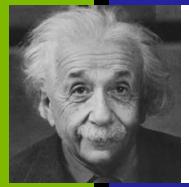
B – PHP

ES03 - Form validation

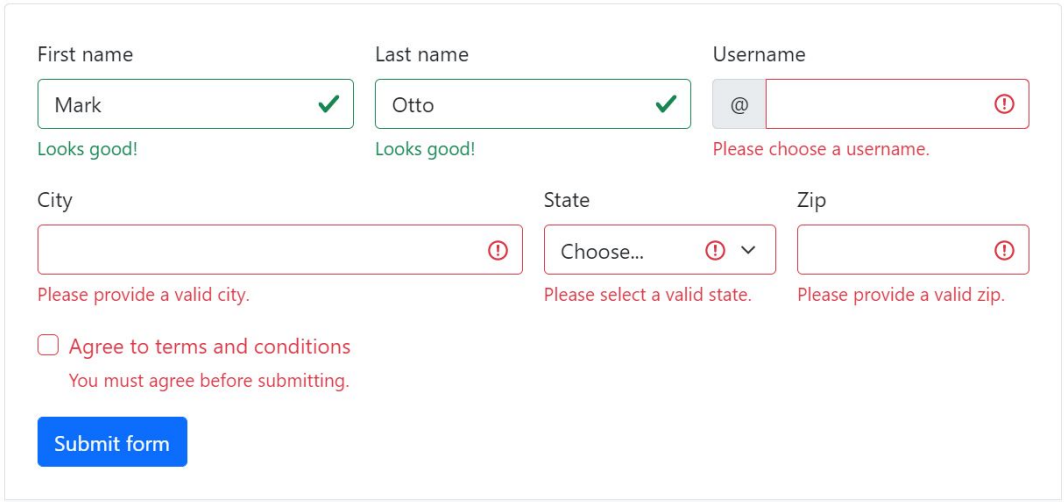
FB - 10.2024 - v1.2

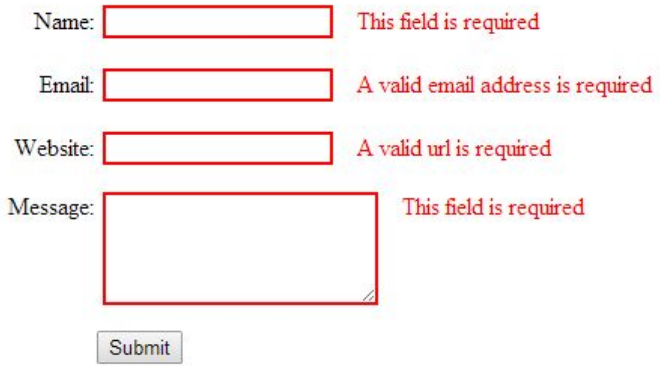


“La teoria è quando si sa tutto e niente funziona. La pratica è quando tutto funziona e nessuno sa il perché. Noi abbiamo messo insieme la teoria e la pratica: non c'è niente che funzioni... e nessuno sa il perché!”

Form validation

Verifica dei dati provenienti dal form.





Indice

- Obiettivi didattici
- Introduzione teorica
  - Stringhe
  - Funzioni
  - Form validation
- Esercitazione
- Valutazione

Obiettivi didattici

Obiettivi didattici

- le stringhe
- utilizzo di funzioni
- i form html
- validazione dei dati del form con php

Introduzione teorica

Stringhe

Le stringhe in PHP

Le stringhe in PHP sono una serie di caratteri, dove ogni carattere corrisponde ad un byte. Ciò significa che PHP supporta solo un set di caratteri di 256 caratteri.

Le stringhe possono essere create utilizzando virgolette singole o doppie, ma le virgolette doppie consentono l'interpolazione delle variabili.

PHP fornisce molte funzioni integrate per la manipolazione delle stringhe, come il calcolo della lunghezza di una stringa con la funzione strlen(), la ricerca di sottostringhe o caratteri con la funzione strpos(), la sostituzione di parte di una stringa con caratteri diversi con la funzione str_replace(), la scomposizione di una stringa con la funzione explode(), la conversione di una stringa in maiuscolo o minuscolo con le funzioni strtoupper() e strtolower(), e molte altre.

Inoltre, PHP offre un operatore di concatenazione di stringhe rappresentato dal carattere del punto (.). È possibile anche estrarre una porzione di una stringa utilizzando la funzione substr(). Per rimuovere gli spazi all'inizio o alla fine di una stringa, si possono utilizzare le funzioni trim(), ltrim() e rtrim().

Quoting

In PHP, il "quoting" si riferisce al processo di inserimento di citazioni (o delimitatori) all'interno di una stringa per indicare in modo esplicito dove inizia e dove finisce la stringa stessa. Le citazioni sono utilizzate per definire il testo letterale di una stringa. PHP supporta diverse forme di quoting, tra cui:

Single Quotes: Le stringhe racchiuse tra singole citazioni (') sono considerate "stringhe con singole citazioni" e tutto il loro contenuto è trattato come testo letterale. Le variabili all'interno di queste stringhe non vengono valutate o espanse.
Esempio:

```
$nome = 'John';
echo 'Il mio nome è $nome'; // Output: Il mio nome è $nome
```

Double Quotes: Le stringhe racchiuse tra doppie citazioni (") consentono la valutazione e l'espansione delle variabili all'interno di esse. Le sequenze di escape come \n e \t sono interpretate all'interno delle doppie citazioni.
Esempio:

```
$nome = 'John';
echo "Il mio nome è $nome"; // Output: Il mio nome è John
```

Quoting

Heredoc: Heredoc è una sintassi che consente di creare stringhe multilinea senza la necessità di citazioni. È utile quando si desidera includere blocchi di testo estesi all'interno di una stringa.
Esempio:

```
$testo = <<<EOD  oppure <<<"EOD"
Questo è un esempio di heredoc.
Puoi includere variabili come $nome.
EOD;
```

Nowdoc: Nowdoc è simile all'heredoc, ma le variabili all'interno di una stringa nowdoc non vengono valutate.
Esempio:

```
$nome = 'John';
$testo = <<<'EOD'
Questo è un esempio di nowdoc.
Le variabili come $nome non vengono valutate.
EOD;
```

Esempio di utilizzo del formato Heredoc

```
$form = <<<FORM
<form method="post">
  <label for="nome">Nome:</label><input type="text" id="nome" name="nome"><br>
  <label for="cognome">Cognome:</label><input type="text" id="cognome" name="cognome"><br>
  <label for="email">Email:</label><input type="email" id="email" name="email"><br>
  <input type="submit" value="Invia">
</form>
FORM;

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $nome = $_POST['nome'];
  $cognome = $_POST['cognome'];
  $email = $_POST['email'];
  $stringa = <<<TESTO
    I dati inseriti nel form sono: <br/>
    Nome e Cognome: $nome $cognome <br/>
    Email: $email
  TESTO;
  $html = $stringa;
} else {
  $html = $form;
}
```

Heredoc e Nowdoc

I formati Heredoc e Nowdoc in PHP sono entrambi utilizzati per definire stringhe letterali che possono contenere singole o doppie virgolette e variabili.

La differenza principale tra i due formati è che Heredoc esegue l'interpolazione delle variabili, mentre Nowdoc non lo fa.

Inoltre, il formato Heredoc consente l'uso di sequenze di escape e l'interpretazione delle variabili, mentre il formato Nowdoc non consente l'uso di sequenze di escape e non interpreta le variabili.

Un'altra differenza tra i due formati è che Heredoc utilizza le virgolette doppie per definire la stringa, mentre Nowdoc utilizza le virgolette singole.

Infine, con l'arrivo di PHP 7.3, è stata introdotta una sintassi più flessibile per Heredoc e Nowdoc, che consente di indentare il marcatore di chiusura e di rimuovere il requisito di una nuova riga dopo il marcatore di chiusura.

Escaping

L'escaping nelle stringhe PHP si riferisce alla pratica di inserire un carattere di escape, ovvero un carattere preceduto da un backslash (\), per rappresentare caratteri speciali all'interno di una stringa. Ad esempio, per rappresentare una singola virgoletta all'interno di una stringa racchiusa tra virgolette doppie, è necessario utilizzare il carattere di escape (\) prima della virgoletta, in questo modo: `"L'auto è \"rossa\""`.

In PHP, i caratteri speciali che richiedono l'utilizzo di un carattere di escape includono le virgolette singole e doppie, il backslash stesso, il carattere di nuova riga (\n), il carattere di ritorno a capo (\r), il carattere di tabulazione (\t), il carattere di allineamento verticale (\v), il carattere di cancellazione (\f) e altri [1].

L'escaping è spesso utilizzato quando si manipolano stringhe che contengono caratteri speciali, ad esempio quando si creano query SQL o quando si manipolano stringhe HTML. In questi casi, è importante utilizzare l'escaping per evitare errori di sintassi o vulnerabilità di sicurezza.

Operatore ternario

Operatore ternario

In PHP, l’operatore ternario permette di scrivere un'espressione "if-else" in una singola riga. La sintassi è la seguente:

```
espressione_condizione ? espressione_vera : espressione_falsa;
```

L'operatore ternario si presenta come una condizione seguita da un "?" e da due espressioni separate da ":". La prima espressione (espressione_vera) viene valutata e restituita se la condizione è vera, mentre la seconda espressione (espressione_falsa) viene valutata e restituita se la condizione è falsa.

Operatore ternario

Esempio:

```
$x = 5;
$y = ($x > 3) ? "x è maggiore di 3" : "x non è maggiore di 3";
echo $y; //stamperà "x è maggiore di 3"
```

In questo esempio, la condizione è $x > 3. Se questa è vera, la prima espressione "x è maggiore di 3" viene restituita e assegnata alla variabile $y. Altrimenti, la seconda espressione "x non è maggiore di 3" viene restituita e assegnata alla variabile $y.

```
username = isset($_POST["user"]) ? $_POST["user"] : "";
```

```
if (isset($_POST["user"])) {
    $username = $_POST["user"];
} else {
    $username = "";
}
```

not null coalescing operator

not null coalescing operator

L'operatore "??", not null coalescing operator, è stato introdotto in PHP 7 e serve a restituire il primo valore che non è null. La sintassi è la seguente:

```
$valore = $operando1 ?? $operando2 ?? $operando3;
```

In questo esempio, viene valutato il primo operando. Se non è null, il suo valore viene assegnato a $valore. Se invece è null, viene valutato il secondo operando e così via, finché non viene trovato un valore che non sia null.

L'operatore "??", quindi, è molto utile per assegnare un valore di default a una variabile se non è ancora stata assegnata un valore. Ad esempio:

```
$nome = $_GET['nome'] ?? 'Anonimo';
```

In questo esempio, se la chiave nome esiste nell'array $_GET, il suo valore viene assegnato a $nome. Altrimenti, viene assegnato il valore "Anonimo".

not null coalescing operator

```
if (isset($_POST["user"])) {
    $username = $_POST["user"];
} else {
    $username = "";
}
```

Possiamo scriverlo in forma più compatta:

```
$username = isset($_POST["user"]) ? $_POST["user"] : "";
```

o ancora meglio:

```
$username = $_POST["user"] ?? "";
```

Funzioni

Le Funzioni in PHP

- Introduzione
- Definizione di funzione, chiamata alla funzione
- Restituzione valori
- Restituzione di più valori
- Scope variabili nelle funzioni. Variabili locali e globali
- Variabili statiche
- Argomenti della funzione.
  - Passaggio di argomenti per valore
  - Passaggio di argomenti per riferimento
  - Valori impliciti o parametri opzionali
  - Numero variabile di argomenti
  - Function named parameters
- Funzioni built-in
- Funzioni ricorsive
- Funzioni anonime
- Funzioni arrow
- Variabili funzione
- Librerie di funzioni

introduzione

Le funzioni in PHP sono uno degli strumenti più importanti a disposizione dei programmatori. Sono un costrutto che racchiude una sequenza di istruzioni in un unico blocco di codice, prende un input sotto forma di parametro, effettua delle elaborazioni sui dati e restituisce un output.

PHP mette a disposizione un gran numero di funzioni integrate, come print_r(), isset(), trim(), etc. Tuttavia, è frequente la necessità di creare funzioni personalizzate (chiamate anche funzioni definite dall'utente) che eseguano specifici compiti.

La sintassi di una funzione PHP personalizzata è la seguente:

```
function nome_funzione($parametro1, $parametro2, ...) {
    // istruzioni
    return $valore;
}
```

dove "nome_funzione" è il nome che vogliamo dare alla funzione, "$parametro1, $parametro2, ..." sono i parametri in ingresso, "$valore" è il valore di ritorno della funzione.

Definizione e chiamata

Definizione di funzione: Per definire una funzione in PHP, si utilizza la parola chiave function, seguita dal nome della funzione e un paio di parentesi tonde. Il corpo della funzione è racchiuso tra parentesi graffe. Ad esempio:

```
function miaFunzione() {
    // corpo della funzione
}
```

Chiamata alla funzione: Per chiamare una funzione, è sufficiente utilizzare il suo nome seguito da parentesi tonde. Ad esempio:

```
miaFunzione();
```

Restituzione valori

la restituzione di valori da una funzione avviene tramite l'istruzione "return". È possibile restituire qualsiasi tipo di valore, inclusi numeri, stringhe, array e oggetti. [ref]

Ad esempio:

```
function somma($a, $b) {
    return $a + $b;
}
$risultato = somma(5, 3); // $risultato conterrà 8
```

Restituzione di più valori

è possibile restituire più valori da una funzione utilizzando un array o un oggetto. Ad esempio, la seguente funzione "getIndirizzo" restituisce un array contenente l'indirizzo:

```
function getIndirizzo() {
    $indirizzo = array("via" => "Via Roma","città" => "Milano");
    return $indirizzo;
}
$indirizzo = getIndirizzo();
echo $indirizzo["via"] ."-". $indirizzo["città"];; // Output: Via Roma - Milano
```

Inoltre, è possibile utilizzare la funzione "list" per assegnare i valori restituiti da una funzione a una serie di variabili. Ad esempio, la seguente funzione "getMultipli" restituisce un array contenente i primi tre multipli di un numero:

```
function getMultipli($num) {
    $doppio = $num * 2; $triplo = $num * 3;
    return array($doppio, $triplo);
}
$a = 7;
list($doppio, $triplo) = getMultipli($a);
echo "I primi multipli di $a sono: $doppio, $triplo";
```

Scope variabili nelle funzioni. Variabili locali e globali

Quando si definisce una variabile all'interno di una funzione, è possibile accedere solo a quella variabile all'interno della funzione e si dice che la variabile è locale alla funzione. Variabili esterne alla funzione aventi lo stesso nome non vengono modificate. Per accedere ad una variabile globale bisogna dichiararla come global all'interno della funzione:

```
$value = 1;

function simple() {

    global $value;
    $value = 2;
}
```

https://zetcode.com/php/function/

Variabili statiche

In PHP, è possibile definire variabili statiche all'interno di una funzione utilizzando la parola chiave "static". Le variabili statiche mantengono il loro valore anche dopo l'uscita dalla funzione e sono accessibili solo all'interno della funzione stessa. Ad esempio:

```
function miaFunzione() {
    static $contatore = 0;
    $contatore++;
    echo $contatore;
}

miaFunzione(); // stampa 1
miaFunzione(); // stampa 2
miaFunzione(); // stampa 3
```

In questo caso, la variabile "$contatore" viene dichiarata come statica all'interno della funzione "miaFunzione" e viene incrementata ad ogni chiamata della funzione

Passaggio di argomenti per valore

In PHP, è possibile passare gli argomenti alle funzioni in due modi: per valore e per riferimento.

Di default, gli argomenti vengono passati per valore, il che significa che viene fatta una copia del valore di ogni variabile e solo questa copia viene usata dalla funzione. In questo modo, se il valore viene modificato all'interno della funzione, il cambiamento non si propaga anche dopo la chiamata.

```
function miaFunzione($argomento) {
    $argomento = $argomento + 1;
    return $argomento;
}

$a = 5;
$b = miaFunzione($a);

echo $a; // Output: 5
echo $b; // Output: 6
```

Passaggio di argomenti per riferimento

Per passare gli argomenti per riferimento, è necessario utilizzare l'operatore "&" nella dichiarazione della funzione e nella chiamata della funzione stessa. In questo modo, la funzione riceverà un riferimento alla variabile originale e non una copia del suo valore. In questo modo, se il valore viene modificato all'interno della funzione, il cambiamento si propaga anche dopo la chiamata. Il passaggio per riferimento è utile quando si desidera modificare il valore di una variabile all'interno della funzione e far sì che la modifica sia visibile anche all'esterno della funzione.

```
function miaFunzione(&$argomento) {$argomento = $argomento + 1;}
$a = 5;
miaFunzione($a);
echo $a; // Output: 6

function scambia(&$a, &$b) {
    $temp = $a;
    $a = $b;
    $b = $temp;
}
$x = 5; $y = 10;
echo "Prima dello scambio: x = $x, y = $y <br>";
scambia($x, $y);
echo "Dopo lo scambio: x = $x, y = $y";
```

Valori impliciti o parametri opzionali

In PHP, è possibile creare funzioni con parametri opzionali, ovvero con valori di default. In questo modo, se un parametro opzionale non viene specificato nella chiamata della funzione, viene utilizzato il valore predefinito. Questo è utile quando si desidera fornire un valore predefinito per un parametro, ma si vuole anche consentire all'utente di specificare un valore diverso se necessario.
Ad esempio, la seguente funzione "saluta" accetta due parametri, ma il secondo parametro ha un valore predefinito:

```
function saluta($nome, $saluto = "Ciao") {
    echo "$saluto, $nome!";
}

saluta("Mario"); // Output: Ciao, Mario!
saluta("Luigi", "Buongiorno"); // Output: Buongiorno, Luigi!
```

In questo caso, la funzione "saluta" accetta due parametri, ma il secondo parametro ha un valore predefinito di "Ciao". Se il secondo parametro non viene specificato nella chiamata della funzione, viene utilizzato il valore predefinito. Tuttavia, se viene specificato un valore diverso, viene utilizzato il valore specificato.

Numero variabile di argomenti

Puoi definire una funzione con un numero variabile di argomenti utilizzando l'operatore ... nell'elenco degli argomenti.

```
function somma(...$numeri) {
    $acc = 0;
    foreach ($numeri as $n) {
        $acc += $n;
    }
    return $acc;
}
echo somma(1, 2, 3, 4);

function miaFunzione() {
    $argomento1 = func_get_arg(0);
    $argomento2 = func_get_arg(1);
    echo "<br>$argomento1 - $argomento2";
}
miaFunzione(5, "ciao");
```

http://204.216.213.176/inf3php/ES03/func_var_args.php

Function named parameters

PHP supporta i parametri nominati, che ti consentono di passare argomenti specificando il nome del parametro durante la chiamata.

```
function inviaEmail($destinatario, $oggetto, $testo, $firma="Saluti da, Mario Rossi") {
    echo <<<TESTO
    Gli argomenti della funzione sono:<br>
    destinatario: $destinatario<br>
    oggetto: $oggetto<br>
    testo: $testo<br>
    firma: $firma<br>
    TESTO;

}
inviaEmail(oggetto: "Saluti", testo: "Ciao!", destinatario: "esempio@email.com");
```

http://204.216.213.176/inf3php/ES03/func_named_arg.php

Funzioni built-in

PHP ha oltre 1000 funzioni built-in che possono essere chiamate direttamente all'interno di uno script per eseguire una specifica attività. Le funzioni built-in di PHP sono suddivise in diverse categorie, tra cui:

- Stringhe: funzioni per manipolare stringhe di testo.
- Numeri: funzioni per manipolare numeri.
- Date e tempo: funzioni per manipolare date e orari.
- Array: funzioni per manipolare array.
- Filesystem: funzioni per manipolare file e directory.
- Database: funzioni per connettersi e manipolare database.
- Matematica: funzioni matematiche avanzate.
- XML: funzioni per manipolare dati XML.
- Mail: funzioni per inviare e ricevere email.
- HTTP: funzioni per manipolare richieste e risposte HTTP

Alcune delle funzioni built-in più comuni e utilizzate sono:

"echo" e "print": utilizzate per stampare testo e variabili a schermo.
"strlen": utilizzata per ottenere la lunghezza di una stringa.
"substr": utilizzata per ottenere una sottostringa di una stringa.
"explode" e "implode": utilizzate per convertire una stringa in un array e viceversa.
"array_push" e "array_pop": utilizzate per aggiungere e rimuovere elementi da un array.
"count": utilizzata per ottenere il numero di elementi in un array.
"date": utilizzata per ottenere la data e l'ora corrente.
"file_get_contents" e "file_put_contents": utilizzate per leggere e scrivere il contenuto di un file.
"mysqli_connect" e "mysqli_query": utilizzate per connettersi e interrogare un database MySQL

Funzioni ricorsive

In PHP, è possibile definire funzioni ricorsive, ovvero funzioni che si richiamano da sole. Le funzioni ricorsive sono utili quando si ha la necessità di risolvere un problema che può essere suddiviso in sottoproblemi più piccoli. Ad esempio:

```
function fattoriale($n) {
    if ($n == 0) {
        return 1;
    } else {
        return $n * fattoriale($n - 1);
    }
}

$risultato = fattoriale(5); // $risultato conterrà il valore 120
```

In questo caso, viene definita una funzione "fattoriale" che calcola il fattoriale di un numero utilizzando la ricorsione. La funzione si richiama da sola fino a quando il valore di "$n" non diventa 0 [rif]

Funzioni anonime

In PHP, è possibile definire funzioni anonime utilizzando la parola chiave "function". Le funzioni anonime non hanno un nome e possono essere assegnate ad una variabile o passate come parametro ad un'altra funzione [ref]. Ad esempio:

```
$funzioneAnonima = function($valore) {
    // istruzioni
};

$funzioneAnonima(5);
```

In questo caso, viene definita una funzione anonima e assegnata alla variabile "$funzioneAnonima". Successivamente, la funzione viene chiamata passando il valore 5 come parametro

Funzioni arrow

In PHP 7.4 è stata introdotta la sintassi delle funzioni arrow, ovvero la possibilità di definire funzioni anonime utilizzando la sintassi delle frecce. Ad esempio:

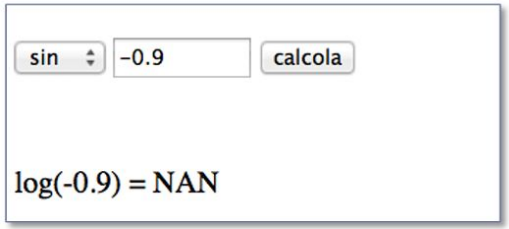
```
$funzioneArrow = fn($valore) => $valore * 2;

$risultato = $funzioneArrow(5); // $risultato conterrà il valore 10
```

In questo caso, viene definita una funzione anonima utilizzando la sintassi delle frecce e assegnata alla variabile "$funzioneArrow". Successivamente, la funzione viene chiamata passando il valore 5 come parametro e il risultato viene assegnato alla variabile "$risultato" [rif]

Variabili funzione

- Sono supportate le variabili funzione
  - Una variabile `$f` contiene il nome di una funzione come stringa
  - La chiamata della funzione si può ottenere come `$f(parametri)`

```
<?php
  $functs = array("sin","cos","tan","exp","log");
  $f = $_REQUEST['functID'];
  $x = $_REQUEST['x'];
?>
<form action="15-functiondemo.php" >
  <select name="functID" >
  <?php
    foreach($functs as $fname)
      echo "<option>${fname}</option>";
  ?>
  </select>
  <input type="text" size="8" name="x" />
  <input type="submit" value="calcola"/>
</form><br /><br />
<?php
  if(!empty($f) && !empty($x)) {
    $result = $f($x);
    echo "${f}(${x}) = ${result}";
  }
?>
```



Librerie di funzioni

E’ possibile creare librerie di funzioni per riutilizzare il codice in più di un programma. Una libreria di funzioni è una raccolta di funzioni di utilità che possono essere richiamate da altri programmi.
Per creare una libreria di funzioni in PHP, è possibile definire le funzioni in un file separato e includere il file in ogni programma che utilizza le funzioni. Ad esempio, la seguente libreria di funzioni "matematica.php" contiene alcune funzioni matematiche:

```
function somma($a, $b) {return $a + $b;}
function differenza($a, $b) {return $a - $b;}
function moltiplicazione($a, $b) {return $a * $b;}
function divisione($a, $b) {return $a / $b;}
```

In questo caso, la libreria di funzioni "matematica.php" contiene quattro funzioni matematiche. Quando un programma ha bisogno di utilizzare una di queste funzioni, può includere il file "matematica.php" utilizzando la funzione "include" o “require”:

```
include "matematica.php";
$a = 5;
$b = 3;
echo somma($a, $b); // Output: 8
echo moltiplicazione($a, $b); // Output: 15
```

La differenza tra le funzioni "include" e "require" in PHP è che "include" genera un warning se il file specificato non viene trovato o non esiste, mentre "require" genera un errore fatale e blocca l'esecuzione dello script. In altre parole, "require" è più severo di "include" e richiede che il file specificato esista e sia disponibile per l'esecuzione dello script.
Entrambe le funzioni sono utilizzate per includere un file esterno in un programma PHP. L'inclusione di un file esterno può essere utile per riutilizzare il codice in più di un programma. Inoltre, l'inclusione di un file esterno può semplificare il codice e renderlo più leggibile.

Form validation

Client-side Form validation

La validazione dei dati provenienti dai form è un'operazione importante per garantire la sicurezza e l'affidabilità dei dati inseriti dagli utenti. La validazione dei dati dei form client-side può essere effettuata utilizzando gli attributi HTML5 e tipi di input specifici. Ecco una lista degli attributi che permettono la validazione dei dati inseriti in un form in HTML5:

- **required**: indica che un campo di input è obbligatorio.
- **pattern**: permette di specificare un'espressione regolare che deve essere soddisfatta dal valore inserito in un campo di input.
- **minlength e maxlength**: specificano la lunghezza minima e massima che può avere un campo di input di tipo "text" o "password".
- **min e max**: specificano il valore minimo e massimo che possono essere inseriti in un campo di input di tipo "number" o "range".
- **step**: specifica l'incremento o il decremento del valore di un campo di input di tipo "number" o "range".
- **type**: specifica il tipo di input, ad esempio "email", "tel", "date", "color", "range", "file", "number", "search", "time", "url", "datetime-local", "month", "week", "hidden".
- **autocomplete**: specifica se il browser deve suggerire o meno i valori inseriti in un campo di input.

HTML5 ha introdotto nuovi tipi di input come "email", "tel", "date", "color", "range", "file", "number", "search", "time", "url", "datetime-local", "month", "week", "hidden" e nuovi elementi come <datalist>, <output>, <progress>, <meter>, <keygen>, <details>, <summary>, <article>, <section>, <nav>, <header>, <footer>, <aside>, <figure>, <figcaption> che semplificano la validazione dei dati inseriti nei form. [ref]

Attributi per gli elementi dei form

Esistono diversi altri attributi che possono essere utilizzati per validare i dati inseriti in un form, come:

min, max, pattern, step, required, autocomplete, autofocus, placeholder, readonly, disabled, multiple, size, maxlength, minlength, list, checked, value, name, id, class, style, title, lang, dir, tabindex, accesskey, onfocus, onblur, onselect, onchange, onsubmit, onreset, onclick, ondblclick, onmousedown, onmouseup, onmousemove, onmouseover, onmouseout, onkeypress, onkeydown, onkeyup [ref]

Server-side Form validation

In PHP, la validazione dei dati può essere effettuata utilizzando le funzioni di validazione e sanitizzazione fornite dal linguaggio.

Ecco alcuni passi comuni per la validazione dei dati provenienti dai form in PHP:

- Rimuovere eventuali caratteri superflui dai dati inseriti dall'utente, come spazi, tabulazioni e ritorni a capo, utilizzando la funzione trim().
- Verificare che i dati obbligatori siano stati inseriti dall'utente.
- Verificare che i dati inseriti rispettino i formati previsti, ad esempio che un indirizzo email contenga il carattere "@" e un punto.
- Verificare che i dati inseriti siano sicuri e non contengano codice malevolo, come script JavaScript o SQL injection.

Per effettuare la validazione dei dati, è possibile utilizzare le funzioni di validazione e sanitizzazione fornite da PHP, come filter_var(), preg_match() e htmlspecialchars().

Form validation

Ecco un esempio di codice PHP per la validazione dei dati di un form:

```
<?php
// Verifica che il form sia stato inviato
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Rimuove eventuali spazi superflui dai dati inseriti dall'utente
  $name = trim($_POST["name"]);
  $email = trim($_POST["email"]);

  // Verifica che i dati obbligatori siano stati inseriti dall'utente
  if (empty($name)) {
    $name_error = "Il nome è obbligatorio";
  }

  // Verifica che l'indirizzo email sia valido
  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $email_error = "L'indirizzo email non è valido";
  }

  // Verifica che i dati inseriti siano sicuri
  $name = htmlspecialchars($name);
  $email = htmlspecialchars($email);
}
?>
```

Gestore della Form: Controllo dei parametri

```
/* Verifica dei dati immessi */

/* Elimina spazi superflui */
$cognome   = trim($cognome);
$commento  = trim($commento);
$matricola = trim($matricola);

/* la variabile "errore" indica se tra i dati abbiamo
   trovato un errore */
$errore=FALSE;

/* verifica che almeno uno tra matricola e cognome siano
   non vuoti */
if (!$cognome && !$matricola) {
  echo "<B>Errore! Almeno uno tra nome e matricola devono
           essere specificati!</B><BR />";
  $errore=TRUE;
};
```

Gestore della Form: Controllo dei parametri

```php
/* verifica che la matricola sia numerica */
if ($matricola && ! ctype_digit($matricola)) {
  echo "<B>Errore! Matricola deve essere numerica!</B><BR />";
  $errore=TRUE;
};

/* ... e il cognome alfabetico */
if (preg_match('/^[a-zA-Z]*$/', $cognome)) {
  echo "<B>Errore! Cognome deve essere alfabetico!</B><BR />";
  $errore=TRUE;
};


if (!$errore) {
  /* Inizia la costruzione della query */
  ...
};
```

Controlli sui campi

- Controllare se il campo richiesto è stato inserito (campo obbligatorio)
- PHP: Filter - Manual
- PHP: filter_var - Manual (guida)
- PHP: ctype_digit - Manual
- PHP: is_numeric - Manual
- PHP: is_string - Manual
- PHP: checkdate - Manual

Regular Expressions (Regex)

Le espressioni regolari (regex) sono uno strumento potente per eseguire operazioni di ricerca, sostituzione e convalida su stringhe. Vengono ampiamente utilizzate per analizzare, elaborare e manipolare testi.

Un'espressione regolare è una sequenza di caratteri che forma un modello di ricerca. Quando cerchi dati in un testo, puoi utilizzare questo modello di ricerca per descrivere ciò che stai cercando. Un'espressione regolare può essere un singolo carattere o un “pattern” più complicato. Le espressioni regolari possono essere utilizzate per eseguire tutti i tipi di operazioni di ricerca e sostituzione di testo. [ref1][ref2]

Esempio - controllo indirizzo email

```
$email = "name@company.com";
if (preg_match("/^[a-zA-Z0-9._-]+@[a-zA-Z0-9-]+\.[a-zA-Z.]{2,5}$/", $email)) {
     echo "$email is a valid email address";
} else {
  echo "$email is NOT a valid email address";
}
```

- “”/.../”” inizio e fine dell'espressione regolare
- ^ indica l'inizio della stringa.
- [a-zA-Z0-9._-]+ corrisponde a uno o più caratteri alfanumerici, punti, trattini o underscore, che rappresentano la parte locale dell'indirizzo email.
- @[a-zA-Z0-9-]+ corrisponde al simbolo @ seguito da uno o più caratteri alfanumerici o trattini, che rappresentano il dominio dell'indirizzo email.
- \. corrisponde a un punto letterale. NOTA: "." corrisponde a qualsiasi carattere, tranne il carattere di nuova riga
- [a-zA-Z.]{2,5} corrisponde da due a cinque caratteri che possono essere lettere (maiuscole o minuscole) o punti, che rappresentano l'estensione del dominio dell'indirizzo email.
- $ indica la fine della stringa.

Regular Expressions - quantificatore

Un quantificatore dopo un token o un gruppo specifica la frequenza con cui l'elemento precedente può verificarsi.

```
?      - 0 or 1 match
*      - 0 or more
+      - 1 or more
{n}    - exactly n
{n,}   - n or more
{,n}   - n or less (??)
{n,m} - range n to m
```

Esempio - verifica robustezza della password

Per garantire la robustezza delle password durante la registrazione, è possibile implementare una serie di regole che verifichino se le password soddisfano determinati criteri, come la lunghezza minima, la presenza di lettere maiuscole, minuscole, numeri e caratteri speciali. Ecco come puoi farlo:

- **Lunghezza Minima:** Assicurati che la password abbia una lunghezza minima. Ad esempio, una password deve contenere almeno 8 caratteri.
- **Lettere Maiuscole e Minuscole:** Richiedi la presenza sia di lettere maiuscole che minuscole nella password. Questo rende la password più robusta.
- **Numeri:** Assicurati che la password contenga almeno un numero.
- **Caratteri Speciali:** Richiedi la presenza di almeno un carattere speciale nella password, come "!@#$%^&*()_+[]{}|;:,.<>/?". I caratteri speciali aggiungono ulteriori complessità alla password.

```
// Funzione per verificare la robustezza della password
function isStrongPassword($password) {
    // Controlla la lunghezza minima
    if (strlen($password) < 8) {return false;}
    // Controlla la presenza di lettere maiuscole e minuscole
    if (!preg_match('/[a-z]/', $password) || !preg_match('/[A-Z]/', $password)) {return false;}
    // Controlla la presenza di almeno un numero
    if (!preg_match('/[0-9]/', $password)) {return false;}
    // Controlla la presenza di almeno un carattere speciale
    if (!preg_match('/[!@#$%^&*()_+[\]{}|;:,.<>/?]/', $password)) {return false;}
    return true;
}
```

Esercitazione

Esercizio a

creazione form e validazione html

Esercizio a - validazione dati form

Creare una pagina sito per la gestione dell'anagrafica utenti.

- Creare un form che raccoglie le informazioni dell'utente
  - nome (obbligatorio, solo caratteri alfabetici e spazio)
  - cognome (obbligatorio, solo caratteri alfabetici, spazio e apostrofo)
  - data di nascita (obbligatorio)
  - codice fiscale (opzionale)
  - email (obbligatoria)
  - cellulare (opzionale, numerico di 12 cifre: si memorizza anche il prefisso di 2 cifre. Es un numero in Italia: 391234567890)
  - indirizzo: via/piazza, cap, comune, provincia (obbligatori). N.B. devono essere attributi atomici
  - nickname (deve essere diverso da nome e cognome)
  - password (maiuscola, numero, minimo 8 caratteri, carattere speciale)
- Usare il tipo più adatto per gli elementi del form
  - https://developer.mozilla.org/en-US/docs/Learn/Forms
  - https://developer.mozilla.org/en-US/docs/Web/HTML/Element#forms
- Usare gli attributi degli elementi html per validare i dati inseriti
  - https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes
- Visualizzare i dati inseriti
  - creare una pagina php per la visualizzazione dei dati inseriti nel form

Esercizio b

validazione dati form in php

Esercizio a - validazione dati form

- Verificare che i dati inseriti tramite form siano validi e visualizzarli
  - validare i dati sia lato client che lato server *(vedi slide successiva)*
  - visualizzare in rosso gli eventuali errori

Esercizio a - validazione dati form

- Controlli lato server
  - Il nickname deve essere diverso da nome e cognome
  - per l'email fare il controllo utilizzando la funzione php filter_var, es.:
    ```
    filter_var($_POST["email"], FILTER_VALIDATE_EMAIL)
    ```
  - per ogni campo ignorare gli spazi prima e dopo la stringa utilizzando la funzione trim
  - usare la funzione htmlspecialchars per evitare l'inserimento dei caratteri speciali
- Se i dati inseriti rispettano i criteri di validazione li visualizziamo a schermo

Esercizio a - validazione dati form

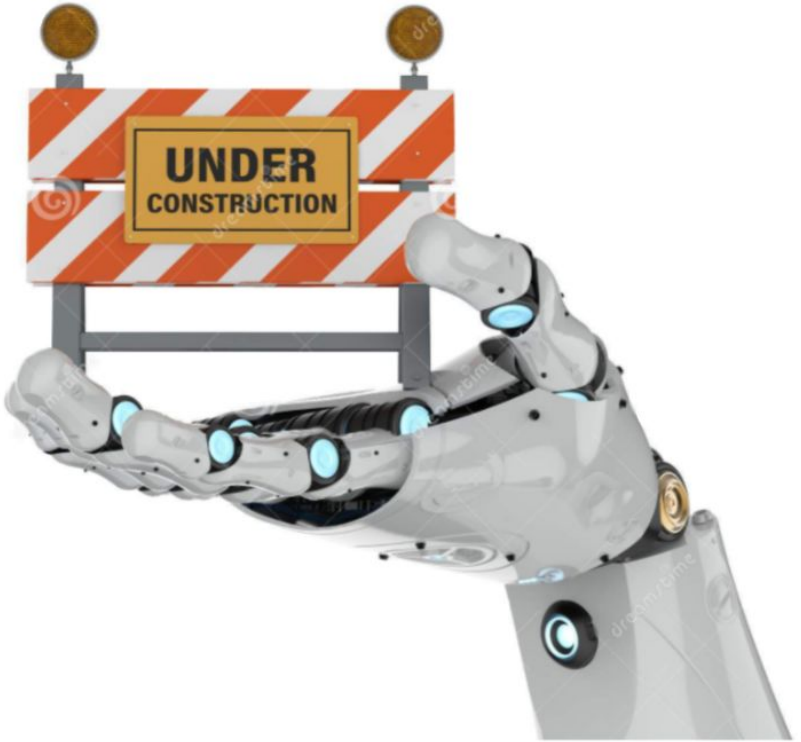
Parti del Documento Tecnico (con Google docs consegnare subito su classroom)

- Titolo
- Obiettivo
  - inserire l'obiettivo della esercitazione
- Introduzione teorica
  - lasciare vuota questa parte
- Progetto
  - descrizione dettagliata e chiara del problema da risolvere, diagrammi, ...
- Implementazione
  - link al codice sorgente
- Collaudo
  - mostrare e spiegare i test effettuati

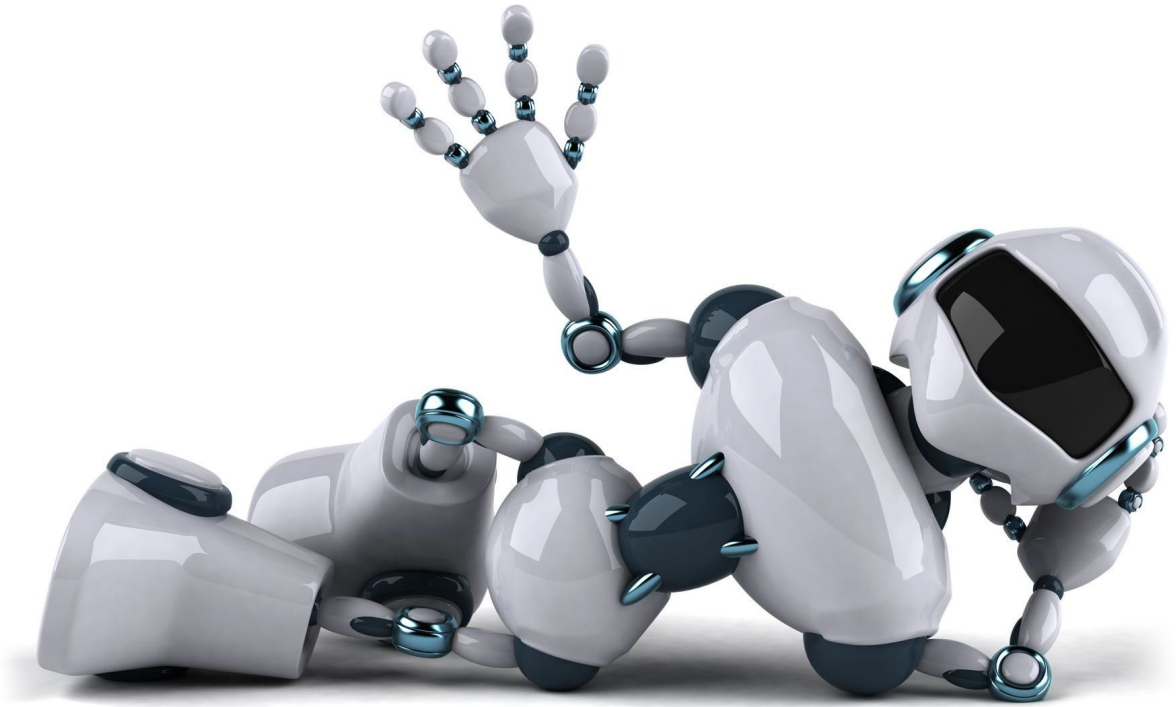
Valutazione

Consegna del lavoro

Si ricorda l'importanza di svolgere e consegnare i progetti assegnati.



Grazie per l'attenzione



Link/Riferimenti

- https://fb-labs.blogspot.com/
- https://github.com/filippo-bilardo
- https://www.youtube.com/channel/UCoBNbHeKNdgeXjMiWhXuH8A/playlists
- Form validation
  - https://www.tutorialspoint.com/php/php_validation_example.htm
  - https://www.javatpoint.com/form-validation-in-php
  - https://www.w3schools.com/php/php_form_required.asp
  - https://www.javatpoint.com/form-validation-in-php
- Espressioni regolari
  - PHP: preg_match - Manual
  - https://www.guru99.com/php-regular-expressions.html
  - https://zetcode.com/php/regex/
  - https://www.geeksforgeeks.org/php-regular-expressions/
  - https://www.tutorialrepublic.com/php-tutorial/php-regular-expressions.php
  - https://www.html.it/pag/60273/le-espressioni-regolari-in-php/
  - https://www.html5pattern.com/
- https://getbootstrap.com/docs/5.2/examples/checkout/

Revisioni

v1.0 13/10/22 - versione iniziale
v1.1 12/10/23 - ampliata l'introduzione teorica
v1.2 09/10/24 -