Session & cookies

UNIVERSITY OF TORONTO

HTTP is Stateless

→ it simply allows a browser to request a single document from a web server

→ it remembers nothing between invocations

→ Short lived

→ **EVERY** resource that is accessed via HTTP is a single request with no threaded connection between them.

HTTP is Stateless

example.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="theme.css">
</head>
<body>
<img src="w3schools.jpg" alt="W3Schools.com" />
<img src="smiley.gif" alt="Smiley face">
</body>
</html>
```

HTTP is Stateless

example.html

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="theme.css">
</head>
<body>
<img src="w3schools.jpg" alt="W3Schools.com" />
<img src="smiley.gif" alt="Smiley face">
</body>
</html>
```

There will be **four** TCP connections negotiated and opened, **four** data transfers, **four** connections closed

HTTP is Stateless

Mostly true for HTTP 1.0

HTTP 1.1 adds persistent connection mechanisms to boost performance.

HTTP is Stateless

HTTP is stateless, it makes a lot of sense when sharing static information like html, pdf, images over HTTP (1.0).

But as we started using web application, ecommerce sites, we started adding **ad hoc** states on top of HTTP for various reasons.

Motivation - adding state to HTTP

amazon.com and site like them knows if you have visited them before. Some for example are,

- Remember me?
- Already logged in
- My preferences
- What's new?
- MY DATA!

How - adding state to HTTP

How do they do this?

How does a client uniquely identify itself to a server?

How does the server provide specific content to each client?

Adding state to HTTP (1.0 and earlier)

There are various ways to add and maintain states on top of HTTP (not as an integral part):

Client mechanisms:

- Cookies
- Hidden variables
- URL rewriting
- Local storage (for HTTP 1.1 and we have covered this)

Server mechanisms:

- sessions

Cookies

A small amount of information sent by a server to a browser, and then sent back by the browser on future page requests.

Motivation:

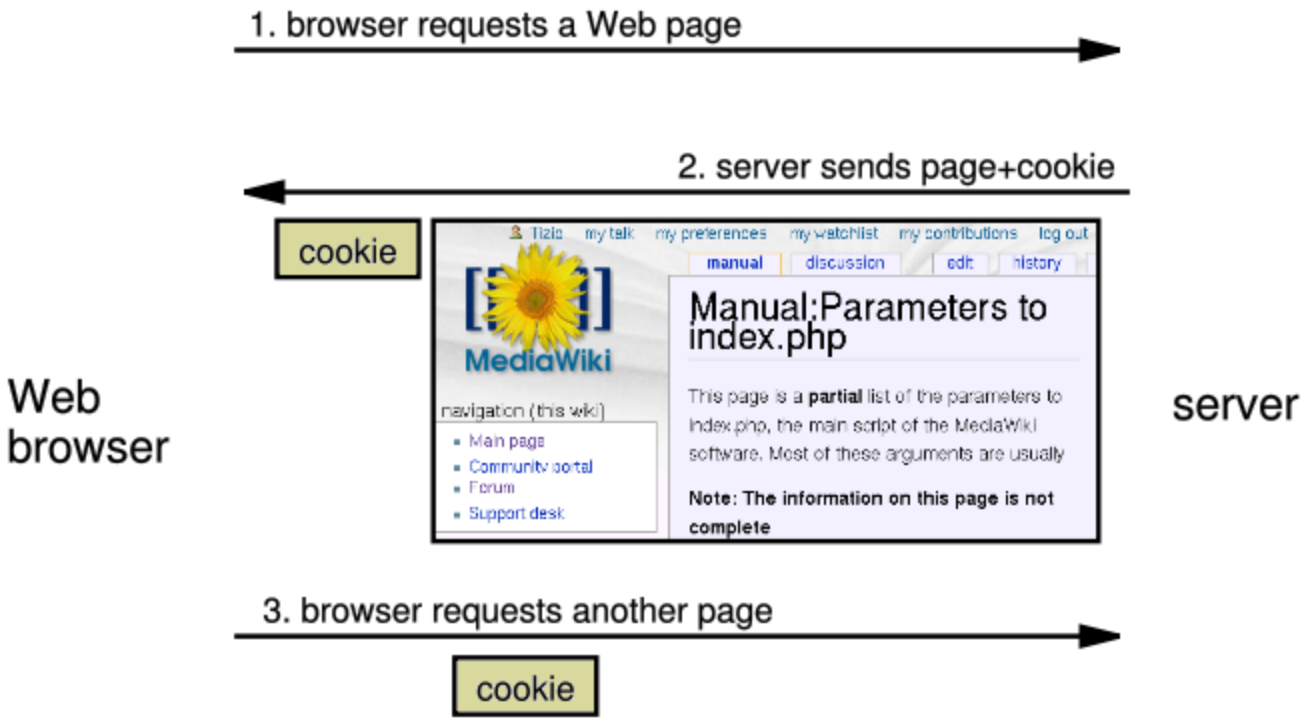
→ authentication

→ user tracking

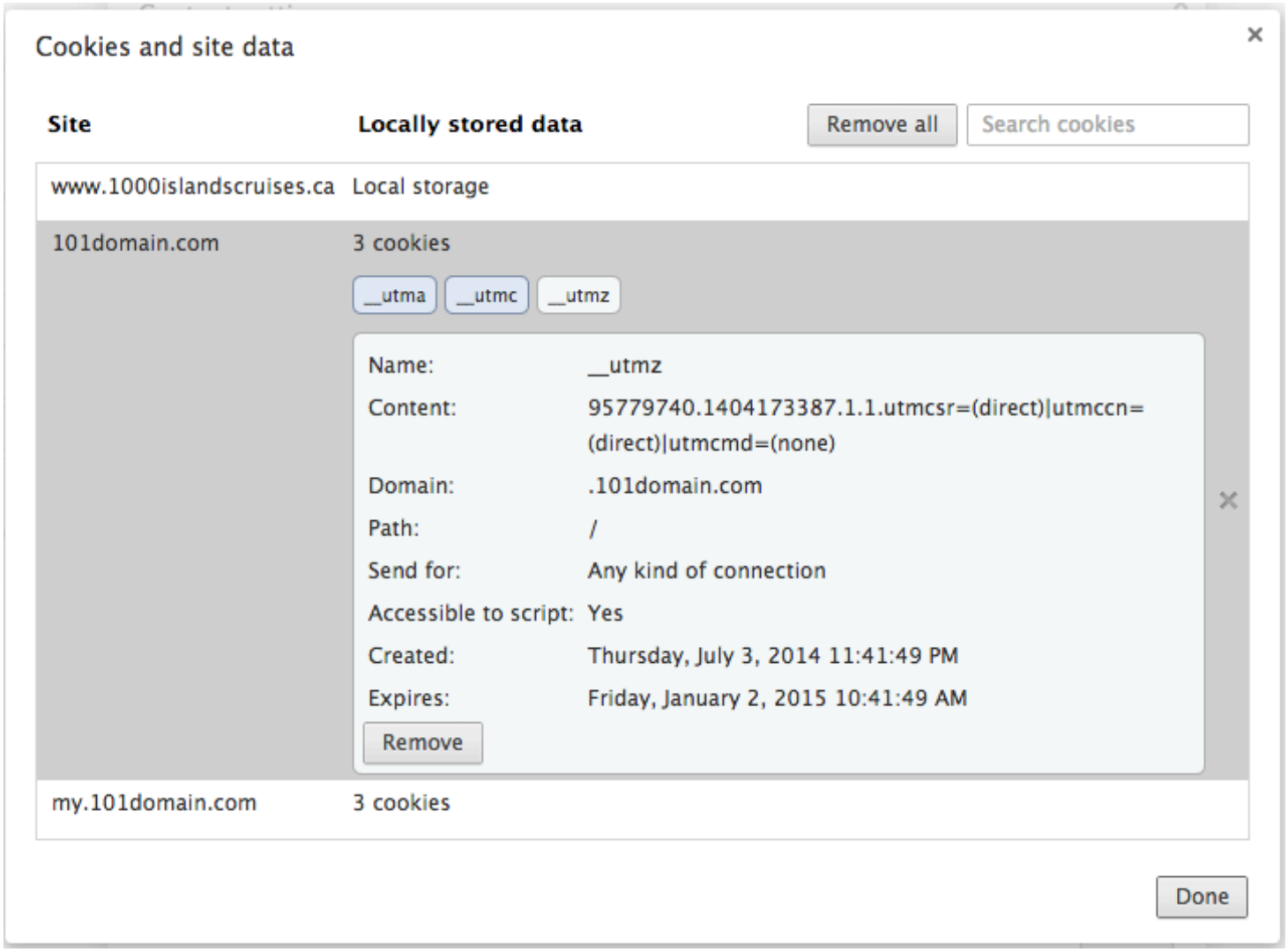
→ maintaining user preferences, shopping carts, etc.

Cookies

- Set by the server
- Sent with the HTTP Response header
- Returned by browser with HTTP Request



Cookies - in chrome



UNIVERSITY OF TORONTO

Setting Cookie - using node.js

```
var http = require('http');

http.createServer(function (request, response) {

  response.writeHead(200, {
    'Set-Cookie': 'mycookie=test',
    'Content-Type': 'text/plain'
  });
  response.end('Hello World\n');
}).listen(1234);

console.log('Server running at http://127.0.0.1:1234/');
```

Note: A cookie is a header field, and needs to be set before printing the response body

Setting Cookie - using javascript

- JS has a global document.cookie field (a string)
- you can manually set/get cookie data from this field (sep. by ;), and it will be saved in the browser

```
document.cookie = "username=smith;password=12345";
```

or

```
Cookies.set("username", "smith");
...
alert(Cookies.get("username"));   // smith
```

How long does a cookie exist?

session cookie : the default type; a temporary cookie that is stored only in the browser's memory

- when the browser is closed, temporary cookies will be erased
- can not be used for tracking long-term information
- safer, because no programs other than the browser can access them

How long does a cookie exist?

persistent cookie : one that is stored in a file on the browser's computer

- can track long-term information
- potentially less secure, because users (or programs they run) can open cookie files, see/change the cookie values, etc.

Limitation of Cookies

- Browser (users) can refuse cookies
- Size limit / expiration policy

http://browsercookielimits.x64.me/

```
document.cookie="username=John Doe;
expires=Thu, 17 Jul 2014 5:00:00 GMT";
```

Hidden Variables

Cookies stores state in browsers, if you want to Store state in web pages, Add hidden variables.

→ Supported by all browsers

→ State is sent inside each web page.

```
<input type="hidden" name="secret"
value="Don't tell anyone!!">
```

For form based applications only. Following hyperlinks causes a loss of state.

Hidden Variables

→ Following hyperlinks causes a loss of state.

→ Current submitted page represents current state independent of what was done previously.

URL Rewriting

Store state in the URL: Rewrite URLs so that they include state variables

- Each URL is now a get request
- Supported by all browsers
- Requires all URLs contain all state information (long URLs)
- Current submitted page represents current state independent of what was done previously.

Session - Persistent State

Current state is stored at the server (i.e., in a file, database)

- Each request includes a token identifying the browsers session (tokens can be passed via cookies, hidden variables, URL rewriting).
- At each request, the executing script uses the token to fetch session state

Session hijacking! Add unique value + signature

Session - Persistent State

```
var session = sessions.lookupOrCreate(request,{
  lifetime:604800
 });

 response.writeHead(200, {
  'Content-Type': 'text/plain',
  'Set-Cookie', session.getSetCookieHeaderValue()
 });
```

http://blog.nodejitsu.com/sessions-and-cookies-in-node/

References

1. http://www.webstepbook.com/supplements/slides/lecture25-cookies.shtml

2. http://blog.nodejitsu.com/sessions-and-cookies-in-node/