Progettazione di Servizi Web e Reti di Calcolatori

Politecnico di Torino – Prof. Antonio Lioy

AA 2019-2020, esercitazione di laboratorio n. 9

Per scrivere file HTML/CSS/JS/PHP occorre usare un normale editor di testo (es. Notepad++). Per file HTML si può avere indifferentemente estensione ".htm" o ".html". Per CSS ".css". Per Javascript ".js" e per PHP ".php".

Se le pagine sono di tipo statico e sono scritte correttamente con link relativi, allora è possibile posizionarle in una qualunque cartella del filesystem e navigarle direttamente aprendole con un qualunque browser.

Le pagine PHP di questa esercitazione sono invece di tipo dinamico lato server. Per usarle correttamente bisogna accedervi attraverso un server HTTP integrato con un interprete PHP, e *non* accedere direttamente al corrispondente file su disco locale come fatto nelle esercitazioni precedenti. Un esempio di URL corretta (da inserire come indirizzo della pagina nel browser) sarebbe:

`http://195.231.2.153/lab8/pagina.php` (notare "http" ad inizio link)

mentre un esempio errato sarebbe:

`file:///C;/PWR/lab8/pagina.php` (notare "file" ad inizio link, chiaro indice di **errore!!!**).

Negli esercizi che prevedono l'introduzione di numeri, verificarne il corretto funzionamento non solo introducendo dati validi (es. "5", "-3", "+7", "2.3", "5e3") ma anche agendo volutamente in modo scorretto:

- introducendo dati errati (es. "5mila", "cinquemila", "5 mila", "2,3");
- inviando il form senza aver inserito tutti i dati richiesti.

Si suggerisce di sviluppare ogni esercizio prima in forma base (ossia senza particolari controlli di errore) e quindi facendo le necessarie modifiche per trattare anche il caso di introduzione di dati errati.

Si ricorda che per validare il codice HTML delle pagine contenenti elementi Javascript client-side è necessario usare un validatore installato all'interno del browser. A questo scopo, si consiglia l'uso del plugin "HTML validator" (come nelle precedenti esercitazioni).

Lo stesso concetto e *modus operandi* si applicano anche alla validazione delle pagine PHP, visto che il codice HTML viene generato dall'esecuzione dello script e non è quindi disponibile sul server ma solo sul cliente che riceve il risultato dell'esecuzione dello script.

Per ciascun esercizio, si suggerisce di realizzare l'invio dei dati dei form prima tramite il metodo il metodo GET (particolarmente utile per fare il debug del sito perché visualizza all'interno della URL i parametri passati ed i relativi valori) e poi tramite il metodo il metodo POST (procedura più professionale perché nasconde i dati all'utente e non lascia traccia dei parametri e dei valori nel log del server HTTP).

Per ogni esercizio in cui ciò sia possibile e sensato, verificare lato client, prima dell'invio al server, che i dati inseriti rispettino le specifiche indicate nel testo dell'esercizio (es. siano numeri interi) ed in caso contrario i dati non siano inviati al server ma sia visualizzato un messaggio di errore.

N.B. Questo controllo "lato client" *non* implica la possibilità di non sviluppare gli opportuni controlli anche lato server. Infatti la pagina PHP potrebbe essere acceduta con intenti malevoli, ricevendo quindi valori errati o contraffatti; ciò rende quindi indispensabili i controlli lato server.

Esercizio 9.1

Utilizzando l'esempio sull'uso della variabile $_SERVER visto a lezione, realizzare la pagina PHP "server_info.php" che visualizzi le seguenti informazioni:

> Benvenuto,
> hai richiesto la pagina <URL> usando il metodo <metodo_HTTP>.
> Stai usando il browser <nome_browser> dal nodo con indirizzo IP <IP_client>.
> Sei collegato al server <nome_server> (indirizzo IP <IP_server> e porta <porta_server>/TCP).

Si noti che la <URL> deve essere costruita partendo da diverse variabili dell'array $_SERVER, ad esempio:

> http://localhost/lamiapagina.php

Esercizio 9.2

Realizzare una pagina PHP (con JS per il lato client) che fornisca come risposta l'indicazione della data e dell'ora sia del server sia del client. Ad esempio:

> Benvenuto!
> Qui sul server sono le <ora> del <data>.
> Sul client sono le <ora> del <data>

Esercizio 9.3

Creare una pagina HTML che contenga un form con:

- due campi di testo per l'inserimento di due numeri interi composti al massimo da tre cifre (ad esempio 130);
- un gruppo di pulsanti per scegliere una sola operazione da svolgere tra somma, differenza, prodotto e quoziente;
- un pulsante (Calcola) per inviare le informazioni del form ad una pagina PHP che calcolerà e visualizzerà il risultato dell'operazione nella forma

  <risultato> = <operando1> <operazione> <operando2>

- un pulsante (Cancella) per riportare il form allo stato iniziale.

In caso di ricezione di dati errati, la pagina PHP non deve svolgere l'operazione ma segnalare l'errore rilevato.

Esercizio 9.4

Creare una pagina HTML che contenga un form con:

- un campo di testo che permetta l'introduzione di una temperatura scritta nel seguente formato: il segno + o –, massimo tre cifre per la parte intera, il punto e quindi una sola cifra per la parte frazionaria;
- un pulsante (Invia) per inviare le informazioni del form ad una pagina PHP;
- un pulsante (Cancella) per riportare il form allo stato iniziale.

A seconda della temperatura, la pagina PHP deve visualizzare uno dei seguenti messaggi:

- "Oggi è una giornata molto fredda" (se la temperatura è inferiore a 0);
- "Oggi è una giornata fredda" (se la temperatura è compresa tra 0 e 10);
- "Oggi è una giornata tiepida" (se la temperatura è compresa tra 11 e 18);
- "Oggi è una giornata calda" (se la temperatura è compresa tra 19 e 25);
- "Oggi è una giornata molto calda" (se la temperatura è maggiore di 25).

Esempi di temperature scritte correttamente: +3.2, -273.0, +27.4. Esempi di temperature indicate in modo errato: -273, 27.4, 0. Nel caso che la temperatura introdotta non rispetti questa specifica, la pagina PHP deve visualizzare un messaggio di errore.

Esercizio 9.5

Creare una pagina HTML contenente un form con un campo di input testuale multi-riga (ossia di tipo textarea) e due pulsanti, rispettivamente Cancella e Invia. Realizzare quindi una pagina PHP che riceva i dati dal form e crei una pagina di risposta che contenga l'elenco ed il numero di parole inserite. All'interno del campo testuale è possibile inserire un numero non predefinito di parole, separate dallo spazio, oppure da un carattere di punteggiatura, ad esempio virgola o punto. Si consiglia di usare la funzione "`preg_split()`" (https://www.php.net/manual/en/function.preg-split.php) per identificare i possibili separatori e quindi le parole. Ad esempio, con il seguente testo di input:

elenco: prima parola, seconda parola. Fine elenco

La pagina PHP dovrà restituire:

Numero di parole inserite: 7

Elenco delle parole inserite:

- elenco
- prima
- parola
- seconda
- parola
- Fine
- elenco

Esercizio 9.6

Modificare l'esercizio 9.5 per stampare il numero di occorrenze di ciascuna parola ed il numero di parole univoche. Si consiglia di creare un array associativo dove la chiave rappresenta la parola ed il valore rappresenta il numero di occorrenze trovate nel testo. Ad esempio, con il testo dell'esercizio 9.5, la pagina PHP modificata dovrà restituire:

Numero di parole univoche: 5

"elenco" – numero di occorrenze: 2

"prima" – numero di occorrenze: 1

"parola" – numero di occorrenze: 2

"seconda" – numero di occorrenze: 1

"Fine" – numero di occorrenze: 1

Opzionalmente si sviluppi il programma in modo che non distingua tra caratteri maiuscoli e minuscoli.

Esercizio 9.7

Creare una pagina HTML che contenga un form con cinque campi di input testuali, "nome", "cognome", "username", "password" ed "email", e due pulsanti rispettivamente per ripristinare il form (Cancella) ed inviare i dati al server (Invia). Il form invia i dati ad una pagina PHP che li valida e visualizza le informazioni ricevute. Nel caso in cui la validazione abbia esito negativo è necessario visualizzare un messaggio che descriva il problema, ad esempio: "la password inserita

non rispetta gli standard di sicurezza!”. Nel caso in cui la validazione abbia esito positivo, la pagina visualizzerà il seguente contenuto:

> Caro <nome> <cognome>, abbiamo ricevuto correttamente i tuoi dati.
>
> Lo username scelto è <username>, la tua password contiene <numero caratteri> caratteri e soddisfa i requisiti minimi di sicurezza. Ulteriori informazioni saranno inviati all’indirizzo mail <indirizzoemail>.

Per validare le informazioni ricevute si applichino le seguenti regole:

- “nome” e “cognome” non devono superare i 30 caratteri, devono iniziare con un carattere alfabetico maiuscolo e devono contenere solo lettere o spazi;
- lo “username” deve contenere almeno 6 caratteri e non più di 20 ed i caratteri ammissibili sono quelli alfanumerici, il punto, il più ed il meno;
- la “password” deve contenere almeno 8 caratteri e non più di 16, scelti tra quelli alfanumerici, e deve contenere almeno due cifre, una lettera maiuscola e una minuscola;
- “email” deve essere un indirizzo di posta elettronica valido (deve contenere il carattere @, un nome di dominio ed un top-level domain - TLD). Esempi di indirizzi di posta elettronica validi sono: ilmio.nome1@ilmiodominio.it, ilmionome@posta.ilmiodominio.com.