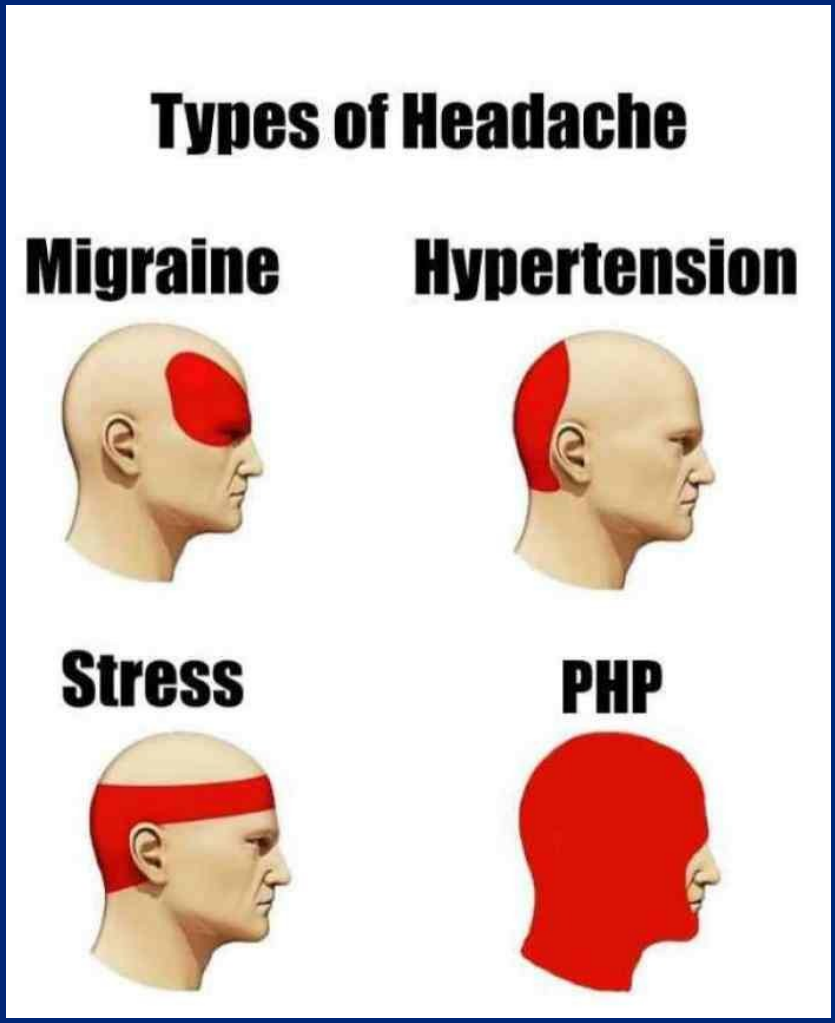
Università di Genova

PHP (2)



Marina Ribaudo, marina.ribaudo@unige.it

Gestione delle password

- Le password degli utenti non si devono **MAI memorizzare in chiaro**
  - Molti attacchi sul web hanno lo scopo di rubare le password degli utenti https://haveibeenpwned.com/
  - In caso di furto di dati, bisogna garantire che l'attaccante non riesca a decifrare le password
  - Oggi - per rispettare le regole del GDPR - bisogna proteggere i dati sensibili (e le password lo sono) https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

PHP: gestione delle password


```
/* User's password */
$pwd = “Mys3cr3tp4ssw0rd!”;
/* MD5 hash */
$hash = md5($pwd); => c0724e88060f5edba1d23a76987d5857
/* SHA1 hash */
$hash = sha1($pwd); => bfa0c4465da107449868ca6b6fb09956ea24faaa
```

PHP: gestione delle password

/* User's password */

$pwd = “Mys3cr3tp4ssw0rd!”;

/* MD5 hash */

$hash = md5($pwd); => c0724e88060f5edba1d23a76987d5857

/* SHA1 hash */

$hash = sha1($pwd); => bfa0c4465da107449868ca6b6fb09956ea24faaa

Warning It is not recommended to use this function to secure passwords, due to the fast nature of this hashing algorithm. See the Password Hashing FAQ for details and best practices.

PHP: gestione delle password

- Gli algoritmi **MD5** e **SHA-1** sono **troppo deboli** per la potenza di calcolo odierna
- Gli hash prodotti da questi algorimti sono vulnerabili ad attacchi “rainbow table” e ad attacchi a dizionario
  - https://crackstation.net/ (a volte basta Google!)
  - https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

PHP: gestione delle password

- Per proteggersi dagli attacchi oggi si usano **funzioni di hash lente da calcolare** e **con sale** (salt), una **stringa casuale** che rende ogni password unica
  - Password: Mys3cr3tp4ssw0rd!
  - Salt: HMqu9nzAB6aOVCrNfmI0Gu (sempre 22 caratteri)
  - Salted password: Mys3cr3tp4ssw0rd!HMqu9nzAB6aOVCrNfmI0Gu
- L'hash viene calcolato sulla salted password

PHP: gestione delle password

/* User's password */

$pwd = “Mys3cr3tp4ssw0rd!”;

$hash = **password_hash**($pwd, PASSWORD_BCRYPT);

$**2y**$**10**$HMqu9nzAB6aOVCrNfmI0GutJYqY6tSU09rSQ/xhX5kjJLKREYc8gu

- **2y** è l’algoritmo di hash usato (secondo parametro della funzione)
- **10** è il costo computazionale (2^10 iterazioni in questo caso)
- HMqu9nzAB6aOVCrNfmI0Gu è il sale (sempre 22 caratteri)
- Hepv.KF4zj6z4J3XXyYRye.VXnPsMA2è l’hash di password+hash (sempre 31 caratteri)

https://www.php.net/manual/en/function.password-hash.php

PHP: dati in arrivo dal client

“ … all input is evil until proven otherwise …”

I dati in input dal client devono essere **sempre validati**!

PHP ha varie funzioni che permettono di verificare il contenuto dei dati forniti in input

Il test dipende dalle abilità del programmatore...

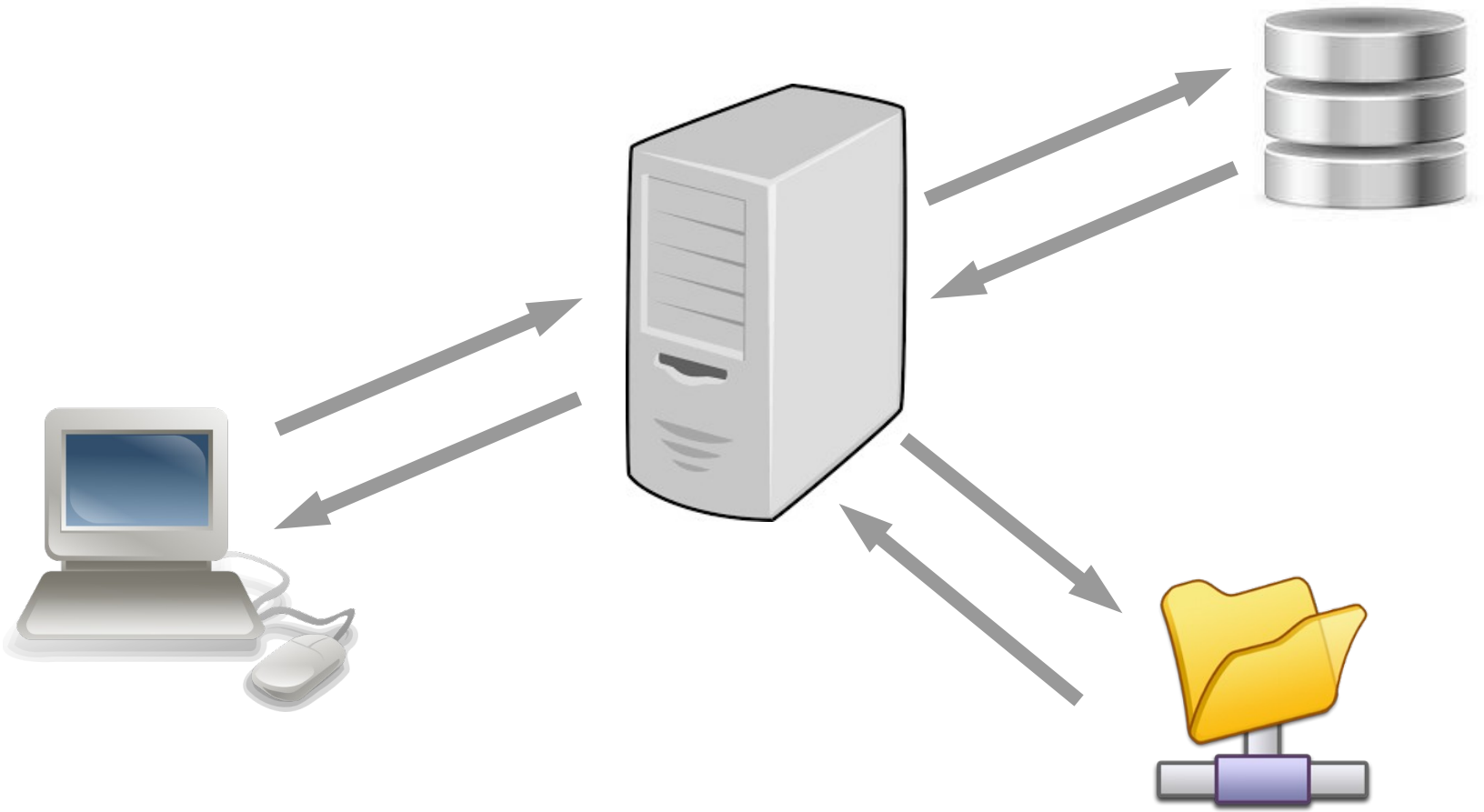
PHP: dati in arrivo dal client

*“Input validation is both the **most fundamental defense** that a web application relies upon and **the most unreliable**. A significant majority of web application vulnerabilities arise from a validation failure, so getting this part of our defenses right is essential.”*

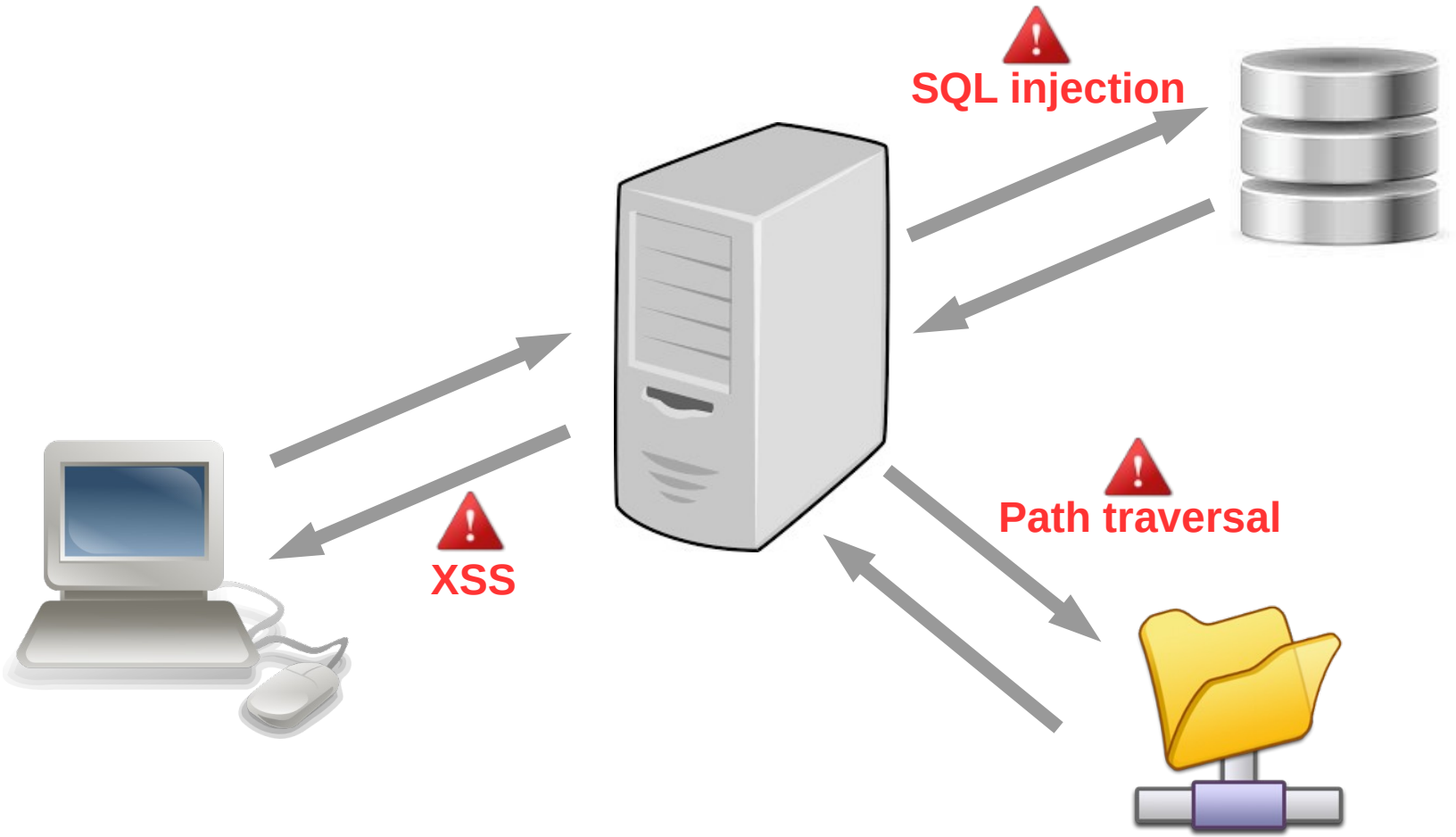
PHP: dati in arrivo dal client

*“For example, if I receive a **piece of data containing a name**, I may validate it fairly loosely to allow for apostrophes, commas, brackets, spaces, and the whole range of alphanumeric Unicode characters (not all of which need literally be alphabetic according to Western languages). As a name, **we’d have valid data which can be useful for display purposes** (it’s first intended use). However, **if we use that data elsewhere** (e.g. a database query) we will be **putting it into a new context. In that new context, some of the characters we allow would still be dangerous** - our name might actually be a carefully crafted string intended to perform an SQL Injection attack.”*

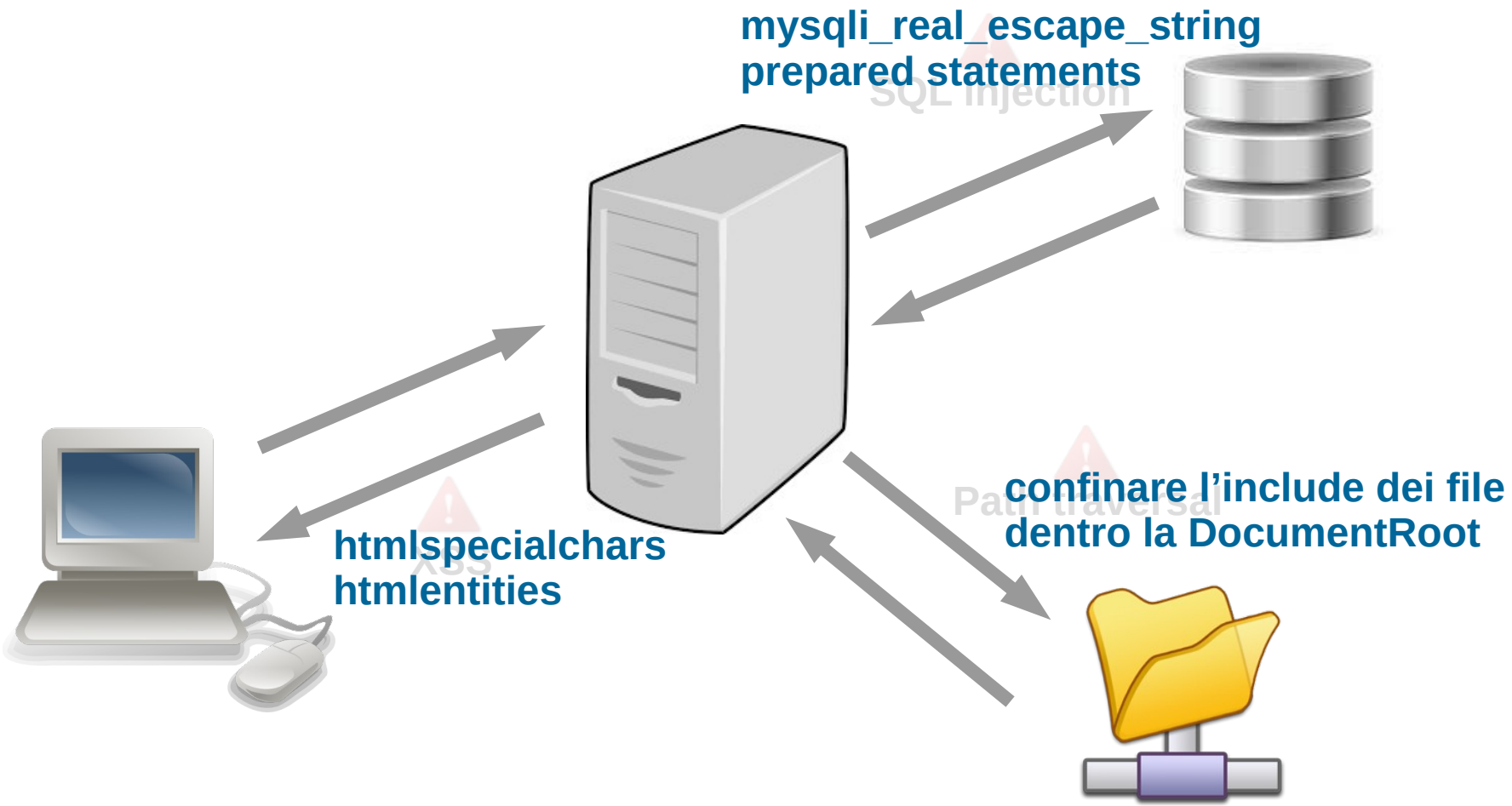
PHP: contesti di input/output



PHP: contesti di input/output



PHP: contesti di input/output



PHP: “utilities” input

isset($var)

- true (1), se $var esiste e non è NULL
- false (0) altrimenti

empty($var)

- true (1), se $var non esiste oppure il suo valore è false
- false (0), altrimenti

**non genera warning se la variabile non esiste*

PHP: “utilities” input

trim($str)

toglie spazi, \n, \r, \t, \v, \0 ad inizio e fine stringa

filter_var($str[, int $filter = FILTER_TYPE])

applica il filtro selezionato (regex) alla stringa

PHP: “utilities” output

htmlspecialchars($str, ENT_QUOTES[, $charset])

converte alcuni caratteri speciali (tra cui &, “, ‘, <, >) nelle entità HTML corrispondenti per evitare attacchi XSS (Cross-Site Scripting)

La funzione **htmlentities()** ha un comportamento simile a htmlspecialchars(), ma **converte tutti i caratteri che hanno un’entità HTML equivalente**, non solo quelli speciali

Quindi converte una gamma molto più ampia di caratteri, inclusi gli accenti, i simboli matematici e altri caratteri

PHP: parsing input

int($var) converte $var in intero, se possibile
(simile a int **intval**(mixed $var [, int $base = 10]))

Esempio: 12a -> 12, a12 -> 0

doubleval($var)

strval($var)

is_numeric($var) verifica se $var è un numero o una stringa

PHP: espressioni regolari

Si possono usare le **espressioni regolari** per definire dei **pattern** su stringhe di testo

PHP usa le espressioni regolari di Perl e offre delle **funzioni** per verificare se un certo pattern occorre in una stringa

Mediante l'uso di simboli come +,?,-,^,*,[ ],{ } si possono creare **espressioni complesse** per validare email, indirizzi IP, ecc.

PHP: espressioni regolari

Metacharacter	Description	Example
.	Matches any single character except a new line	/./ matches anything that has a single character
^	Matches the beginning of or string / excludes characters	/^PH/ matches any string that starts with PH
$	Matches pattern at the end of the string	/com$/ matches guru99.com,yahoo.com Etc.
*	Matches any zero (0) or more characters	/com*/ matches computer, communication etc.
+	Requires preceding character(s) appear at least once	/yah+oo/ matches yahoo
\	Used to escape meta characters	/yahoo+\.com/ treats the dot as a literal value
[...]	Character class	/[abc]/ matches abc
a-z	Matches lower case letters	/a-z/ matches cool, happy etc.
A-Z	Matches upper case letters	/A-Z/ matches WHAT, HOW, WHY etc.
0-9	Matches any number between 0 and 9	/0-4/ matches 0,1,2,3,4

The above list only gives the most commonly used metacharacters in regular expressions.

PHP: espressioni regolari

Esempi

/[0-9]{8,14}/

/[a-z\s]{10,20}/

/[a-zA-Z0-9\s]{1,}/

/^.{2}$/

PHP: espressioni regolari

Esempi

/[0-9]{8,14}/

/[a-z\s]{10,20}/

/[a-zA-Z0-9\s]{1,}/

/^.{2}$/

```
<?php
$str = "Ciao ciao ";
$pattern = "/^[a-z\s]{10,20}$/";
echo preg_match($pattern, $str);
?>
```

PHP: espressioni regolari

Esempi

/[0-9]{8,14}/

/[a-z\s]{10,20}/

/[a-zA-Z0-9\s]{1,}/

/^.{2}$/

```
<?php
$str = "Ciao ciao ";
$pattern = "/[0-9a-zA-Z\s]{10,20}/";
echo preg_match($pattern, $str);
?>
```

PHP: cookie

setcookie (string $**name**, string $**value**,
int $expires_or_option,
string $path, string $domain,
bool $secure, bool $httponly): bool

setcookie() defines a cookie to be sent along with the rest of the HTTP headers. Like other headers, cookies must be sent *before* any output from your script (this is a protocol restriction). This requires that you place calls to this function prior to any output, including `<html>` and `<head>` tags as well as any whitespace.

https://www.php.net/manual/en/function.setcookie.php

PHP: cookie

- Quando si torna su un sito che ha lasciato un cookie, il valore del cookie si legge nell'array superglobale **$_COOKIE[ ]**
- Per cancellare un cookie si può portare indietro la sua data di scadenza oppure assegnare al cookie il valore nullo

PHP: cookie


```
<?php
    $name=”mycookie”;
    $value = “cookie value here”;
    $expires = mktime(0,0,0,01,01,2025);
    setcookie($name,$value,$expires);

?>

<?php
   // set the expiration date to one hour ago
   setcookie("mycookie", "", time()-3600);
?>
```

PHP: cookie


```
<?php
if (isset($_COOKIE["mycookie"])) {

    <read the value of the cookie>
    <return a personalized homepage>
}
else {

    <show standard homepage>
}
?>
```

PHP: cookie

Warning: Cannot modify header information - headers already sent by
(output started at /var/www/phpexamples/registration.php:10)
in /var/www/phpexamples/registration.php on line 21

Cookies

PHP transparently supports HTTP cookies. Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the setcookie() or setrawcookie() function. Cookies are part of the HTTP header, so setcookie() must be called before any output is sent to the browser. This is the same limitation that header() has. You can use the output buffering functions to delay the script output until you have decided whether or not to set any cookies or send any headers.

https://www.php.net/manual/en/features.cookies.php

PHP: header()

- La funzione header() permette di **gestire gli header delle risposte HTTP**
  - Utile per controllare il tipo di payload, la cache del browser, il redirect
- Dovrebbe essere chiamata **prima di generare qualsiasi output** perché una volta inviato anche un solo spazio bianco al client, non è più possibile modificare gli header
- Nota: si può configurare il server per abilitare la **bufferizzazione dell'output** oppure usare la funzione PHP ob_start()

PHP: header()

- Redirect

```
<?php
  header("Location: www.unige.it");
  exit();
```

- Tipo di payload

```
<?php
   header("Content-Type: application/json");
   echo json_encode($data);
```

- Controllo della cache

```
<?php
header("Cache-Control: no-cache, must-revalidate"); // HTTP 1.1

header("Expires: Sat, 1 Jan 2000 00:00:00 GMT");  // scadenza nel passato
```


Practice
Practice
Practice