**Confronti Loose (deboli) e Strict (forti) con PHP
(versione Settembre 2014)**

A differenza di linguaggi come il C++ il PHP offre due famiglie operatori per testare l’uguaglianza e la diversità di espressioni. Non capire la differenza tra loro espone a errori DIFFICILI da scoprire.

Una coppia di operatori è quella che opera il cosiddetto *confronto loose (debole)* e sono quelli normalmente usati: ‘==’ e ‘!=’. Purtroppo sono proprio questi che possono esporci agli errori. Il rischio è generato da due caratteristiche poco conosciute ma che possono avere effetti negativi in situazioni molto comuni:

1. Il cast implicito (cioè una conversione automatica e quindi di cui è facile non rendersi conto) delle stringhe in numeri.

Se una stringa NON inizia con una costante letterale numerica valida allora il valore convertito è zero; diversamente viene effettuata la conversione in numero; ad esempio:

 if (“non ci credo” == 0) echo ‘Eppure è vero!!!’;

mentre a nessun programmatore C++ verrebbe in mente di tentare un simile confronto tra espressioni di tipo diverso il PHP in presenza dell’operatore di uguaglianza debole converte la stringa “non ci credo” nel valore numerico corrispondente. Ora è evidente che questa stringa non contenga nulla (in particolare non inizia per) di numerico. Perciò la conversione automatica ottiene come risultato zero che poi == confronta con 0 concludendo che l’espressione logica (“non ci credo” == 0) è VERA! Provare per credere: l’echo stamperà il messaggio.

Un altro esempio:

 if (“12non ci credo” == 0) echo ‘Eppure è vero!!!’;
 Questa volta la conversione riconosce il valore 12; quindi è come se avessimo scritto:
 if (12==0) echo …; condizione falsa anche per == e non verrà stampato nulla

Invece:
 if (“3.14non ci credo” == 3.14): la parte iniziale viene convertita in 3.14 VERO
 if ("non 3.14 ci credo" == 3.14): la parte iniziale non è un numero: FALSO
 ma if ("non 3.14 ci credo" == 0) VERO
 if (“0” == “000”) VERO
 if (“3” == “3.0”) VERO

Passiamo alla seconda caratteristica …
2. Il PHP considera i seguenti valori tutti equivalenti a FALSE in un contesto di confronto quando si usano gli operatori di confronto debole:

‘’, “”, null, 0, “0”, array senza elementi

Anche in questo caso possono verificarsi situazioni ‘trabocchetto’ come quella del prossimo esempio: la funzione *strpos(stringa\_in\_cui\_cercare, stringa\_cercata)* serve a cercare una stringa all’interno di un’altra; se c’è viene restituita la posizione alla quale è stata trovata; se non c’è viene restituito FALSE (notate come in un linguaggio più rigoroso sui tipi come il C++ non potremmo definire una funzione con due possibili tipi diversi per il risultato!). Consideriamo il seguente codice:

 if ( strpos(‘Minnie e topolino’,’Minnie’) )
 echo "trovato!".$nl;
 else
 echo "non trovato …".$nl;

Nonostante sia evidente che la stringa cercata sia presente nella prima (Minnie è proprio al suo inizio) verrà invece stampato il messaggio ‘non trovato …’. Ecco il perché: come ricorderete la prima posizione di una stringa è 0; quindi strpos che trova ‘Minnie’ all’inizio di ‘Minnie e topolino’ restituisce 0. Ma per quanto detto pochi paragrafi fa in un confronto debole 0 viene considerato uguale a FALSE. Ed il confronto di prima potrebbe essere scritto come:

 if ( strpos(‘Minnie e topolino’,’Minnie’ != FALSE ) )
 echo "trovato!".$nl;
 else
 echo "non trovato …".$nl;

Per l’operatore di confronto debole != come si diceva 0 e FALSE sono equivalenti quindi:
 strpos(‘Minnie e topolino’,’Minnie’) != FALSE
è falso e viene perciò eseguita la parte else.

Tutte queste situazioni si risolvono usando invece i due operatori forti === e !== (quindi con un = in più rispetto ai deboli): questi verificano l’IDENTICITA’ tra le espressioni (stesso tipo e stesso valore senza conversioni):

 if (“3.14non ci credo” === 3.14) FALSO
 if ("non 3.14 ci credo" === 0) FALSO
 if (“0” === “000”) FALSO
 if (“3” === “3.0”) FALSO

Per strpos il controllo corretto per trovato / non trovato diventa invece:
 if ( strpos(‘Minnie e topolino’,’Minnie’ !== FALSE ) )
 echo "trovato!".$nl;
 else
 echo "non trovato …".$nl;

Perciò come regola generale stare molto attenti quando si confrontano stringhe o variabili che potrebbero avere come valore qualche cosa di convertibile in numero ed a funzioni che restituiscono un valore che viene assimilato a FALSO, come lo zero, e che sono usate in una espressione logica.

Per avere tabelle riassuntive di tutti i casi possibili cercate con google ‘PHP loose comparison table’.