VSR://edu/2020/evs/

02 – URI, HTTP, Cookies

//// Design of Distributed Systems ////////

Mahda Noura, MSc.

VSR.Informatik.TU-Chemnitz.*de*

1

URI, URN, URL

What is the difference between URI and URL?

URI is an abstract resource identifier (may be a unique name of the resource – URN or it's location – URL)

URL describes a location of the resource and the protocol used to access it

What is the meaning of the URL scheme?

URL scheme describes a method to access a resource

Often (but not always) corresponds to some specific protocol:

- http
- ftp
- news
- ssh
- file
- ldap
- ...

What, why and how should be encoded in URLs?

What?
- Segments of URLs

Why?
- Reserved characters : / ? # [ ] @ ! $ & % ' ( ) * + , ; =
- Non-ASCII characters
- Unsafe characters (whitespace, “,<,>...)

How?
- % + 2 Hexadecimal digits
- Hexadecimal digits correspond to the ASCII-value of the character
- Each byte of the UTF-8 encoding for non-ASCII symbols

Exercise 1: URL

Which URLs are syntactically correct and which are not:


1. *http://www.tu-chemnitz.de/informatik*


2. *http://tu-chemnitz.de/informatik*


3. *http://www.tu-chemnitz.de:443/informatik*


4. *http://www.tu-chemnitz.de/informatik?show=all?group=true*


5. *http://www.tu-chemnitz.de/informatik?show=all%20group=true*


6. *file:///c:/windows/php.ini*


7. *ftp://www.tu-chemnitz.de/informatik?show=all&group=true*


8. *ftp://bob:pass@www.tu-chemnitz.de/informatik*


9. *mailto://user@example.org*

2 HTTP

HTTP Methods

- GET
- POST
- PUT
- HEAD
- DELETE
- OPTIONS

HTTP Requests

```
GET          /scripts/guestbook.php HTTP/1.1
Host:        vsr.informatik.tu-chemnitz.de
Connection:  keep-alive
User-Agent:  Mozilla/5.0
Accept:      text/html
```

HTTP Requests

```
POST            /scripts/guestbook.php HTTP/1.1
Host:           vsr.informatik.tu-chemnitz.de
Connection:     keep-alive
User-Agent:     Mozilla/5.0
Accept:         text/html
Content-Length: 50
Content-type: application/x-www-form-urlencoded

name=Stefan&text=This%20is%20a%20Guestbook%20Entry
```

HTTP Response

```
HTTP/1.1 200 OK
Date:           Thu, 01 Dec 2016 12:30:24 GMT
Server:         Apache/2.2.31
Content-Length: 1322
Connection:     Keep-Alive
Keep-Alive:     timeout=3, max=100
Content-Type:   text/html
```

HTTP Status Codes

- 20X Success
  - *200 OK*
- 30x Redirection
  - *301 Moved Permanently*
  - *302 Found (Moved Temporarily)*
  - *303 See other*
- 40x Error
  - *400 Bad Request*
  - *401 Unauthorized*
  - *403 Forbidden*
  - *404 Not Found*
- 50x Server error
  - *500 Internal Server Error*
- *10x Information*
  - *101 Switching protocols*

GET vs POST

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

2

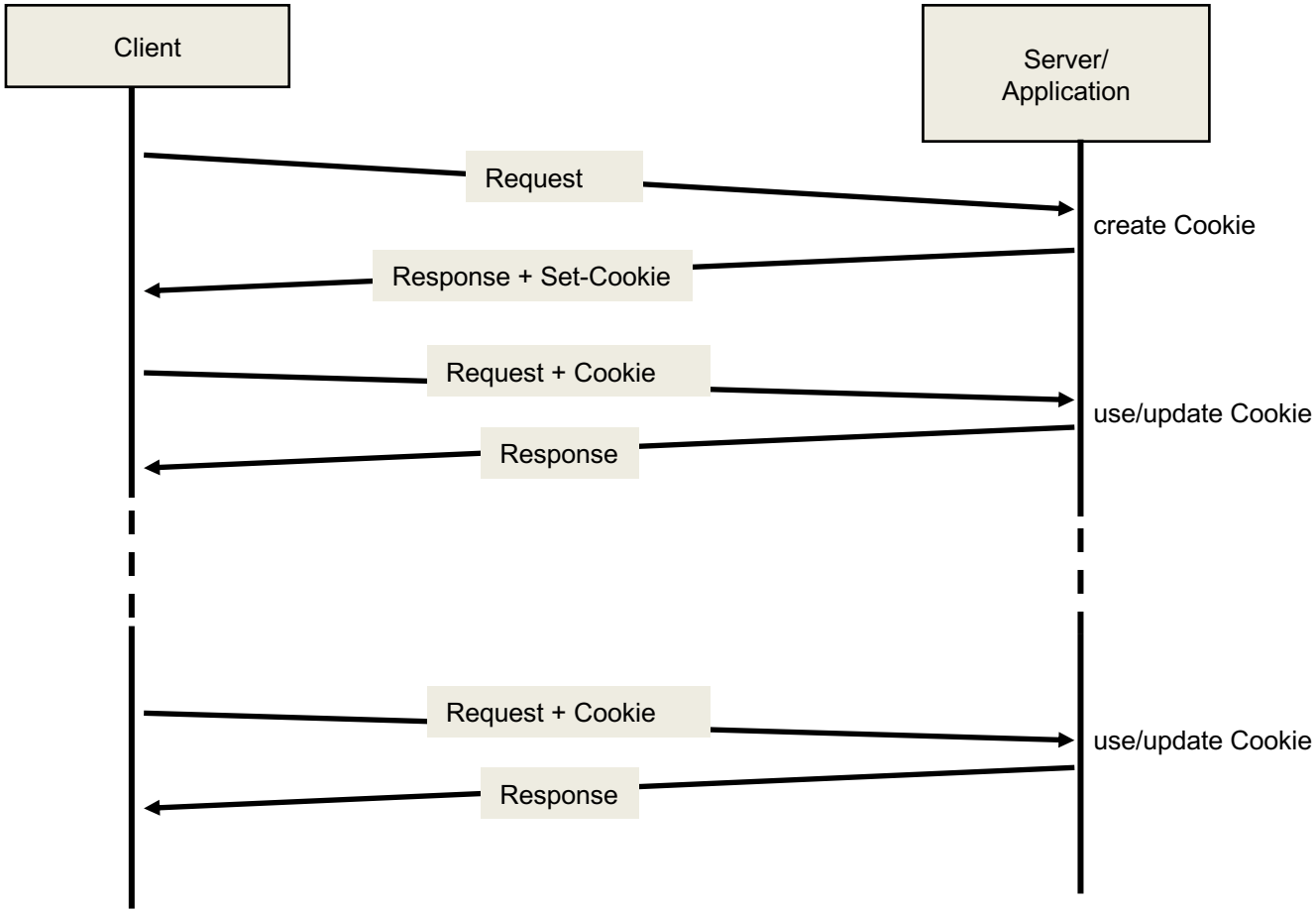
Cookies

A HTTP Cookie

- is a small text information stored by the webbrowser on the computer of the user
- it extends the stateless HTTP protocol by means to remember stateful information

In detail:

- A mechanism to store a small amount of data (up to 4KB) at the client [RFC6265]
- A cookie is associated with a specific web site
- Cookie is sent in HTTP header
- Cookie is sent with each HTTP request
- Can last for only one session (until browser is closed) or can persist across sessions
- Can expire some time in the future


Client
Server/
Application
Request
create Cookie
Response + Set-Cookie
Request + Cookie
use/update Cookie
Response
Request + Cookie
use/update Cookie
Response

Creation of a Cookie

```
HTTP/1.1 200 OK
Date:              Thu, 11 Nov 2019 12:30:24 GMT
Server:            Apache/2.2.31
Content-Length: 1322
Connection:        Keep-Alive
Keep-Alive:        timeout=3, max=100
Content-Type:      text/html
Set-Cookie:lastUser=Mahda;
expires=Tue, 10 Nov 2020 19:30:00 GMT;
Max-Age=2592000;
Path=/scripts/guestbook.php
```

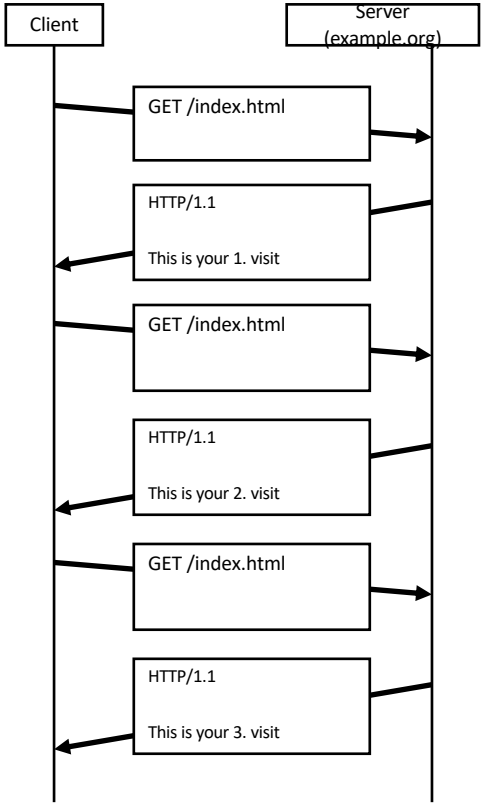
Cookie Data in a HTTP Request

```
GET             /scripts/guestbook.php HTTP/1.1
Host:           vsr.informatik.tu-chemnitz.de
Connection:     keep-alive
User-Agent:     Mozilla/5.0
Accept:         text/html
Cookie:         lastUser=Mahda
```

Application areas

- Session management (usually supported by *Session* objects in programming languages)
- Personalization
- Tracking

Exercise 3: Cookies



VSR

mytuc.org/tgxs

Thank you!

mahda.noura@informatik.tu-chemnitz.de

VSR.*Informatik*.TU-Chemnitz.*de*