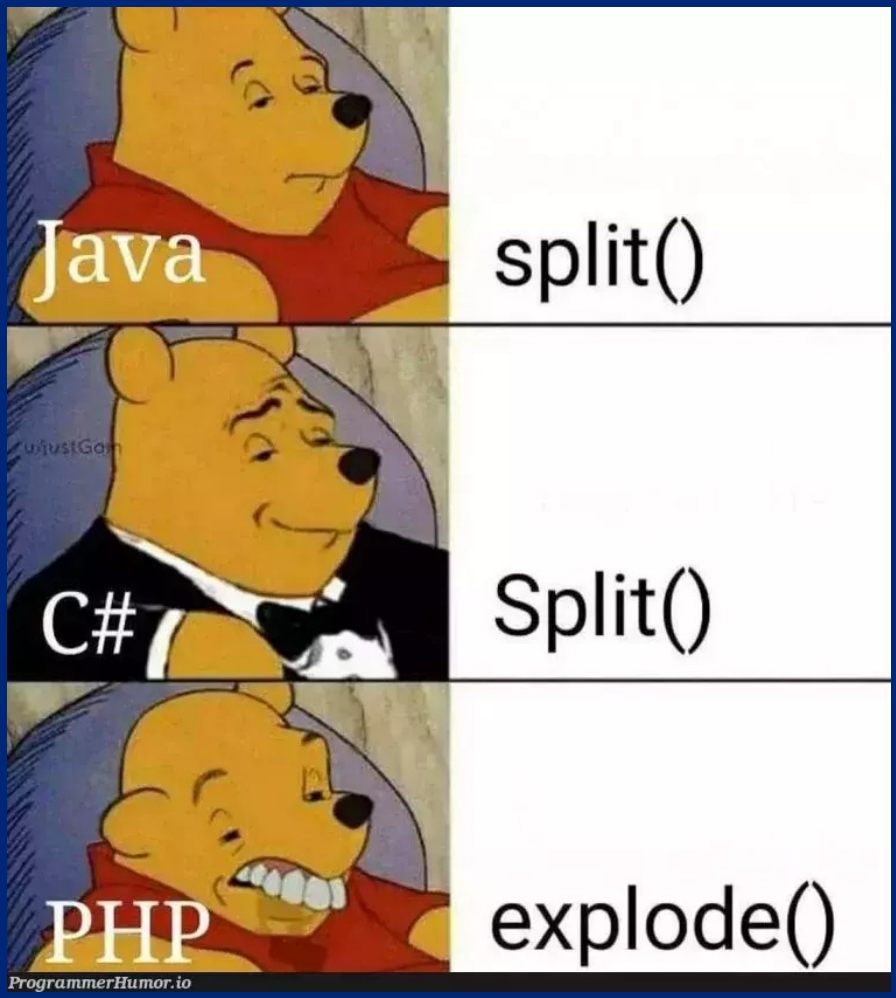
Università di Genova

PHP (3)



Marina Ribaudo, marina.ribaudo@unige.it

HTTP stateless

- Il web server non ha "memoria" delle comunicazioni HTTP successive, anche se arrivano dallo stesso client

- Per questo motivo, per realizzare applicazioni web complesse, sono stati introdotti altri meccanismi
  - Cookie (client)
  - Access control (server)
  - Session control (server)

Session control

Il controllo di sessione permette di tener traccia dell'utente durante la sua interazione con un sito web

“Suppose you are building one e-commerce site, to allow anyone to buy the product you must ask them to log-in with their user name and until they log out your system must track the user in every step, this concept is called as “session tracking”.

Now why do we need to track the session, answer is very simple. ***HTTP is stateless protocol****, and when you refresh the page, it lost everything, which your project should not!”*

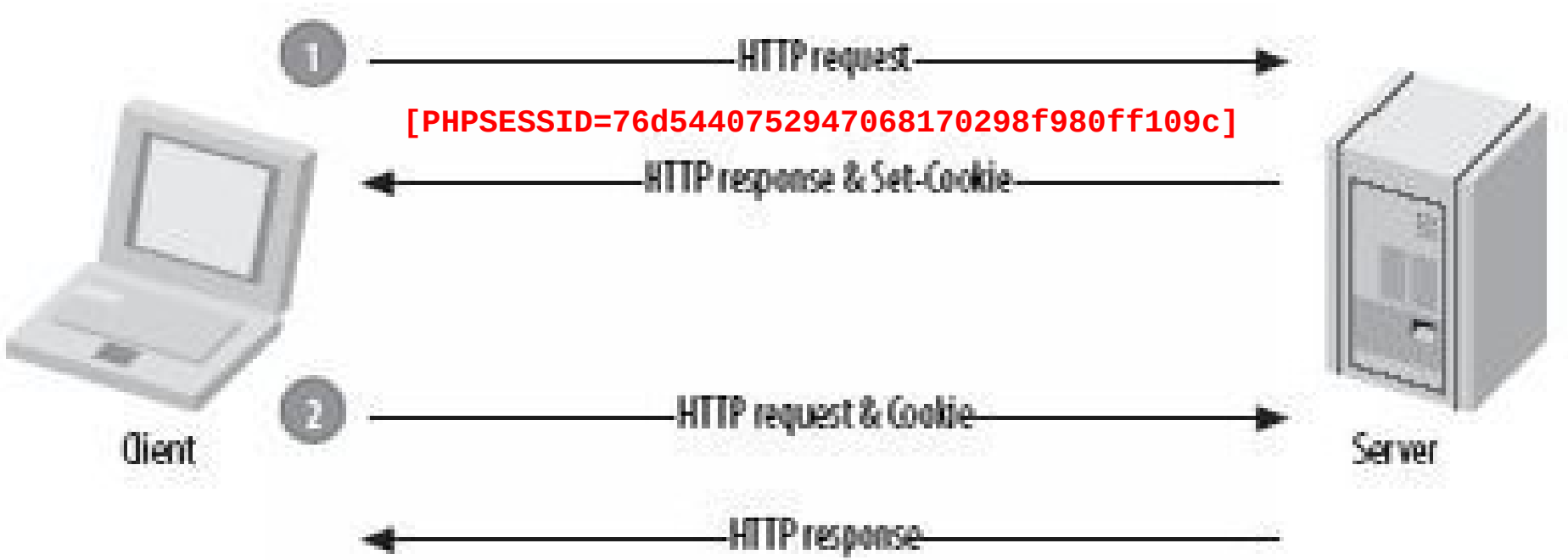
Session control

Per gestire il controllo di sessione si usa l'**array superglobale $_SESSION**

Inoltre in PHP "… a visitor accessing your web site is assigned an **unique id**, the so-called **session ID**. This is either stored in a cookie on the user side or is propagated in the URL …"

Vedi: http://www.w3schools.com/php/php_sessions.asp

Session control



[dal libro Essential PHP security]

Session control: how to

- Iniziare la sessione
- Creare le variabili di sessione
- Usare le variabili di sessione
- Rilasciare le variabili di sessione
- Chiudere la sessione

Iniziare la sessione

session_start();

Verifica se l'utente ha già un identificatore di sessione

Se non lo trova, ne crea uno, altrimenti rende accessibili le variabili di sessione già create in precedenza per quell'utente

Quando si usano le sessioni è **obbligatorio inziare tutti gli script delle pagine riservate con session_start()**

Creare le variabili di sessione

$_SESSION[“key”] = <value>;

La variabile di sessione viene creata assegnando una **chiave** all’array superglobale **$_SESSION** e un **valore** corrispondente

La variabile esiste e viene “tracciata” fino a quando non si termina la sessione

Usare le variabili di sessione

Con isset() si verifica se la variabile di sessione esiste

```
if (isset($_SESSION[“key”]))  {
    <here session variable exists>
    <present content for logged users>
}
else {
    <redirect to login page>
}
```

Se sì, l'utente è autorizzato a proseguire, altrimenti lo si rimanda alla pagina di login

Usare le variabili di sessione

Con isset() si verifica se la variabile di sessione **non** esiste

```
If (!isset($_SESSION[“key”]))  {
    <redirect to login page>
    exit();
}

<here session variable exists>
<present content for logged users>
```

Se la variabile di sessione non esiste, si rimanda alla pagina di login e si esce dallo script

Rilasciare le variabili di sessione


```
unset($_SESSION["key"]);
```

La sessione esiste ancora, ma la variabile con chiave key non è più registrata come variabile di sessione

```
session_unset();   // deprecato
```

```
$_SESSION = array();
```

Chiudere la sessione

session_destroy();

Cancella l’identificatore di sessione

Identificatore di sessione


```
<?php
        session_start();
        echo "SID: " . session_id();
?>
```

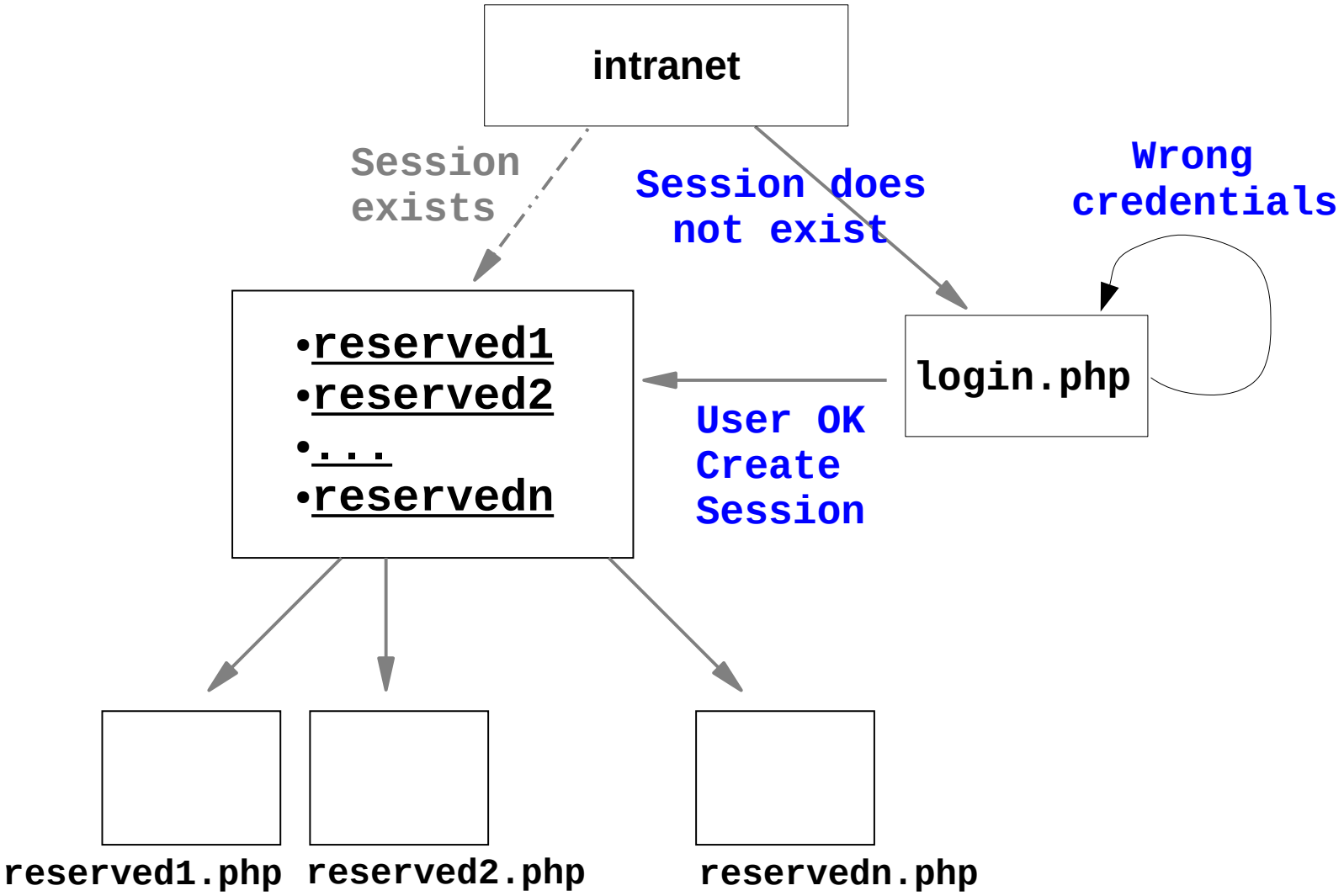
```
SID: 0a777aa640f1bad7ac3f788065a99ba6
```

Durata di una sessione

- Le sessioni in PHP hanno una durata che è definita dalla direttiva **session.gc_maxlifetime** nel file di configurazione **php.ini**
  - default 24 minuti
- Si può impostare un valore diverso per la durata di una sessione utilizzando la funzione **ini_set()**
  - ini_set('session.gc_maxlifetime', 3600);

- Una sessione può terminare anche prima della scadenza se l'utente **chiude il browser** o se il server web termina la sessione per motivi di sicurezza

Session control: esempio



Session control: esempio

- Nella fase di autenticazione
  - Si verifica la correttezza delle credenziali dell'utente
  - Se le credenziali sono corrette
    - si creano una o più variabili di sessione
    - si presentano i servizi dell'area riservata
  - Altrimenti (credenziali errate)
    - si rimanda al login

Session control: esempio

Al login

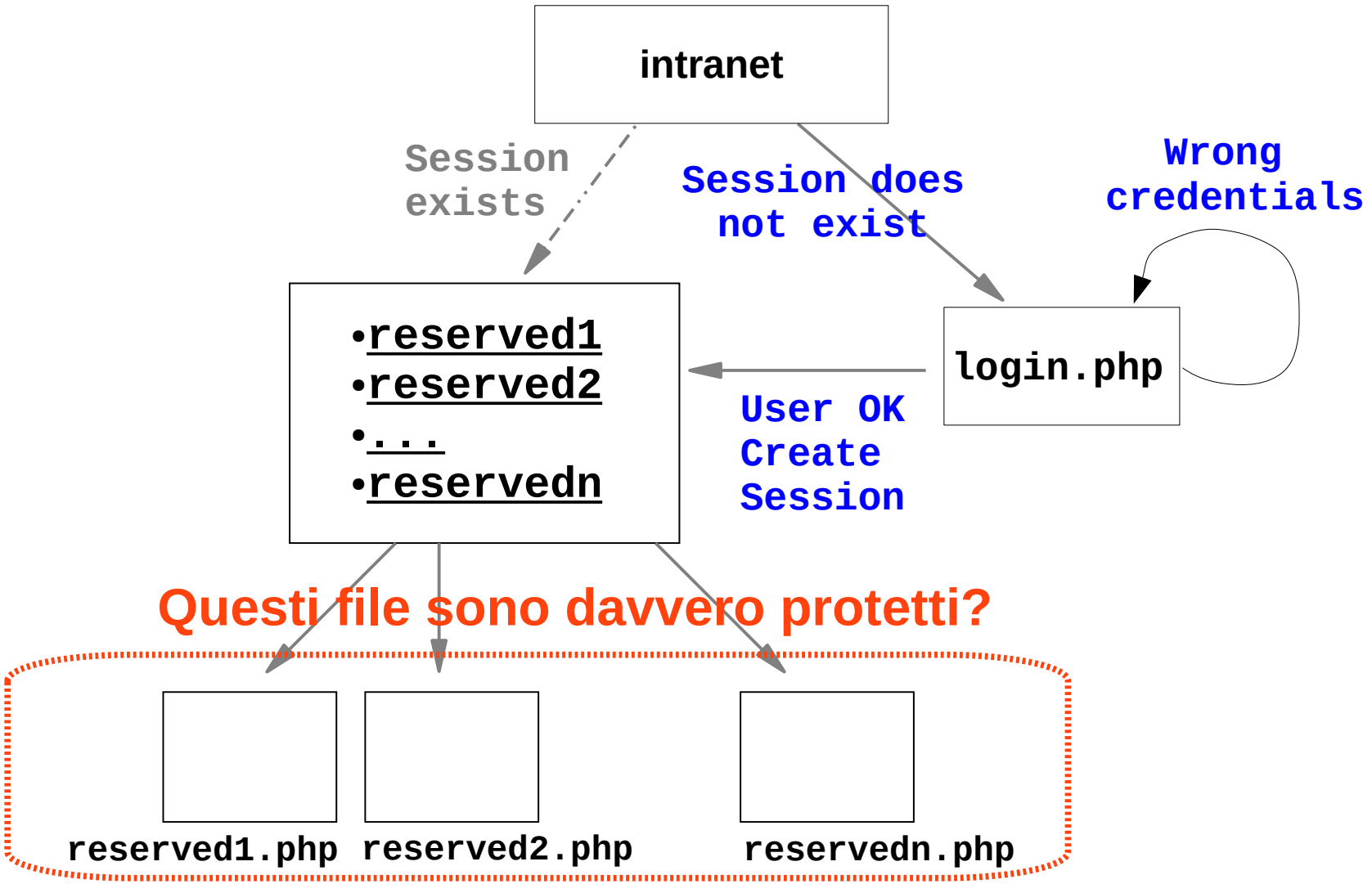
```
<?php
   session_start();

/* check if username and passwor are stored in a User table*/

if <YES> {
     $_SESSION[“key1”] = ...;
     $_SESSION[“key2”] = ...;
     $_SESSION[“key3”] = ...;
     /* present list of services */
   }
else {
      echo “<p class=’error’>Access denied, check your credentials</p>\n”;
   }

?>
```

Session control: esempio



Session control: esempio

- Durante la navigazione nell'area riservata
  - **Tutti i file** che offrono servizi **devono controllare le variabili di sessione**, non basta il controllo sul primo file!

Session control: esempio

- Durante la navigazione nell'area riservata

```
<?php
  session_start();

  If (!isset($_SESSION[“key1”])) {
      <redirect to login page>
      exit();
  }

  <show reserved page>
```

PHP: header()

- Permette di inviare header HTTP al client

- **Importante**: dovrebbe essere invocata **prima** di restituire al client dati in output, basta anche un solo spazio vuoto (o una riga vuota nel file PHP cui corrisponde il carattere “\n”) per causare un errore

PHP: esempi


```
header('Location: URL')  // redirect

header('Content-type: image/jpg') // MIME type

header('Cache-Control: no-cache, must-revalidate') //  no cache
header('Expires: Wed, 01 Jan 2023 05:00:00 GMT')  //  date in the past
```

Nota: bufferizzazione output

PHP invia al client qualunque output non appena disponibile. E' possibile **salvare temporaneamente** quello che dovrebbe essere restituito in output fino a quando non viene detto esplicitamente di spedirlo

Così facendo, **header e dati vengono inviati in modo corretto** anche se alcuni header sono stati creati dopo i primi output HTML

Per abilitare questa funzionalità si può usare la funzione **ob_start()** oppure si può attivare la direttiva nel file php.ini

output_buffering = 4096 // 4096 sono i byte bufferizzati

Nota: bufferizzazione output

“Without output buffering (the default), your HTML is sent to the browser in pieces as PHP processes through your script. With output buffering, your HTML is stored in a variable and sent to the browser as one piece at the end of your script.

Advantages of output buffering for Web developers:

1) Turning on output buffering alone decreases the amount of time it takes to download and render our HTML because it's not being sent to the browser in pieces as PHP processes the HTML.

2) All the fancy stuff we can do with PHP strings, we can now do with our whole HTML page as one variable.

3) If you've ever encountered the message "Warning: Cannot modify header information - headers already sent by (output)" while setting cookies, you'll be happy to know that output buffering is your answer.”

Da StackOverflow