I cicli – FOR e WHILE

I cicli sono delle strutture fondamentali del linguaggio e ci servono ad eseguire un codice più volte. Facciamo un esempio solo a parole. Mettiamo che nel nostro database abbiamo un certo numero di indirizzi email e che volessimo inviare un messaggio ad ognuno di questi indirizzi. Schematicamente dovremo procedere come segue:

- Definire una variabile con il messaggio
- Connetterci al database
- Avviare un ciclo che:
    - Legge una riga del database estrapolando un email
    - Invia l'email con destinatario il dato appena preso e come messaggio quanto é contenuto nella variabile definita all'inizio
    - Finché ci sono righe da leggere riprende dal punto a alla riga successiva

Vediamo però ora nel dettaglio i vari cicli di php

FOR

For è il ciclo più semplice e ci permette di eseguire un'operazione un numero noto di volte; la sua struttura è questa:

```
for(partenza; arrivo; incremento)
{
   // codice da eseguire
}
```

Facciamo subito un esempio concreto contiamo fino a 10.

```
for($i = 1; $i <= 10; $i++)
{
   echo "Ho contato fino a $i<br>";
}
```

Come vedete si deve utilizzare un'unica variabile. Devo dire quanto vale $i al primo ciclo (1 nel nostro caso), fino a che valore di $i deve continuare il ciclo, di quanto incrementare $i per ogni ciclo (ricorderete che $i++ equivale a dire $i = $i + 1).

Facciamo un esempio più intelligente, mostriamo il contenuto di un array

```
$frutti = array("mela", "pera", "pesca", "prugna");

 $max = count($frutti);

 for($i = 0; $i < $max; $i++)
```

```
{
   echo $frutti[$i] . ‘<br>’;
}
```

Dopo aver contato il numero di elementi dell’array $frutti, impostiamo un ciclo che parte da 0 (la prima chiave di un’array è 0 ricordate) e si incrementa di 1 fino a che è minore del numero di elementi dell’array.

A dire il vero, esiste un particolare tipo di for che serve proprio a scorrere il contenuto di un’array:

FOREACH
Come detto foreach ci permette di scorrere un’array, in questo modo

```
$frutti = array("mela", "pera", "pesca", "prugna");

 foreach ($frutti as $frutto)
{
   echo $frutto . '<br>';
}
```

Praticamente diciamo: Finché ci sono elementi nell’array $frutti, assegna il valore di ciascun elemento a $frutto.

Foreach può essere utilizzato con una sintassi un po’ più complessa per scorrere un array associativo:

```
$dati = array("nome" => "Maurizio", "cognome" => "Tarchini", "età" =>
"36");

 foreach ($dati as $key => $valore)
{
   echo $key  . ': ' . $valore . '<br>';
}
```

Adesso facciamo quello che abbiamo detto all’inizio, inviare un messaggio email a diverse persone. Se guardate nella documentazione la funzione mail, noterete che passa diversi argomenti che sono nell’ordine: il destinatario, il soggetto, il messaggio, gli header e parametri come opzioni. Lasciamo perdere tutta la questione degli header che è estremamente complessa. Bisogna però sapere che se voglio inviare un mail in formato html, o aggiungere un’allegato dovrò imposare degli header molto precisi. Per evitare queste ed altre complicazioni, è vivamente consigliato utilizzare la classe PHPMailer per l’invio di email (la vedremo nell’ultimo capitolo del corso). Comunque per inviare un semplicissimo mail testuale, andrà benissimo

anche la funzione mail (vi ricordo che per funzionare, il php.ini dovrà essere correttamente configurato come descritto nel capitolo relativo alla preparazione dell'ambiente di svilupo).

Adesso cominciamo, e cominciamo finalmente a fare qualcosa di divertente.

```
// Creiamo un array associativo con il nome dei nostri amici ed il loro email.

$rubrica = array("Maurizio" => "maurizio@sito.com",
                 "Giovanni" => "giovanni@sito.com",
                 "Marco" => "marco@sito.com");

$oggetto = "prova invio email";

foreach ($rubrica as $nome => $email)
{
    $messaggio = "Ciao $nome,\r\n ti scrivo questo messaggio come prova di un programmino che sto realizzando";

    mail($email, $oggetto, $messaggio);

}
```

Ogni volta che il nostro ciclo verrà eseguito, viene creato un messaggio personalizzato nel senso che il nome sarà il nome abbinato all'email e questo messaggio verrà poi inviato al destinatario.

WHILE

Abbiamo visto che for ci permette di eseguire un numero predefinito di cicli. Ma ci sono diverse situazioni nelle quali ci è impossibile sapere a priori quanti cicli saranno necessari per raggiungere un determinato obiettivo. In questo ci viene incontro la struttura while, la quale continua ad eseguire il ciclo fin tanto che non si verifica la situazione FALSE. Dunque while analizza il valore boleano di un'espressione.

Vediamo

```
while(fino a quando questa espressione è TRUE, continua ad eseguire il seguente codice)
{
  //codice
}
```

In concreto

```php
$inc = 1;

 while($inc <= 10)
{
        echo "Ho contato fino a $inc<br>";
        $inc++;
}
```

Questo script fa esattamente quello che faceva lo script che abbiamo utilizzato come esempio per la struttura for ma in modo concettualmente diverso.
Facciamo il solito esempio stupido: Vogliamo vedere quanti tentativi ci vorranno prima che la somma di tre numeri casuali tra 1 e 10 sia 23. Ovviamente non sappiamo quanti tentativi ci vorranno (e ogni volta che eseguiremo lo script il numero sarà diverso); quindi dovremo affidarci a while.

```php
$tentativi = 1;

 while($result != 23) // fino a che il risultato é diverso da 23 l'espressione è TRUE
{
        $a = rand(1,10);  // prendiamo tre numeri casuali
        $b = rand(1,10);
        $c = rand(1,10);
        $result = $a + $b + $c; // Li sommiamo

        $tentativi++;  // E incrementiamo di 1 il numero di tentativi
}

 // Quando $result è 23, quindi 23 != 23 è FALSE, il ciclo finisce
// Quindi stampiamo i risultati

 echo "Ci sono voluti $tentativi tentativi per avere come risultato $result<br>";
echo "I tre numeri sono: $a, $b, $c.";
```