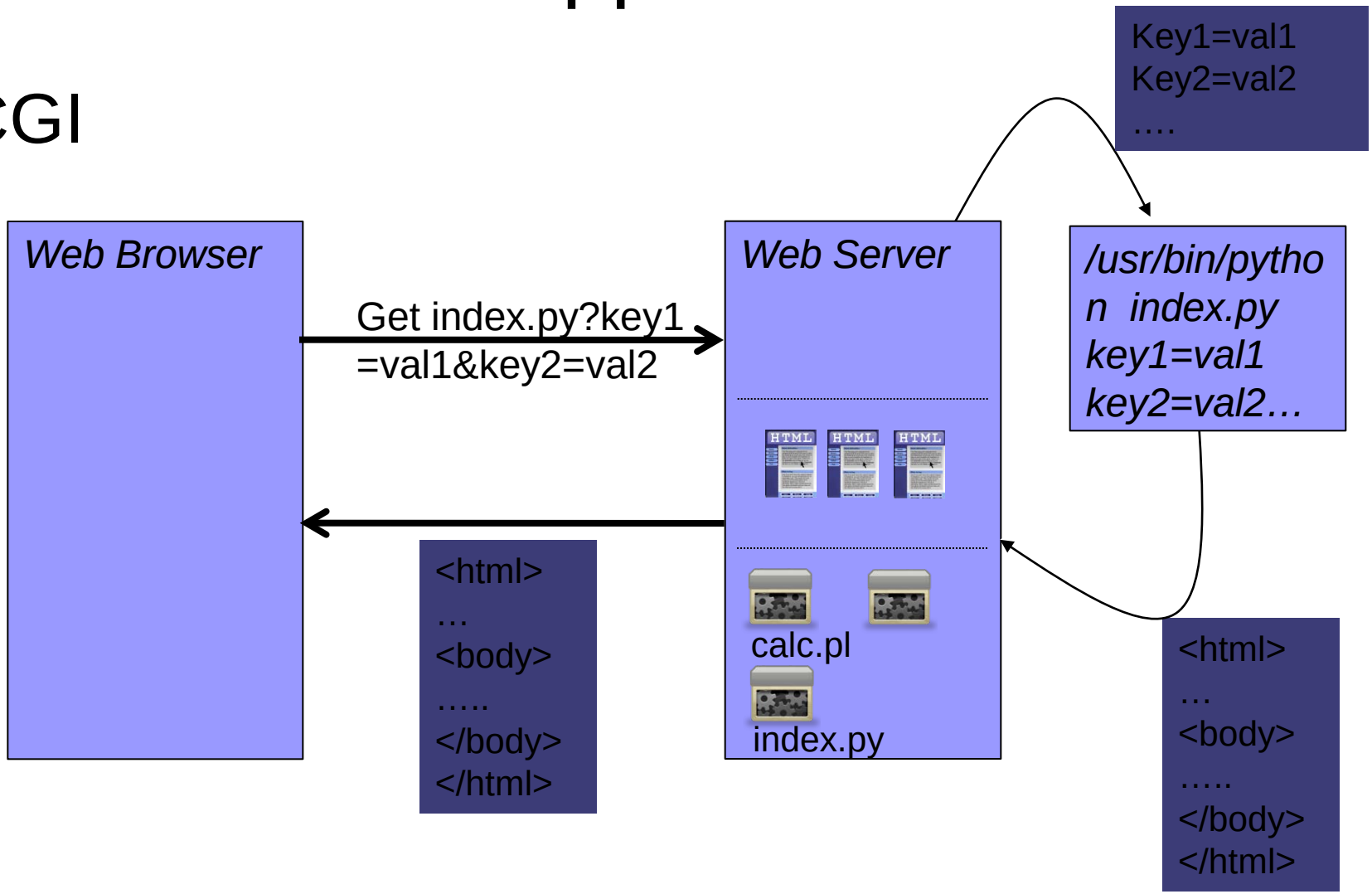
WebServer - WebApp Communication

1. CGI



WebServer - WebApp Communication

1. CGI



CGI: common gateway interface

- How is the communication between Webserver and cgi application happening?
  - via stdin & stdout

CGI: common gateway interface

- Pros
  - Any language could be used to develop the webapp
- Cons
  - Creating a process with every-request
  - Passing sizable data is a problem
  - No framework for webapp dev

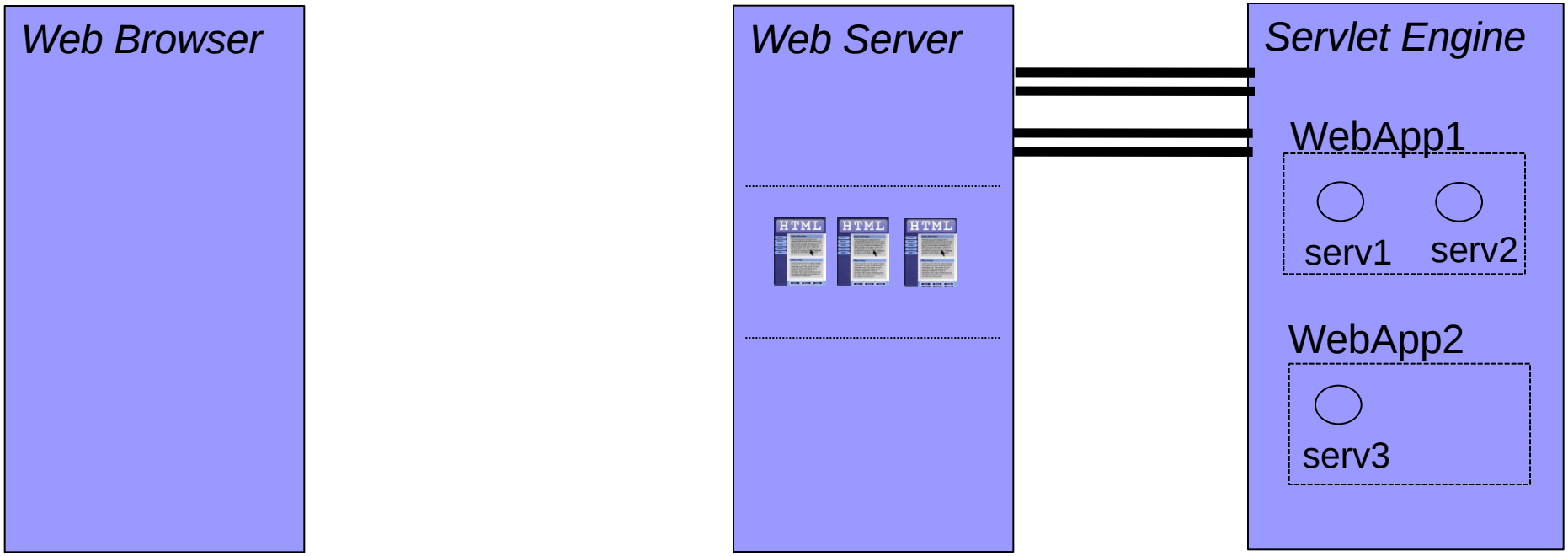
WebServer - WebApp Communication

1. CGI

2. Servlets

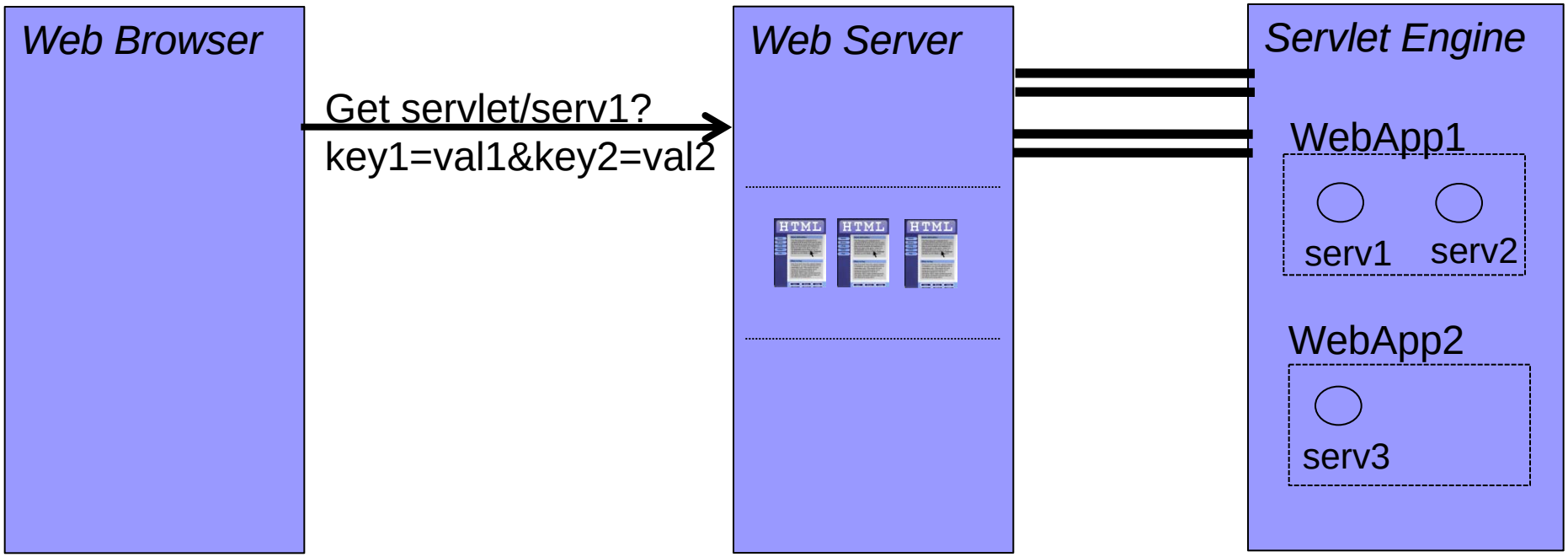
WebServer - WebApp Communication

2. Servlets



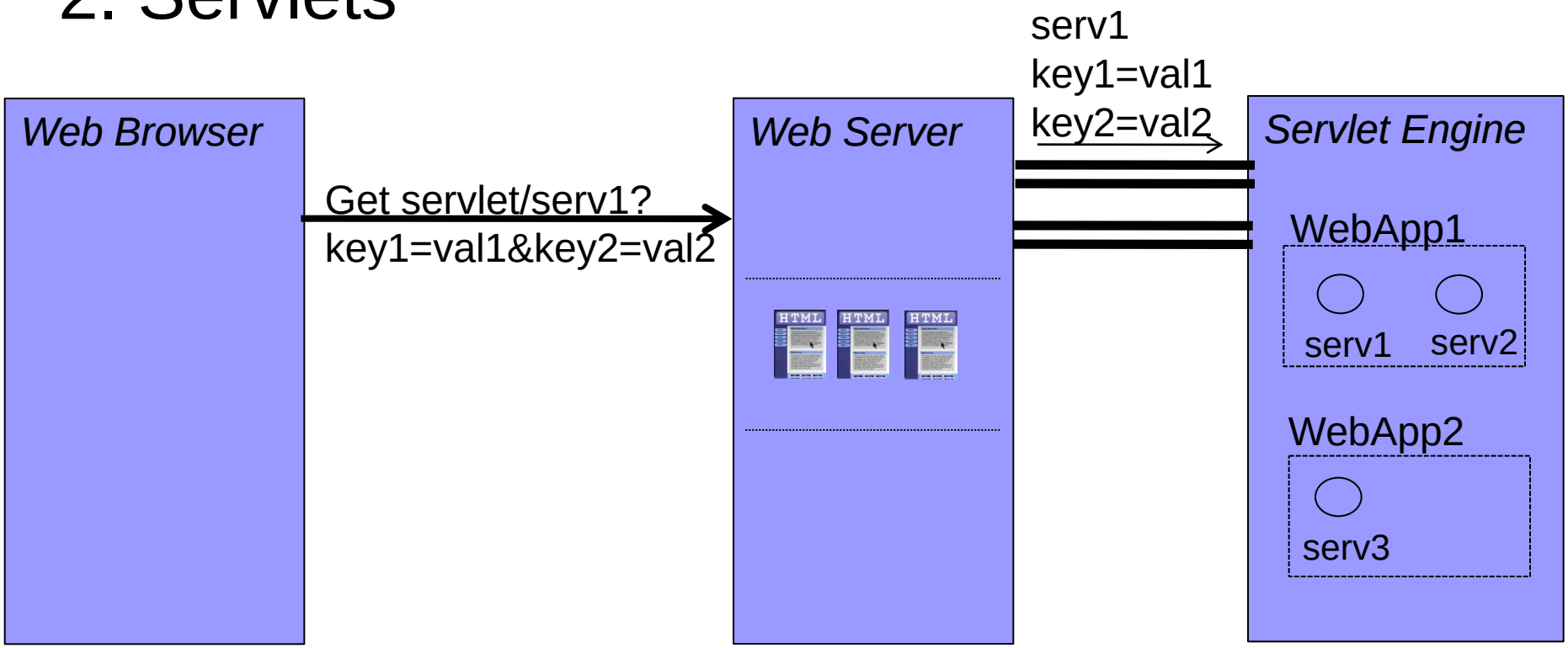
WebServer - WebApp Communication

2. Servlets



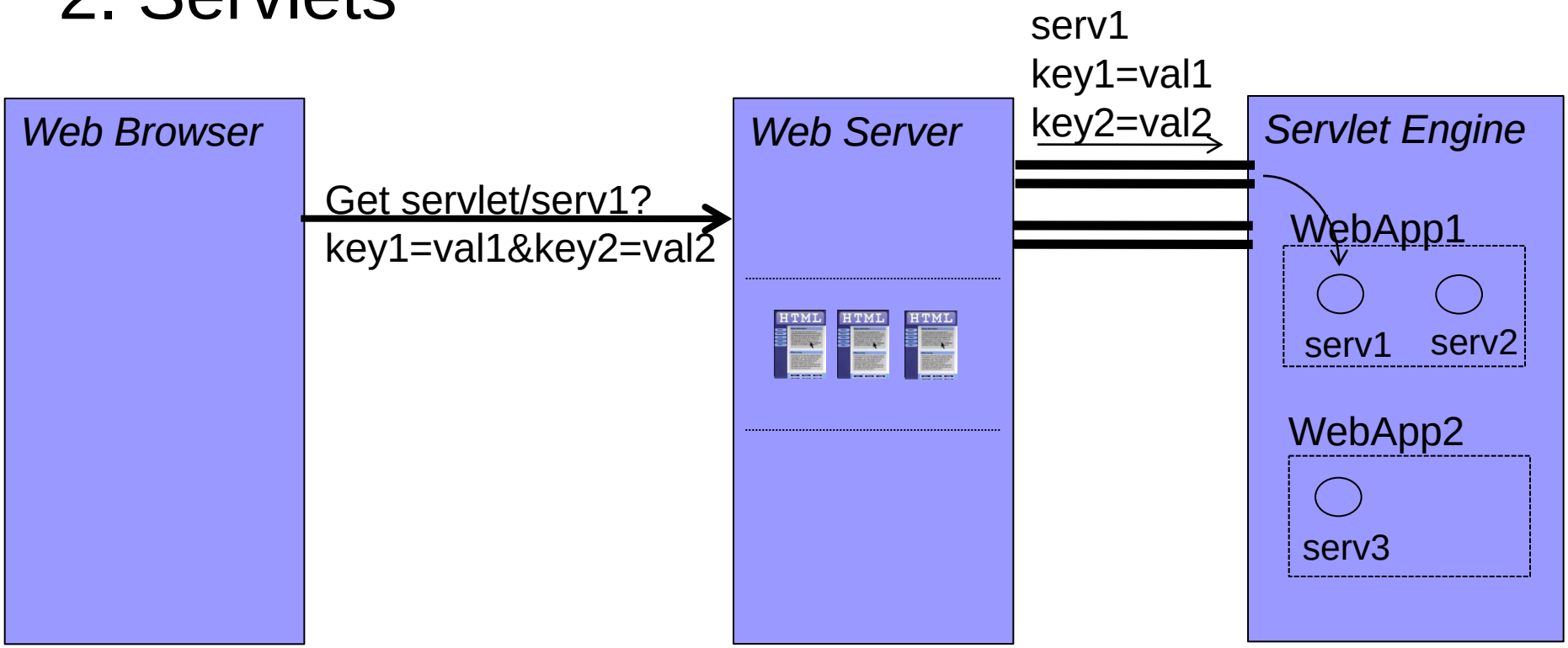
WebServer - WebApp Communication

2. Servlets



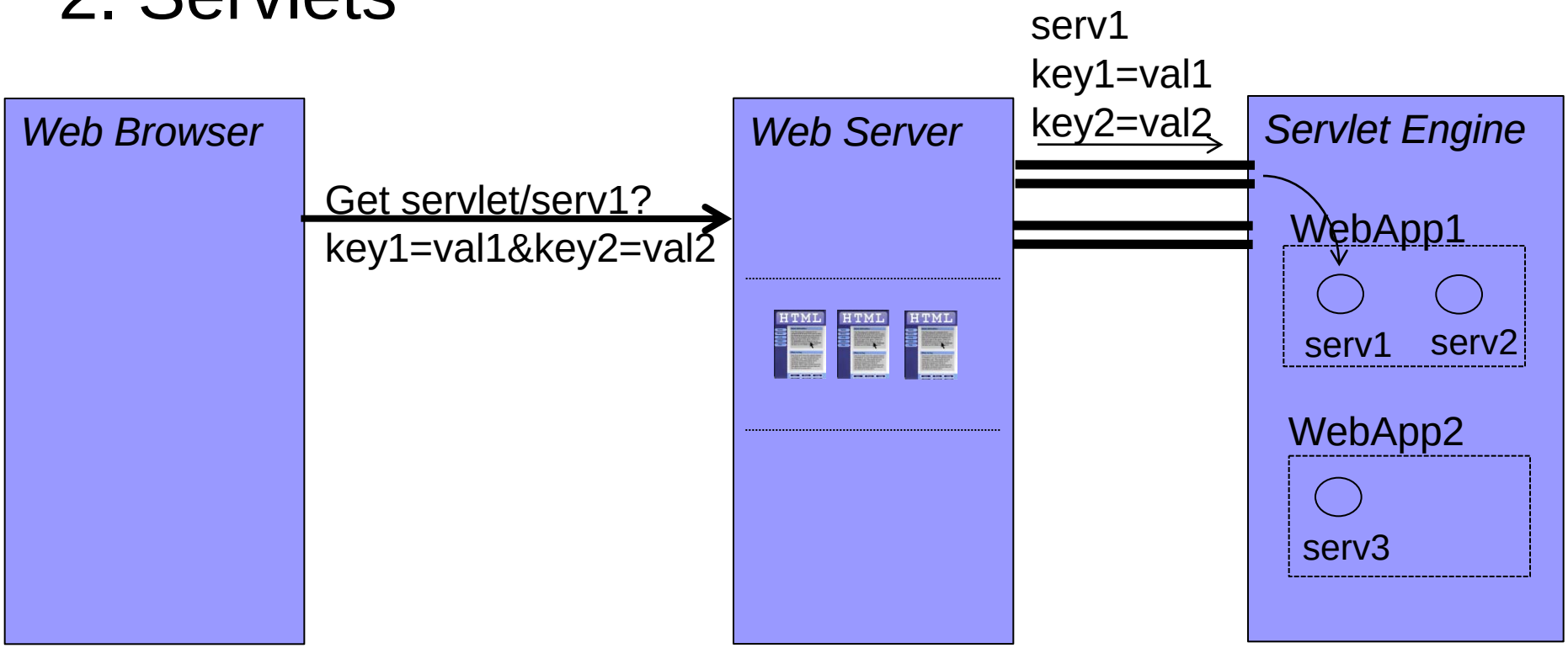
WebServer - WebApp Communication

2. Servlets



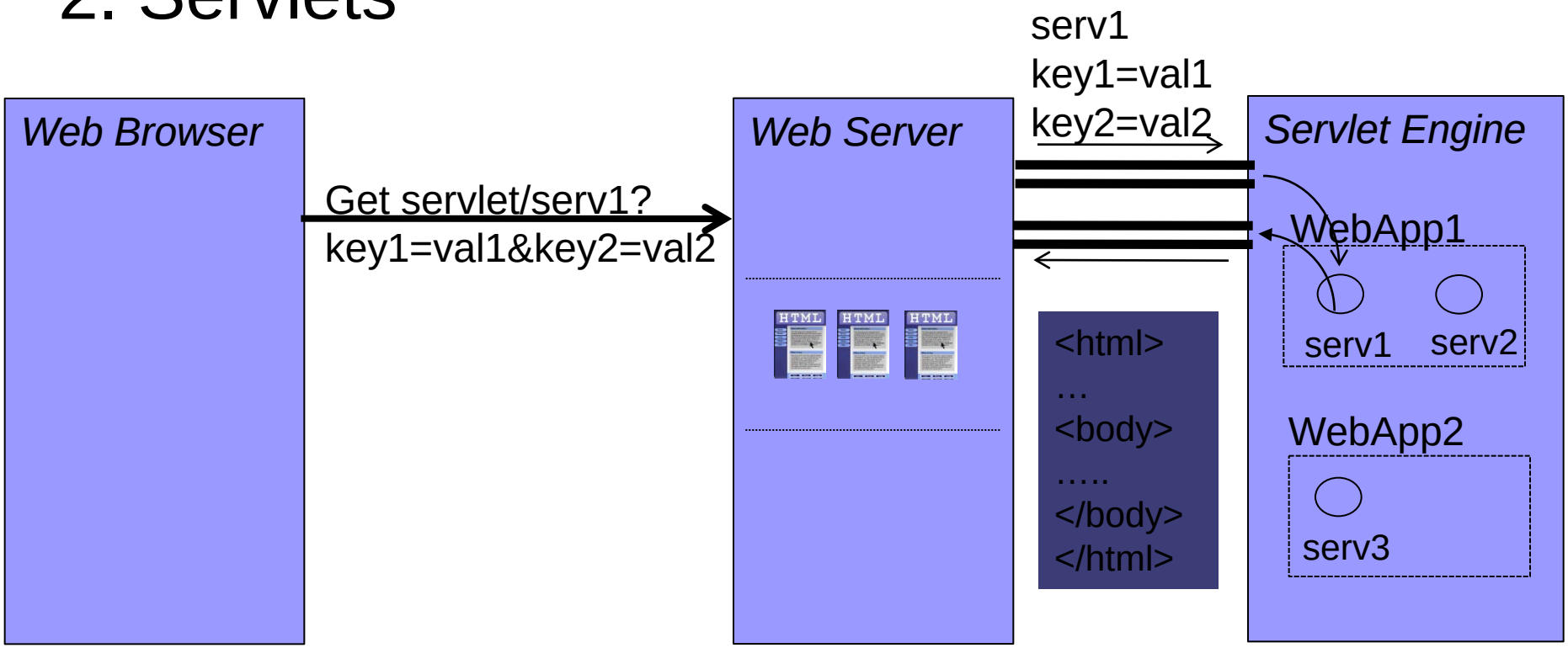
WebServer - WebApp Communication

2. Servlets



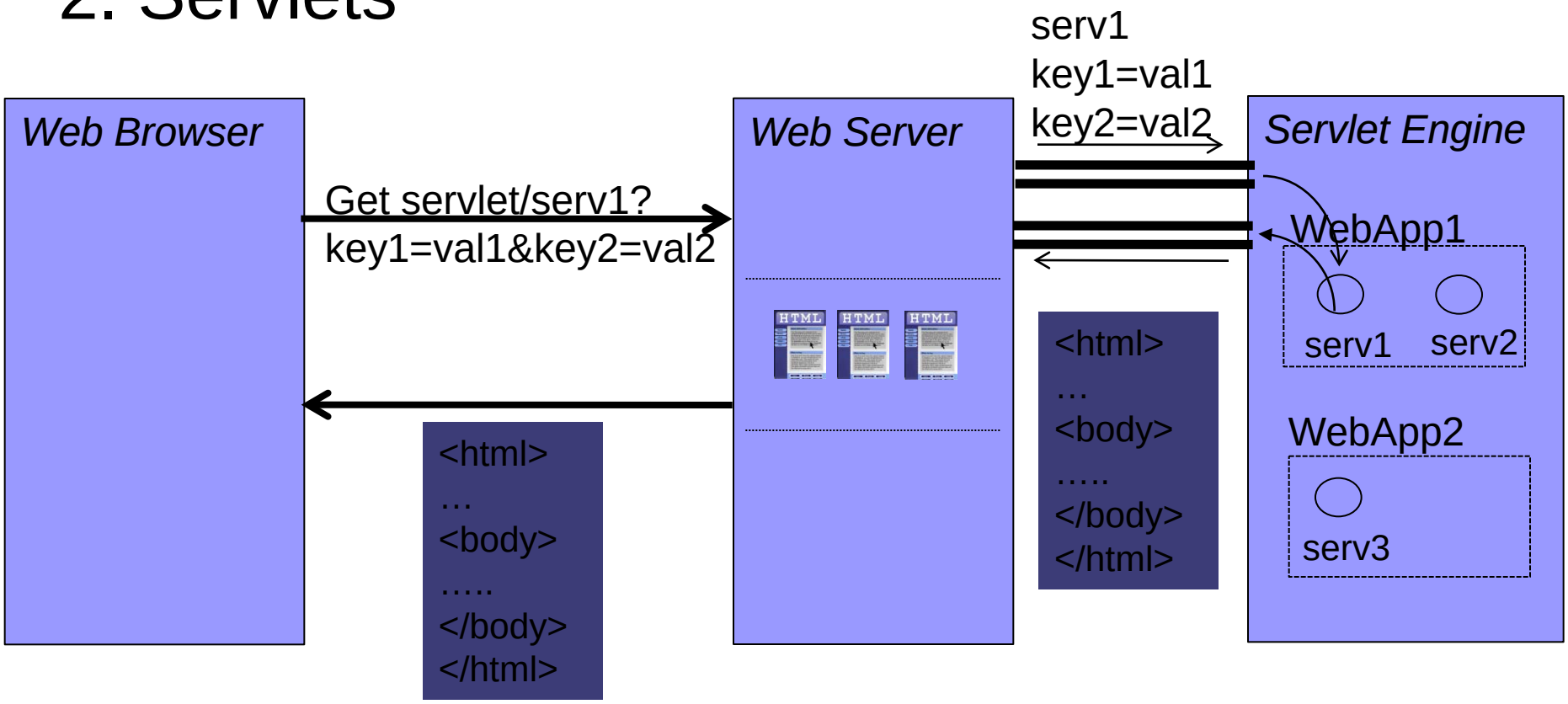
WebServer - WebApp Communication

2. Servlets



WebServer - WebApp Communication

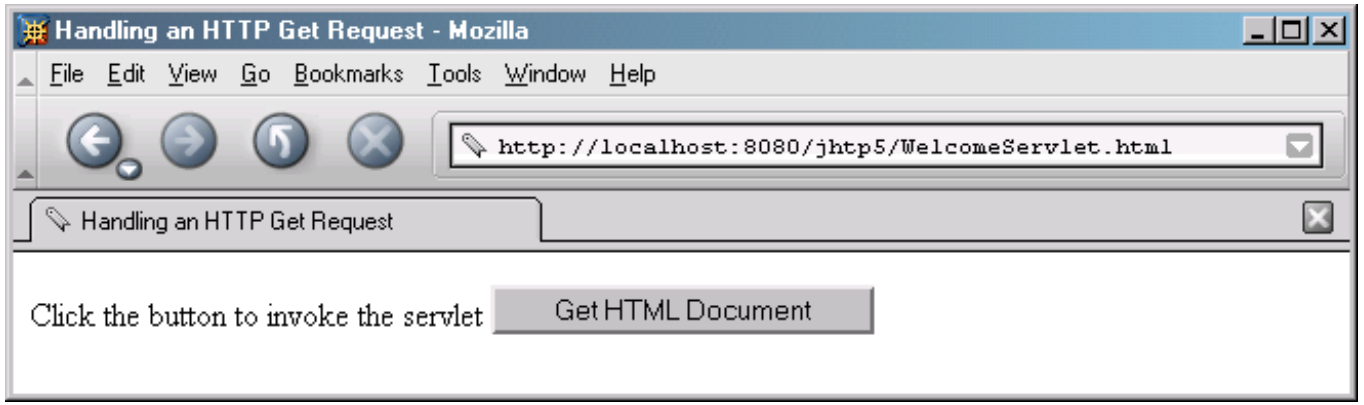
2. Servlets



```
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet extends HttpServlet {
9
10    // process "get" requests from clients
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       response.setContentType( "text/html" );
16       PrintWriter out = response.getWriter();
17
18       // send XHTML page to client
19
20       // start XHTML document
21       out.println( "<?xml version = \"1.0\"?>" );
22
23       out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
24          "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
25          "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
26
```

```
27          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
28
29          // head section of document
30          out.println( "<head>" );
31          out.println( "<title>A Simple Servlet Example</title>" );
32          out.println( "</head>" );
33
34          // body section of document
35          out.println( "<body>" );
36          out.println( "<h1>Welcome to Servlets!</h1>" );
37          out.println( "</body>" );
38
39          // end XHTML document
40          out.println( "</html>" );
41          out.close();  // close stream to complete the page
42       }
43    }
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 24.6: WelcomeServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP Get Request</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp5/welcome1" method = "get">
14
15       <p><label>Click the button to invoke the servlet
16          <input type = "submit" value = "Get HTML Document" />
17       </label></p>
18
19    </form>
20 </body>
21 </html>
```


Handling an HTTP Get Request - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet.html
Handling an HTTP Get Request
Click the button to invoke the servlet
Get HTML Document


A Simple Servlet Example - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome1?
A Simple Servlet Example
Welcome to Servlets!

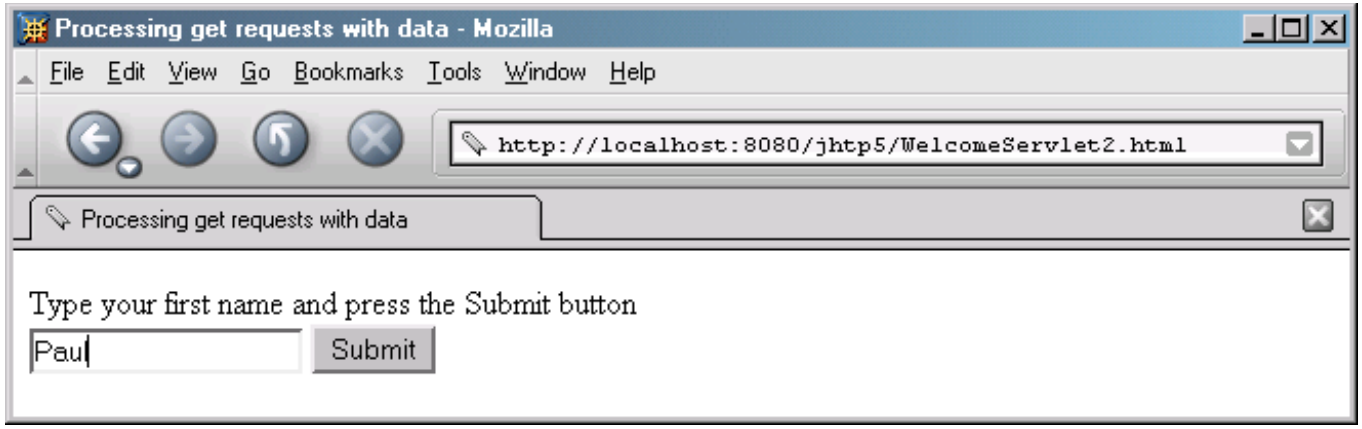
```
1   <!DOCTYPE web-app PUBLIC \
2      "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3      "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
4
5   <web-app>
6
7      <!-- General description of your Web application -->
8      <display-name>
9         Java How to Program JSP
10        and Servlet Chapter Examples
11     </display-name>
12
13     <description>
14        This is the Web application in which we
15        demonstrate our JSP and Servlet examples.
16     </description>
17
18     <!-- Servlet definitions -->
19     <servlet>
20        <servlet-name>welcome1</servlet-name>
21
22        <description>
23           A simple servlet that handles an HTTP get request.
24        </description>
25
```

```
26          <servlet-class>
27             WelcomeServlet
28          </servlet-class>
29       </servlet>
30
31       <!-- Servlet mappings -->
32       <servlet-mapping>
33          <servlet-name>welcome1</servlet-name>
34          <url-pattern>/welcome1</url-pattern>
35       </servlet-mapping>
36
37    </web-app>
```

```
2  // Processing HTTP get requests containing data.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet2 extends HttpServlet {
9
10    // process "get" request from client
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       String firstName = request.getParameter( "firstname" );
16
17       response.setContentType( "text/html" );
18       PrintWriter out = response.getWriter();
19
20       // send XHTML document to client
21
22       // start XHTML document
23       out.println( "<?xml version = \"1.0\"?>" );
24
```

```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing get requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 24.12: WelcomeServlet2.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Processing get requests with data</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp5/welcome2" method = "get">
14
15       <p><label>
16          Type your first name and press the Submit button
17          <br /><input type = "text" name = "firstname" />
18          <input type = "submit" value = "Submit" />
19       </p></label>
20
21    </form>
22 </body>
23 </html>
```


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet2.html
Processing get requests with data
Type your first name and press the Submit button
Paul
Submit


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome2?firstname=Paul
Processing get requests with data
Hello Paul,
Welcome to Servlets!

Servlet Lifecycle (Creation)

- Single instance created
- init() method called
- You can override init() in your subclass of HttpServlet to do some initial code....
- init() is NOT called again on further requests

Servlet Lifecycle (Service Method)

- On each request, the server spawns a new thread and calls service()
- service() checks HTTP request type and calls appropriate doXXXX (Get, Post, Put...)
- don't override service (unless you really know what you're doing)

Servlet Lifecycle (doGet(), doPost())

- Real meat of the web app is here
- doPost() can call doGet(), or viceversa
- no doHead()... system uses headers of doGet() result

Servlet Lifecycle (destroy())

- For some reason (servlet idle, etc) the server may want to remove the servlet from memory
- destroy() allows you to close DB connections, wrap up, etc...
- Don't count on destroy to write persistent state (server may crash before you ever get here!)

Accessing Request Components

- getParameter("param1")
- getCookies() => Cookie[]
- getContentLength()
- getContentType()
- getHeaderNames()
- getMethod()

Environment Variables

- JavaServlets do not require you to use the clunky environment variables used in CGI
- Individual functions:
  - PATH_INFO req.getPathInfo()
  - REMOTE_HOST req.getRemoteHost()
  - QUERY_STRING req.getQueryString()
  - …

Setting Response Components

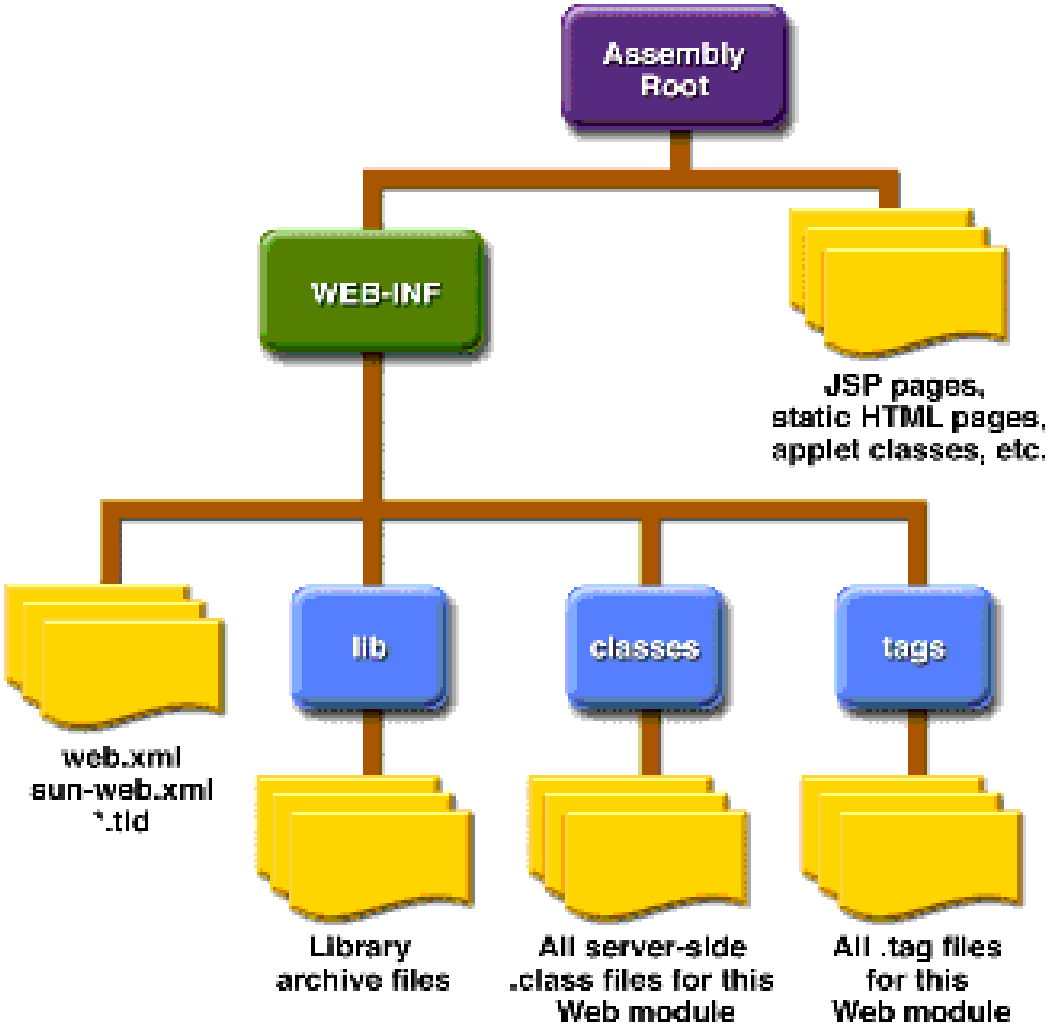
- Set status first!
  - setStatus(int)

    HttpServletResponse.SC_OK...
  - sendError(int, String)
  - sendRedirect(String url)

Setting Response Components

- Set headers
  - setHeader(…)
  - setContentType(“text/html”)
- Output body
  - PrintWriter out = response.getWriter();
  - out.println("<HTML><HEAD>...")

Servlets – WebApp structure

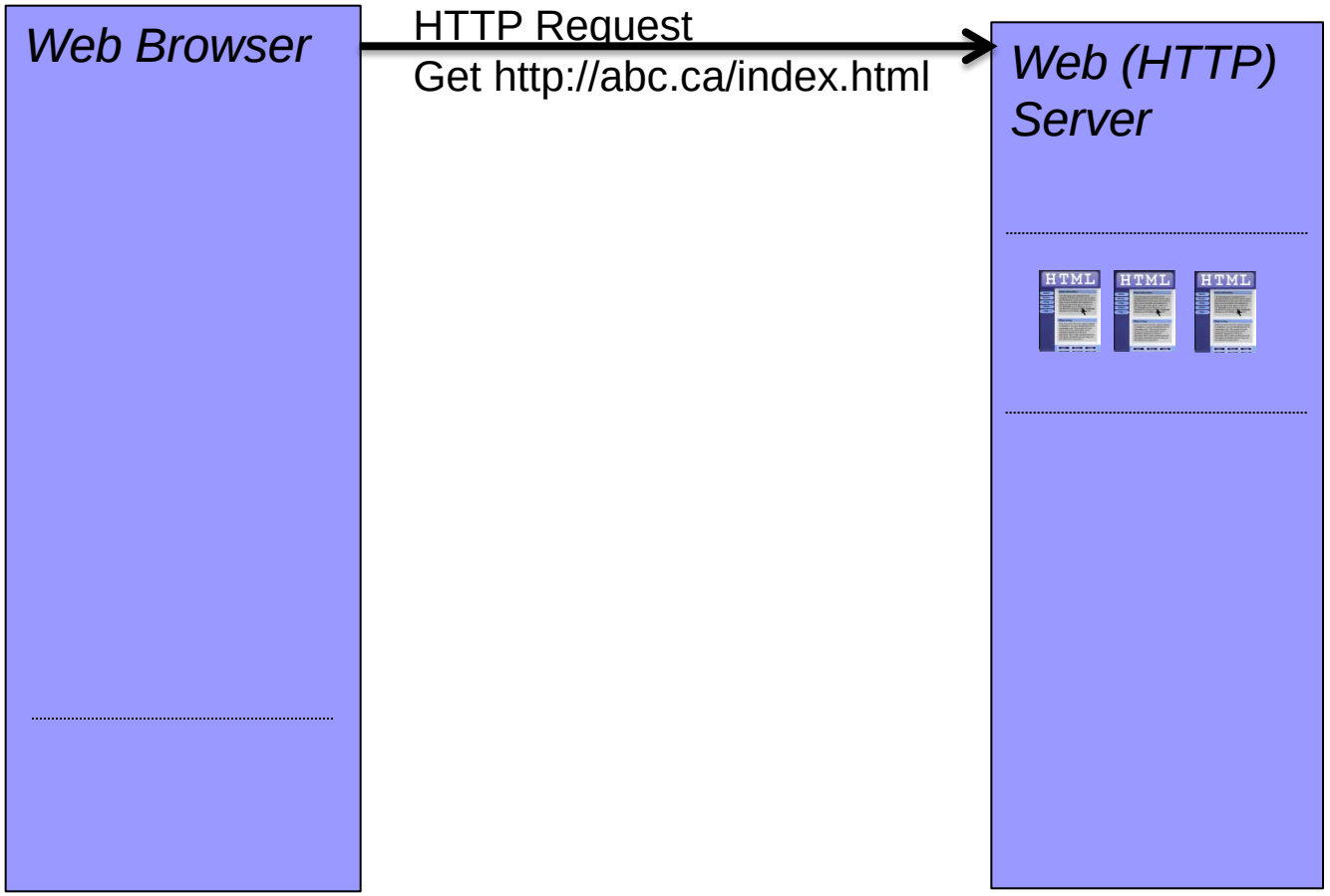


Servlets

- Pros
  - Save Process creation time
    - Lightweight threads instead of OS threads created
    - Single copy of code brought into memory for all threads versus per thread
    - Data (session state) can be stored across threads within servlet container
  - Object oriented model
- Cons
  - It’s actually called JavaServlets… (only Java)

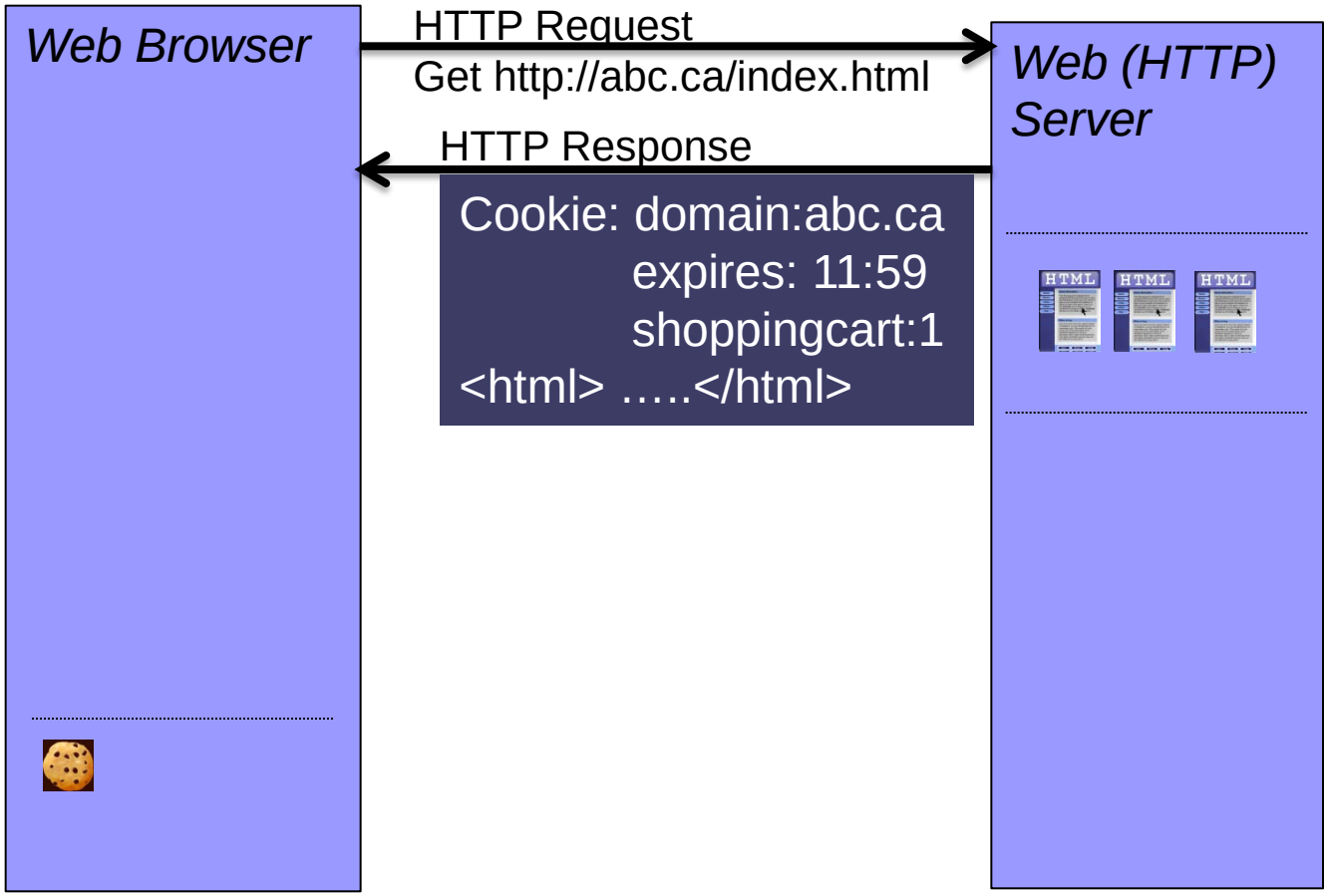
Session Tracking

1. Using Cookies



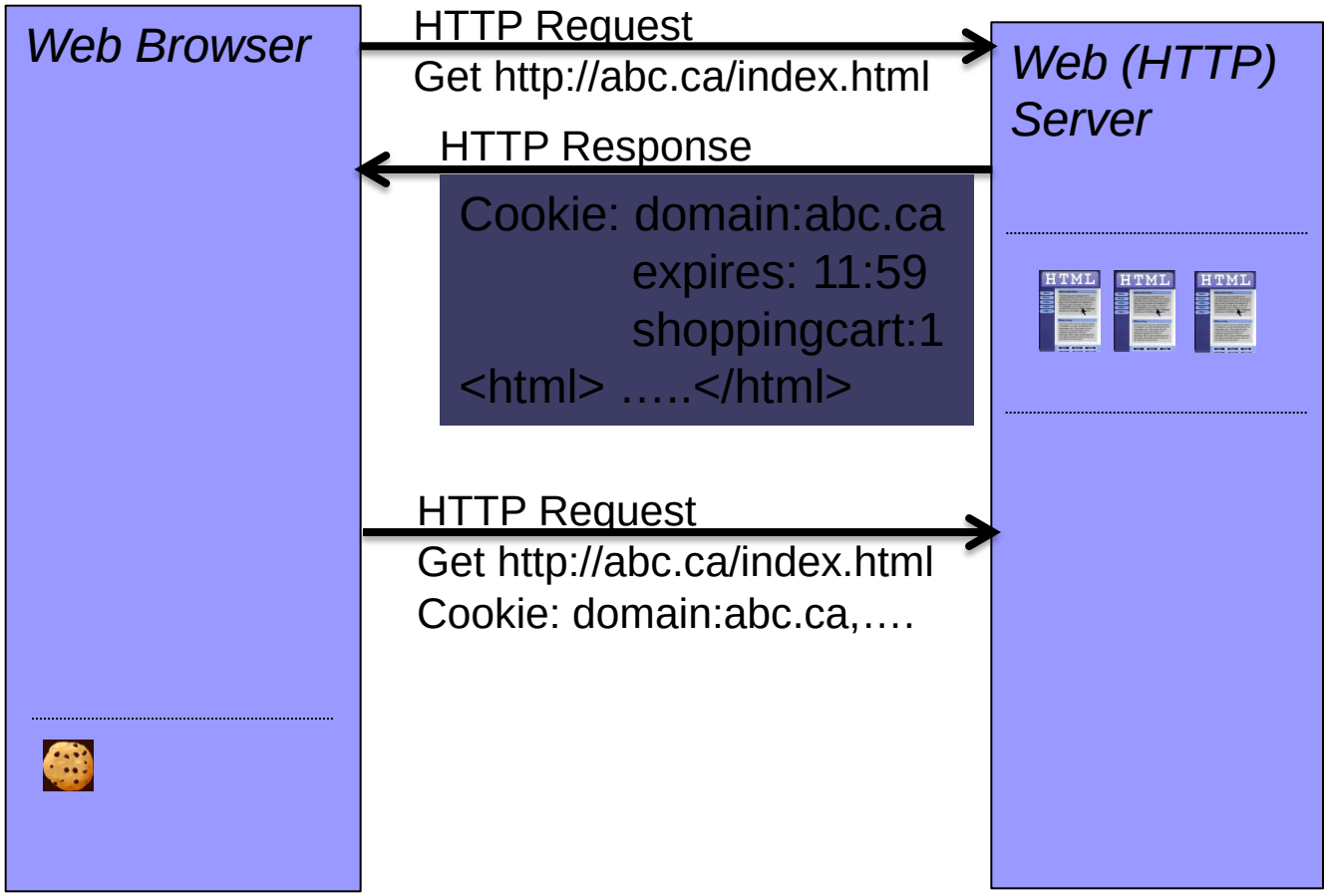
Session Tracking

1. Using Cookies



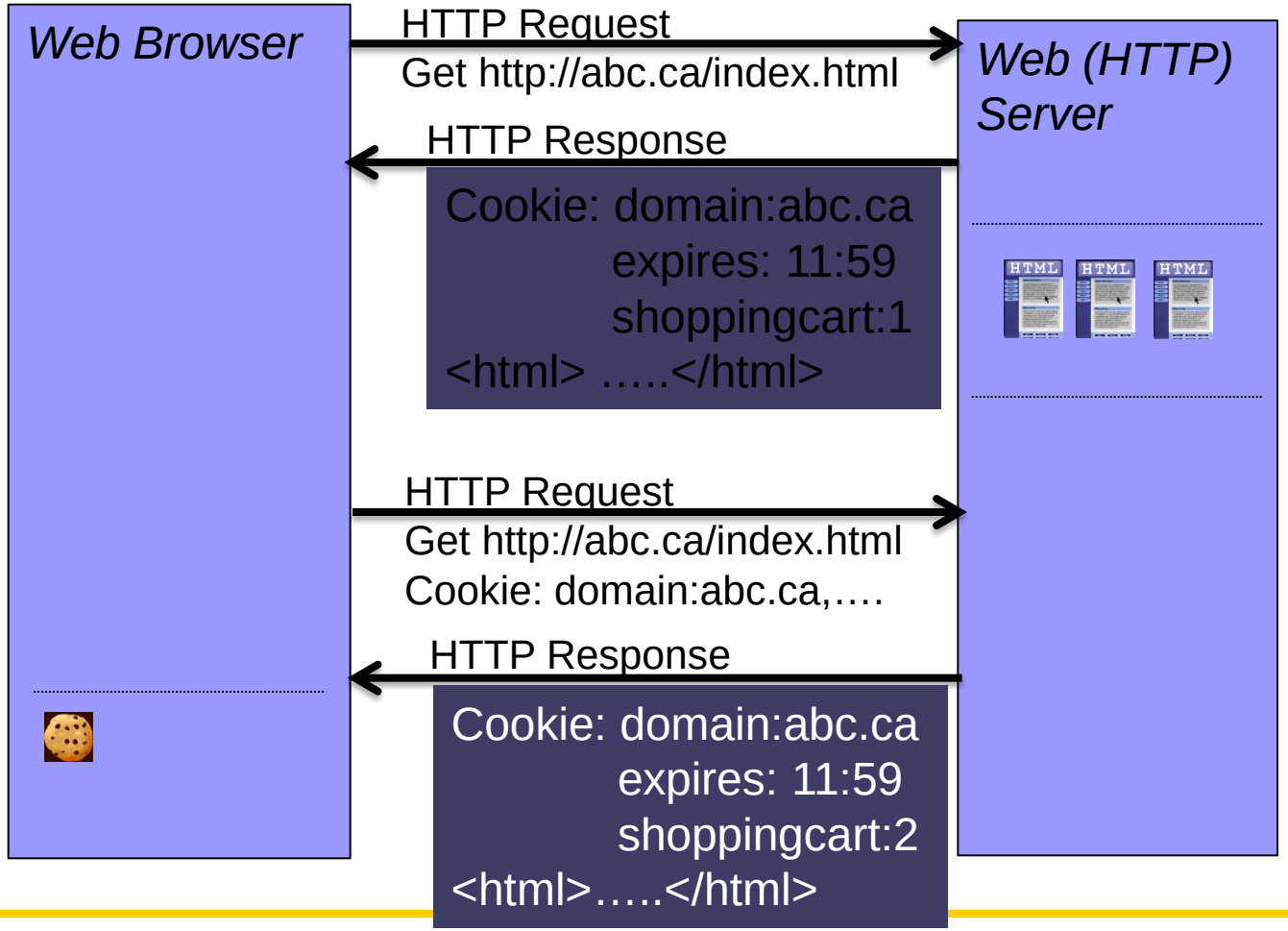
Session Tracking

1. Using Cookies



Session Tracking

1. Using Cookies



Session Tracking

1. Using Cookies

What's in a cookie?

- NAME=VALUE
  - Name value pairs
- expires=<data> *(optional)*
  - Without a date, deleted when browser closed
- path=<path> *(optional)*
- domain=<domain> *(optional)*
- secure *(optional)*

Session Tracking

1. Using Cookies

Cookie issues:

- User can control how they're handled:
  - Don't let server write them to client PC
  - Asks permission (perhaps with a white-list)
  - Always accept them
    - Perhaps except for those on a black-list
- User can view and manage cookies
  - E.g. Firefox's Options->Privacy->Show Cookies

Session Tracking

1. Using Cookies

How to generate a cookie ?

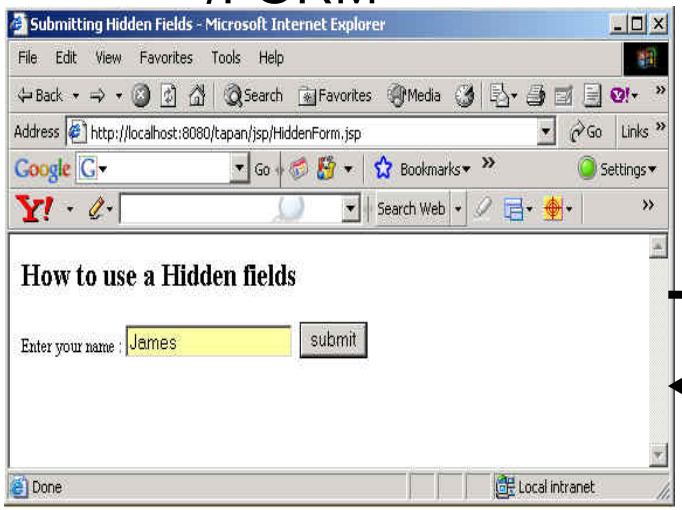
- From server-side:
  - Scripting languages simplify this
    - E.g. PHP setcookie('name', 'value',…);
- From client-side:
  - JavaScript: *document.cookie="user=tom;domain=cs.utoronto.ca;path=/cs309; secure;"*

Session Tracking

2. Using Hidden Form Fields

- Hidden Form Fields can be used to store/pass values

```
<FORM ACTION="login" METHOD="post">
   Enter your name :<input type ="text" name ="name" value = "">
                    <input type="hidden" name="session" value="12">
                    <input type="submit" value="submit">
</FORM>
```



POST /login
HTTP/1.1
Host: localhost
……
name=James&session=12

<html>……</html>

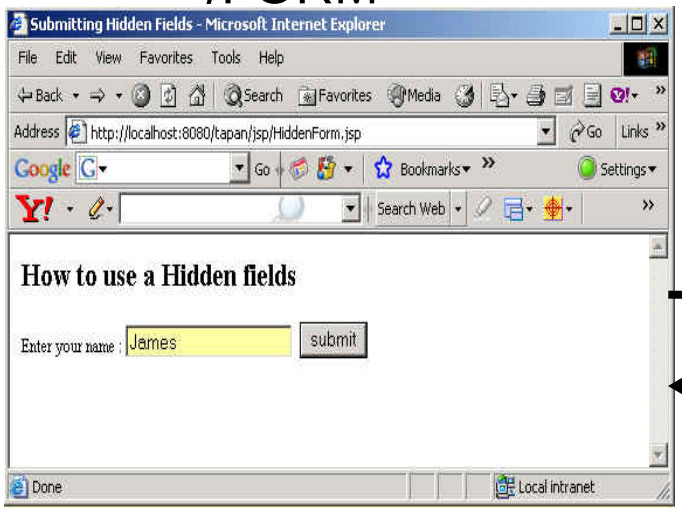
Web Server

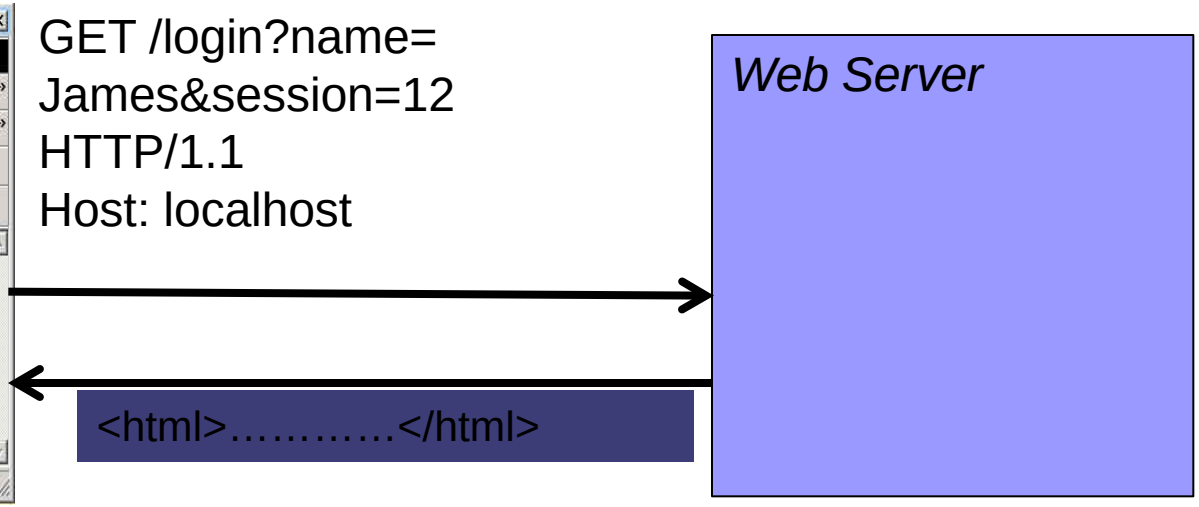
Session Tracking

2. Using Hidden Form Fields

- Hidden Form Fields can be used to store/pass values

```
<FORM ACTION="login" METHOD="get">
   Enter your name :<input type ="text" name ="name" value = "">
                    <input type="hidden" name="session" value="12">
                    <input type="submit" value="submit">
</FORM>
```





Session Tracking

2. Using Hidden Form Fields

- Pros:
  - Easy to implement and supported by most browsers
  - Works even if cookies are disabled or unsupported
- Cons:
  - Hidden fields must be created in a particular sequence
  - Can’t use the back button without loosing information
  - Lots of tedious processing
  - All pages must be the result of form submissions

Session Tracking

3. Using URL Rewriting

- Client appends some extra data on the end of each URL that identifies the session
- Server associates that identifier with data it has stored about that session

Original URL: http://server:port/file
Rewritten URL: http://server:port/file?sessionid=7456

Session Tracking

3. Using URL Rewriting

- Pros:
  - Works even if cookies are disabled or unsupported
- Cons:
  - Lots of tedious processing
  - Must encode all URLs that refer to your own site
  - Links from other sites and bookmarks can fail