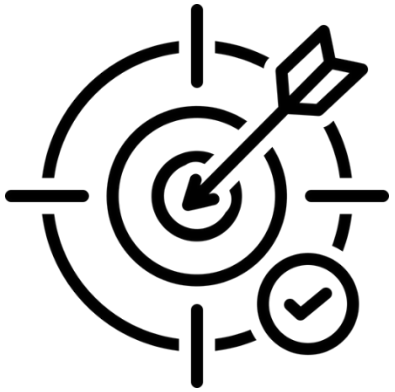
# State Management

## Web Programming and Testing

Mario Simaremare, S.Kom., M.Sc.
Program Studi Sarjana Sistem Informasi
Institut Teknologi Del

# Objectives

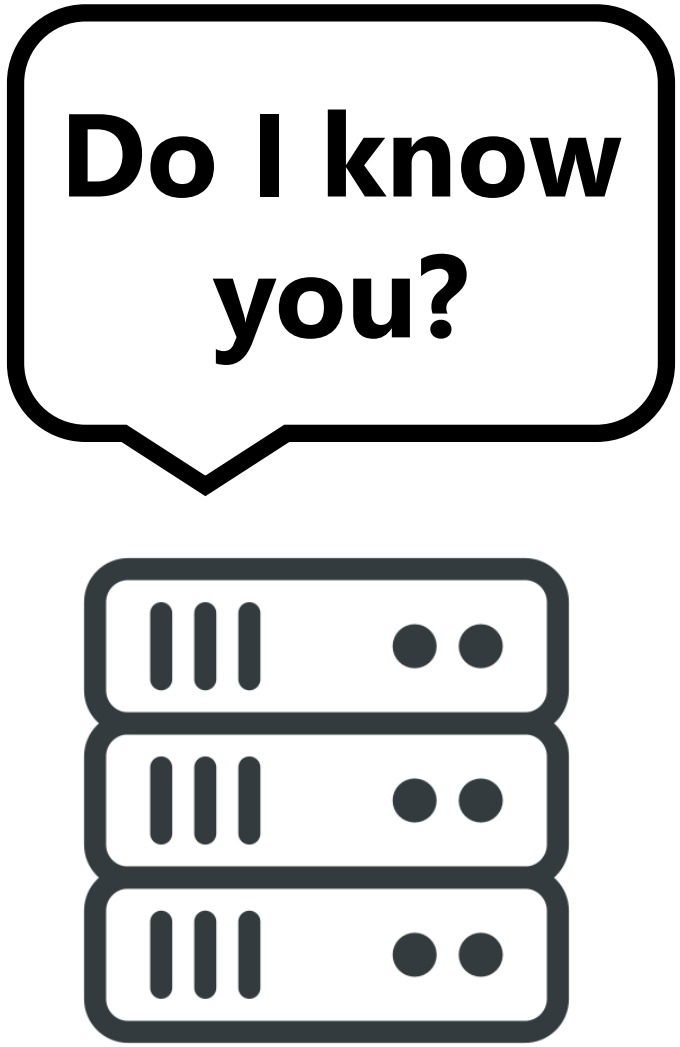
- The objective of this session is the following:
  - The students are able to elaborate the role of state and the importance of state management.
  - The students are able to manage states in either server or client side.



# Outlines

1. Motivation
2. Storing state in the client-side (cookies)
3. Storing state in the server-side (session)

# Motivation


Do I know you?

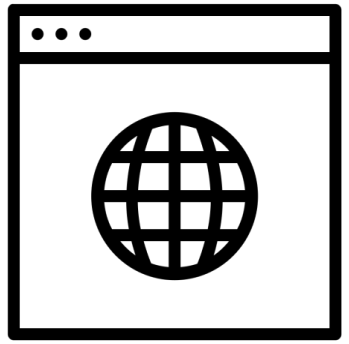
# The Protocols

- HTTP/S protocol is working in request-response cycle.
  - Every cycle is independent.

- The protocols are stateless.
  - Do not store any state whatsoever between the two communicating parties.

# What is State?

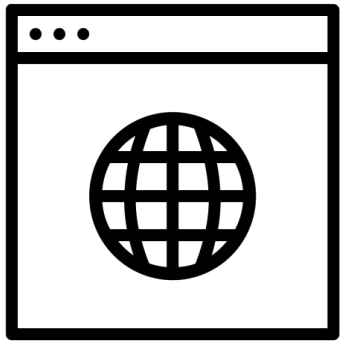
- State is a value associated to a particular user.
  - One or more states are generated due to the interaction between the two parties.



- Achieving the best user experience.
  - To track what have been done during the interactivity.
  - To temporarily record some generated data during the interactivity.
  - Personalization.

# What is State?

- HTTP/S is robust enough to enable state to be created and transmitted between the parties.



- User state management (RFC 6265):
  - The web server – called session,
  - The client (user agent, i.e. browser) – called cookies,
  - Or combination of both at the same time.

Please visit: https://tools.ietf.org/html/rfc6265

# Storing State in The Client-Side

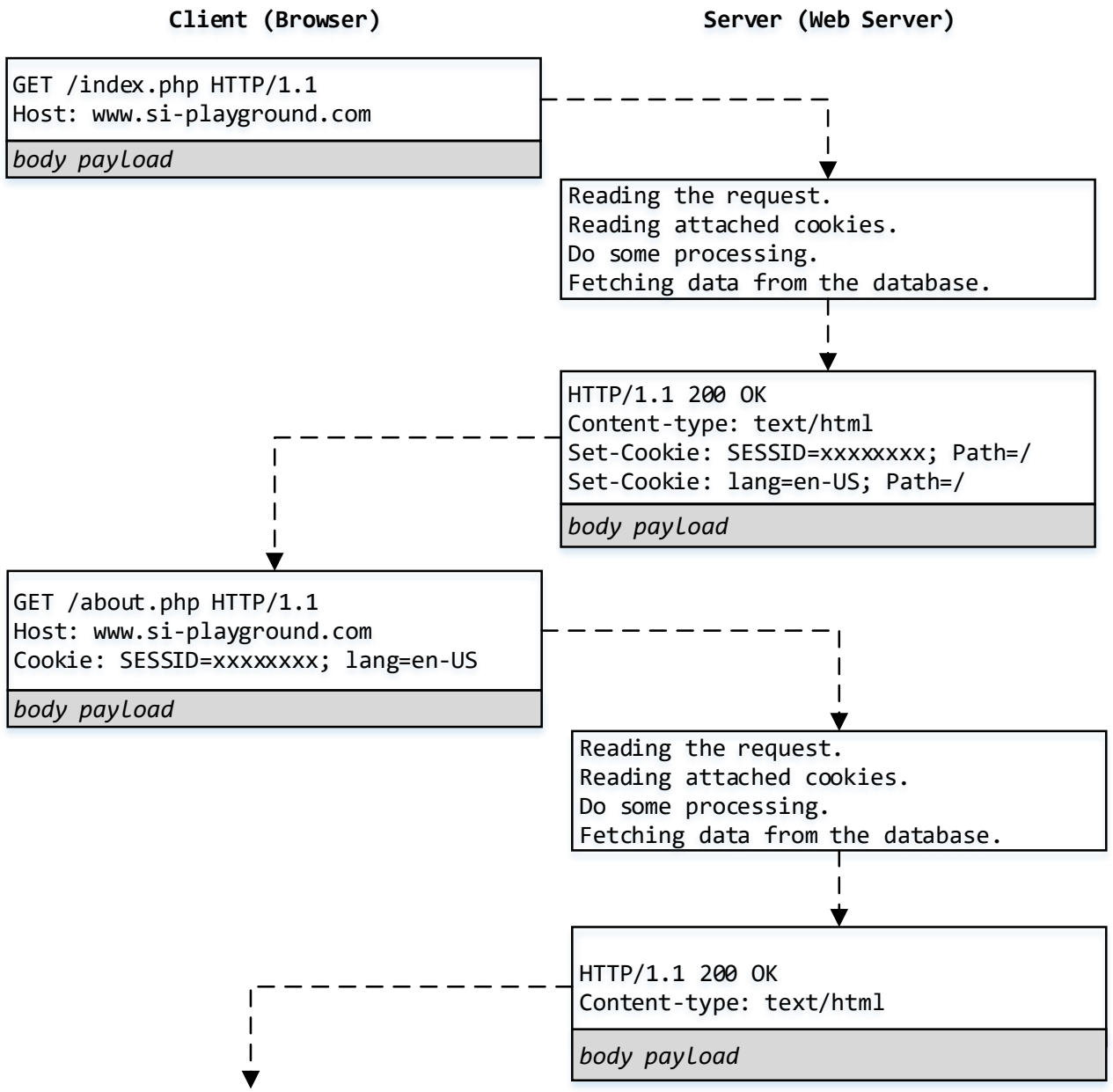
# Cookie (RFC 6265)

- A state is stored inside the browser as a key-value pair (cookie).
  - The key has to be unique.
  - The value should be short (space limitation).
  - An encryption is needed to avoid security issue.
  - An expiration time.

# Cookie (RFC 6265)

- To set a cookie, the server put a `Set-Cookie` entry in the response header.
  - The cookie configuration consists of some attributes.
  - Configuration with multiple attributes are written with semi-colon as the separator.

```
Set-Cookie: name=value; attribute; attribute
```


Client (Browser)
Server (Web Server)
GET /index.php HTTP/1.1
Host: www.si-playground.com
body payload
Reading the request.
Reading attached cookies.
Do some processing.
Fetching data from the database.
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: SESSID=xxxxxxxx; Path=/
Set-Cookie: lang=en-US; Path=/
body payload
GET /about.php HTTP/1.1
Host: www.si-playground.com
Cookie: SESSID=xxxxxxxx; lang=en-US
body payload
Reading the request.
Reading attached cookies.
Do some processing.
Fetching data from the database.
HTTP/1.1 200 OK
Content-type: text/html
body payload

# Cookie Attributes

- Name-value
  - `name=value`
- `Expires`
  - The time when the cookie will be destroyed.
- `Path`
  - Pattern that must exist in order to send the cookie.
- `Domain`
  - List of domain that is allowed to receive the cookie.
- `Secure`
  - Boolean, default: `false`.
  - To enforce cookie to be sent though `HTTPS` only.
- `HttpOnly`
  - Boolean , default: `false`.
  - To avoid access via JavaScript.
- `SameSite`
  - Restriction level for cross-origin request.

# Storing State in The Server-Side

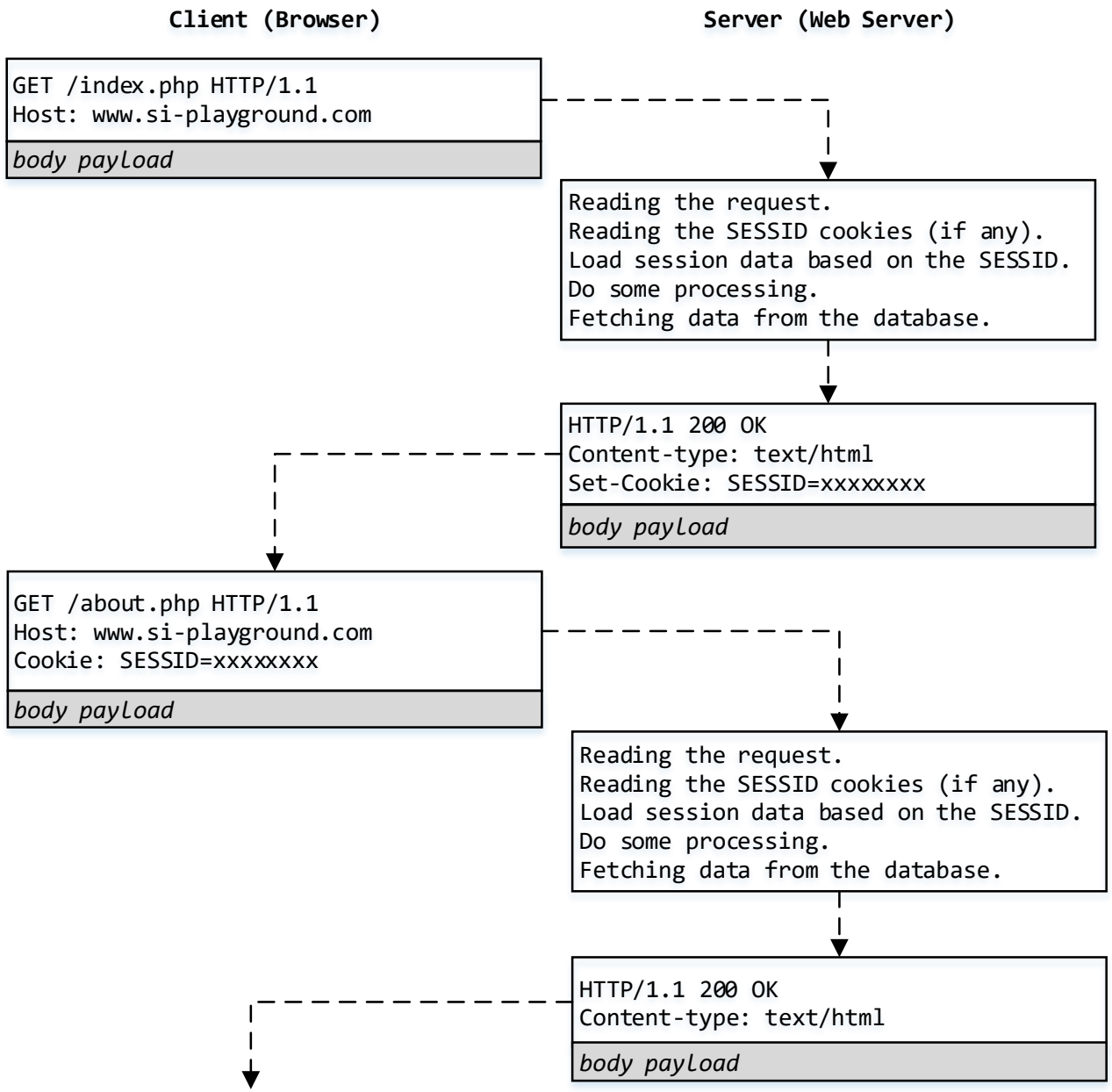
# Session (RFC 6265)

- Storing state in the client-side bring some drawbacks:
  - Prone to security problem (confidentiality and integrity)
  - Overhead in data transmission.


- As an alternative, state could be stored in the server.
  - The client will only need to store its `session identifier` (`SESSID`)
  - The `SESSID` is treated as a cookie.
  - User's session is transparent from others.

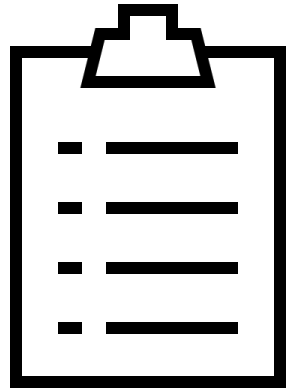
# Session

- Some consideration of using session over cookies.
  - States are temporarily stored in the server's main memory.
  - Server capacity might be a concern.
  - Way much faster processing.
  - Can be stored in the database for scalability.


Client (Browser)
Server (Web Server)
GET /index.php HTTP/1.1
Host: www.si-playground.com
body payload
Reading the request.
Reading the SESSID cookies (if any).
Load session data based on the SESSID.
Do some processing.
Fetching data from the database.
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: SESSID=xxxxxxxx
body payload
GET /about.php HTTP/1.1
Host: www.si-playground.com
Cookie: SESSID=xxxxxxxx
body payload
Reading the request.
Reading the SESSID cookies (if any).
Load session data based on the SESSID.
Do some processing.
Fetching data from the database.
HTTP/1.1 200 OK
Content-type: text/html
body payload

# To-dos

1. Understand deeply the difference between
   - managing state in the client-side and in the server-side.
   - `Path` and `Domain` attributes in the cookies.
   - `Secure`, `HttpOnly`, and `SameSite` attributes.
2. Read the RFC 6265 about the weakness of cookies:
   - Weak Confidentiality (chapter 8.5)
   - Weak Integrity (chapter 8.6)
3. REST Application uses different approach. What is it?



# References

Srinivasan, M. (2012). Web Technology: Theory and Practice. Pearson.

Tatroe, K., et. al. (2020). Programming PHP. O'Reilly.

PHP Manual https://www.php.net/manual/en/

RFC 6265, HTTP State Management Mechanism. https://tools.ietf.org/html/rfc6265#section-8.4

# Thank you