Γεωργίου Γκίρτσου

Η Γλώσσα Προγραμματισμού PHP

(Μέρος 1ο)

Δράμα, Οκτώβριος 2007

Το Συντακτικό της PHP

Η PHP είναι μια γλώσσα που δημιουργήθηκε για να μπορεί να εισάγεται μέσα στην HTML εύκολα (αν και δεν χρειάζεται να το κάνετε με αυτό τον τρόπο). Οι περισσότερες PHP σελίδες έχουν PHP & HTML κώδικα μαζί (intermixed).

Όταν ο server "διαβάζει" μια PHP σελίδα ψάχνει για δύο πράγματα που θα του επιτρέψουν να δει από πού θα αρχίσει να διαβάζει την σελίδα ως PHP και πότε ως HTML. Διαβάζει την σελίδα ως PHP όταν συναντάει τα tags <?php και ?>.

Αν έχετε ρυθμίσει το php.ini αρχείο σας κατάλληλα ώστε να μπορείτε να χρησιμοποιήσετε τα short-tags της PHP (τα οποία είναι ενεργοποιημένα από default) τότε θα μπορείτε να δηλώνετε στον server από πού αρχίζει ένα PHP block με τα ακόλουθα tags : <? και ?>.

Επίσης, μπορείτε να ρυθμίσετε τον server σας να αναγνωρίζει τα ASP tags (<% και %>) ως PHP tags. Αυτό είναι απενεργοποιημένο από default και ο πραγματικός σκοπός του φαίνεται ότι είναι να επιτρέπει στους editors να διαβάζουν το block του κώδικα όχι σαν HTML (ώστε να τον χρωματίζει κατάλληλα).

Ένα πολύ απλό παράδειγμα που δείχνει την εισαγωγή της PHP μέσα στην HTML είναι το παρακάτω :

```
<h1>Καλώς Ήρθατε στην ιστοσελίδα της
        <?php echo $company; ?>!
</h1>
```

Όταν ο παραπάνω κώδικας εκτελεστεί, πολύ απλά θα εμφανίσει το όνομα της εταιρίας στην θέση του PHP block (φυσικά θα πρέπει να έχουμε ορίσει τη μεταβλητή $company).

Γενικά, οι γραμμές της PHP πρέπει να τελειώνουν με ένα ελληνικό ερωτηματικό (semicolon) αν και δεν χρειάζεται να χρησιμοποιούμε ερωτηματικό όταν βάζουμε αγκύλες (θα το καταλάβουμε καλύτερα όταν κοιτάξουμε τα if/else κλπ statements).

Για παράδειγμα :

```
<?php
        echo "<p>Μία γραμμή κώδικα</p>";
        echo "<p>Μια άλλη γραμμή κώδικα</p>";
?>
```

Οι Μεταβλητές (Variables)

Μπορείτε επίσης να συνδυάσετε μικρά τμήματα της PHP στην HTML, όπως αυτό στην εκτύπωση της ημερομηνίας :

```
<h1>Σήμερα είναι <?= $date; ?>.</h1>
```

Η σύνταξη "<?=" στην PHP όταν ακολουθείται από μεταβλητή, είναι η συντομογραφία της συνάρτησης echo().

Μπορείτε να εισάγετε σχόλια στον κώδικά σας. Τα σχόλια αγνοούνται από τον server και όποια σχόλια υπάρχουν δεν στέλνονται στον browser του χρήστη. Ακολουθούν τρεις τρόποι με τους οποίους μπορούμε να εισάγουμε σχόλια :

 # -> Ακριβώς όπως στην PERL, ορίζει ως σχόλιο ό,τι ακολουθεί στην ίδια γραμμή

// -> Ακριβώς όπως στην Javascript, ορίζει ως σχόλιο ό,τι ακολουθεί στην ίδια γραμμή

/ */ -> Ορίζει ως σχόλιο ο,τιδήποτε υπάρχει ανάμεσα σ' αυτά τα σύμβολα. Είναι ακριβώς η ίδια σύνταξη που χρησιμοποιούμε στην C.*

Παραδείγματα κώδικα με τη χρήση σχολίων :

```
<?php
        echo "Γεια σας!"; # εμφανίζει το μήνυμα "Γεια σας!"
        echo "Γεια σας!"; // εμφανίζει το μήνυμα "Γεια σας!"
        /*
                Εμφανίζει το μήνυμα "Γεια σας!"
        */
        echo "Γεια σας!";
?>
```

Τα περισσότερα σχόλια που θα δείτε ορίζονται ως // και /* */. Το σχόλιο που συμβολίζεται με την δίεση (#) χρησιμοποιείται σπάνια, αν και είναι έγκυρο.

Οι μεταβλητές στην PHP ορίζονται με το σύμβολο του δολαρίου ($). Για να ορίσουμε ως τιμή "Γεια σας!" στη μεταβλητή $a πολύ απλά γράφουμε :

```
$a = "Γεια σας!";
```

Τα strings (σειρά χαρακτήρων) πρέπει να εσωκλείονται από εισαγωγικά, αλλά μπορούν να περιέχουν και μεταβλητές.

```
$a = 4;
$string = "Η τιμή του a είναι $a";
// $string = "Η τιμή του a είναι 4";
```

Στην PHP οι μεταβλητές δεν χρειάζεται να δηλωθούν στην αρχή του script και ούτε χρειάζεται να ορίσουμε τύπο δεδομένων (type definition). Σημειώστε ότι μπορεί να εμφανιστεί προειδοποίηση (εξαρτάται από την τιμή του error_reporting στο php.ini) αν προσπαθήσετε να εμφανίσετε μια μεταβλητή που δεν έχει δηλωθεί.

Για παράδειγμα :

$a = 4;
$c = $a + $b;
// $c = 4
αλλά ένα μία προειδοποίηση εμφανίζεται "Warning: Undefined variable.."

Οι προειδοποιήσεις δεν εμποδίζουν το σφάλμα απ' το να συνεχίσει να εκτελείται. Αν όμως ξεχάσετε να βάλετε ένα semicolon στο τέλος μιας από τις γραμμές, τότε θα πάρουμε Σφάλμα Parser και δεν θα μπορέσει να συνεχιστεί η εκτέλεση.

Δεδομένου ότι οι μεταβλητές στην PHP δεν πληκτρολογούνται, δεν χρειάζεται να ανησυχείτε για την εκτέλεση μαθηματικών πράξεων/ εξισώσεων σε περίπτωση που δώσετε λάθος τύπο, όπως π.χ. στην C.

$a = 4;
$b = "5";
$c = $a + $b;
// $c = 9;

Επίσης η PHP υποστηρίζει boolean μεταβλητές στις οποίες δίνεται η τιμή είτε 1 είτε 0 (1 = true, 0 = false) :

$a = true;
$b = 1;
// $a = $b
$c = false;
$d = 0;
// $c = $d

Οι Συντελεστές της PHP

Η PHP υποστηρίζει τους ακόλουθους τελεστές (operators) :

Αριθμητικούς Operators

Η PHP υποστηρίζει τους standard μαθηματικούς τελεστές :

$a = 4;
$b = 2;
// Πρόσθεση : $a + b = 6
// Αφαίρεση : $a - $b = 2
*// Πολλαπλασιασμός : $a * $b = 8*
// Διαίρεση : $a / $b = 2
// Τοις εκατό ($a / $b) : $a % $b = 0
// Σταδιακή Αύξηση : $a++ (θα είναι ίσο του 5 αφού $a = 4)

Assigment Operators (Τελεστές Ανάθεσης)

Οι πιο σημαντικοί assigment operators στην PHP είναι το "=" και η τελεία ".". Το ίσον είναι προφανές, ορίζει μια τιμή σε μια μεταβλητή :

$a = 4;
$b = $a;
// b = 4

Οι Τελεστές Σύγκρισης

Η PHP υποστηρίζει τους ακόλουθους τελεστές :

$a == $b // ελέγχει αν οι τιμές τους είναι ίσες
$a != $b // ελέγχει αν οι τιμές τους δεν είναι ίσες
$a < $b // ελέγχει αν η πρώτη τιμή (του $a) είναι μικρότερη της δεύτερης (του $b)
$a > $b // ελέγχει αν η πρώτη τιμή (του $a) είναι μεγαλύτερη της δεύτερης (του $b)
$a <= $b // ελέγχει αν η πρώτη τιμή (του $a) είναι μικρότερη ή ίση της δεύτερης (του $b)
$a >= $b // ελέγχει αν η πρώτη τιμή (του $a) είναι μεγαλύτερη ή ίση της δεύτερης (του $b)

Η PHP υποστηρίζει και τους τελεστές σταδιακής αύξησης/μείωσης :

$a = 5;
$a++;
// $a = 6
$b = 5;
$b--;
//$b = 4

Οι Τελεστές Σύνδεσης

Η τελεία "." ενώνει δύο τιμές :

```
$sentence_a = "Η γρήγορη καφέ ";
$sentence_b = "αλεπού έκανε άλμα...";
$sentence_c = $a . $b;
//$sentence_c = "Η γρήγορη καφέ αλεπού έκανε άλμα...";
```

Οι Πίνακες (Arrays)

Η PHP υποστηρίζει και αριθμητικούς πίνακες (numerical arrays) (στοιχεία ενός πίνακα αποθηκευμένα σύμφωνα με την αριθμητική τους σειρά) όπως και associative arrays (στοιχεία ενός πίνακα αποθηκευμένα σύμφωνα με αλφαβητική σειρά).

```
$a = array(1, 2, 3, 4);
//$a[0] = 1
//$a[1] = 2
//$a[2] = 3
//$a[3] = 4
$b = array("name"=>"Fred", "age" => 30);
//$b['name'] = "Fred"
//$b['age'] = 30
```

Οι Εντολές If/Then

Ένα από τα στοιχεία της PHP είναι τα if/then statements. Τα if/then statements σάς επιτρέπουν να ελέγξετε αν μια έκφραση είναι αληθής και μετά, αν ναι ή όχι, να κάνετε την κατάλληλη ενέργεια.

Για παράδειγμα :

```
$a = 1;
if ($a) {
        echo "Αληθής!";
}
```

Αφού $a = 1, τότε η PHP καταλαβαίνει ότι υπάρχει τιμή στην $a (οπότε έχει οριστεί, δηλ. η έκφραση είναι αληθής) και εμφανίζει το μήνυμα. Αν ήταν να το διαβάσουμε με λόγια θα λέγαμε : "*Αν είναι αληθής, τότε εμφάνισε "Αληθής!"*".

Ένα άλλο παράδειγμα, αλλά αυτή τη φορά με το else :

```
$a = 5;
$b = "10";
if ($a > $b) {
        echo "Το $a είναι μεγαλύτερο από το $b";
} else {
        echo "Το $a δεν είναι μεγαλύτερο από το $b";
}
```

Η PHP δε νοιάζεται αν το $a είναι ακέραιος (integer) και αν το $b είναι string. Αναγνωρίζει ότι το $b μπορεί να δουλέψει και ως ακέραιος. Μετά αν η έκφραση είναι αληθής "*Αν το $a είναι μεγαλύτερο του $b τότε εμφάνισε ότι το $a είναι μεγαλύτερο του $b, αλλιώς (else) εμφάνισε ότι το $a δεν είναι μεγαλύτερο του $b*".

Είναι επίσης σημαντικό να σημειώσουμε ότι δεν λέμε "*το $a δεν είναι μικρότερο από το $b*" μιας και το $a μπορεί να είναι ίσο του $b. Αν είναι έτσι, τότε το δεύτερο μέρος της έκφρασης, το else statement, είναι το μέρος όπου θα εκτελεστεί.

Προσέξτε τη χρήση των αγκυλών που ορίζουν ποιες ενέργειες θα γίνουν και πότε. Επίσης προσέξτε ότι δεν βάζουμε semicolons όταν έχουμε αγκύλες.

Ένα τελικό παράδειγμα με if/elseif/else :

```
if ($a == $b) {
        // κάνε κάτι
} elseif ($a > $b) {
        // κάνε κάτι άλλο
} else {
        // αν κανένα από τα παραπάνω, τότε κάνε αυτό...
}
```

Οι Εντολές Switch

Στη γλώσσα προγραμματισμού C, το switch statement προέκυψε από την ανάγκη για κάτι παραπάνω στα if/then όταν κάνουμε υπολογισμούς. Με το if/then statement "κλειδώνεστε" σε μία απλή έκφραση. Τα switch statements σάς δίνουν πρόσθετους ελέγχους των δεδομένων ακόμη και αν κάποια έκφραση έχει συναντηθεί.

Επίσης τα switch statements μπορούν να σας "σώσουν" από την πληκτρολόγηση πολλών if/elseif/elseif... statements.

Η PHP ακολουθεί τη σύνταξη της C για το switch statement :

```
$a = "100";
switch ($a) {
        case(10):
                echo "Η τιμή είναι 10";
                break;
        case (100):
                echo "Η τιμή είναι 100<br />";
                break;
        case (1000):
                echo "Η τιμή είναι 1000";
                break;
        default:
                echo "<p>Έχετε ορίσει αριθμό;</p>";
}
```

Όπως μπορείτε να δείτε, έχουν 4 βασικά μέρη :

1. Το switch αναγνωρίζει ποια τιμή θα επεξεργαστεί ή έκφραση σε κάθε ξεχωριστή περίπτωση (case). Στο παραπάνω παράδειγμα, λέμε στο switch statement μας ότι θα επεξεργαστούμε τη μεταβλητή $a.
2. Το case statement αξιολογεί τη μεταβλητή που περάσατε από την εντολή switch. Μπορείτε να χρησιμοποιήσετε case() ακριβώς με τον ίδιο τρόπο που θα χρησιμοποιούσατε ένα if statement. Αν είναι true (αληθές), τότε ότι ακολουθεί μετά το case, εκτελείται μέχρι ο parser να βρει μία εντολή break όπου εκεί σταματάει να εκτελεί τις ενέργειες του case.
3. Τα "breakpoints" ορίζονται από την εντολή break και κάνει έξοδο ο parser από το switch.
4. Το default είναι μια ειδική περίπτωση. Εκτελείται αν καμία από τις παραπάνω περιπτώσεις δεν έχει εκτελεσθεί.

Οι Βρόγχοι For (For Loops)

Οι βρόγχοι είναι χρήσιμοι επειδή μπορούν να επεξεργαστούν δεδομένα που βρίσκονται μέσα σ' ένα array.

Παράδειγμα :

```
for ($i = 0; $i < sizeof($array); $i++) {
        // κάνε κάτι με το array[$i];
}
```

Για τα loops τρία πράγματα χρειάζεται να οριστούν για να χρησιμοποιηθούν :

1. Counter (=μετρητής). Στο παραπάνω παράδειγμα, $i = 0. Μπορείτε να περάσετε σε μια ήδη ορισμένη μεταβλητή ή να ορίσετε μια τιμή στη μεταβλητή για τη δήλωση.
2. Ο όρος (συνθήκη) που απαιτείται για να συνεχιστεί ο βρόγχος. Στο παραπάνω παράδειγμα, το $i είναι μικρότερο από το μέγεθος του $array, οπότε ο βρόγχος for συνεχίζει να εκτελείται.
3. Δήλωση για να τροποποιήσει τον μετρητή σε κάθε πέρασμα του βρόγχου. Στο παραπάνω παράδειγμα το $i++ αυξάνει τον μετρητή σε κάθε πέρασμα.

Τα Foreach Loops

Τα foreach loops δίνουν τη δυνατότητα να περάσουμε (διασχίσουμε) από κάποια στοιχεία που είναι αποθηκευμένα σ' ένα array :

```
$array = array("name" => "Jet", "occupation" => "Bounty Hunter" );
foreach ($array as $val) {
        echo "<P>$val";
}
```

Εμφανίζει τα εξής :

```
<P>Jet
<P>Bounty Hunter
```

Μπορείτε επίσης να χρησιμοποιήσετε το foreach loop για να πάρετε το key (=κλειδί) των τιμών του array (π.χ. σε ποια θέση βρίσκονται) :

```
foreach ($array as $key => $val) {
        echo "<P>$key : $val";
}
```

Εμφανίζει τα εξής :

```
<P>name : Jet
<P>occupation : Bounty Hunter
```

Τα While Loops

Τα while loops είναι άλλος ένας τρόπος που μπορούμε να κάνουμε loop σε κάποια δεδομένα. Μία συχνή χρήση του while είναι όταν παίρνουμε δεδομένα από έναν πίνακα μιας βάσης δεδομένων.

```
while ($row = mysql_fetch_array($result)) {
        // κάνε κάτι με την resulting row
}
```

Μπορείτε επίσης να χρησιμοποιήσετε την εντολή *continue* για να αποφύγετε την τρέχουσα επανάληψη του βρόγχου, κατόπιν συνεχίζετε :

```
while ($row = mysql_fetch_array($result)) {
        if ($row['name'] ! = "Jet")) {
                continue;
        } else {
                // κάνε κάτι με το row
}
```

Τα Do While Loops

Τα Do While loops δουλεύουν ακριβώς όπως τα while loops με τη μόνη διαφορά ότι αξιολογεί την έκφραση στην ολοκλήρωση του loop, και όχι κατά την εκτέλεση του while loop όπως θα γινόταν σ' ένα κανονικό while loop.

Μια σημαντική διαφορά των Do While Loops με τα while loops είναι ότι τα Do While Loops εκτελούνται τουλάχιστον μία φορά. Ένα κανονικό while loop μπορεί να μην εκτελεστεί καθόλου (φυσικά εξαρτάται από την έκφραση). Το ακόλουθο Do While Loop παράδειγμα εκτελείται μία φορά εμφανίζοντας το $i :

```
$i = 0;
do {
        print $i;
} while ($i>0);
```

Ενώ το παρακάτω δεν εκτελείται καθόλου :

```
$i = 0;
while ($i > 0) {
        print $i;
}
```

Ειλικρινά δεν βρίσκω λόγο στη χρήση των Do While Loops, εκτός κι αν συναντήσετε κάποιο περίεργο πρόβλημα που το απαιτεί και να χρειάζεται να το χρησιμοποιήσετε.

Οι Συναρτήσεις του Χρήστη (User Functions)

Εκτός από τα built-in functions που μας παρέχει η PHP ο προγραμματιστής μπορεί να δημιουργήσει την δική του function για να κάνει κάποια συγκεκριμένη ενέργεια.

Προσέξτε ότι αν θέλετε να χρησιμοποιήσετε μέσα στην function κάποια μεταβλητή που δεν έχει δηλωθεί μέσα στην function, τότε πρέπει να την δηλώσετε ως global.

```
function check_age($age) {
        if ($age > 21) {
                return 1;
        } else {
                return 0;
        }
}

if (check_age($age)) {
        echo "Γεια σας!";
} else {
        echo "Απαγορευμένη πρόσβαση!";
        exit;
}
```

Η function (=συνάρτηση) θα εκτελεστεί από το script την κατάλληλη στιγμή με το ακόλουθο συντακτικό :

```
$age = 10;
check_age($age);
// εμφανίζει  το μήνυμα "Απαγορευμένη πρόσβαση!"
```

Ο Αντικειμενοστραφής Προγραμματισμός

Η PHP υποστηρίζει αντικειμενοστραφή προγραμματισμό (Object Oriented Programming) με τη χρήση των classes. Όπως και οι άλλες OOP γλώσσες προγραμματισμού, τα classes μπορούν να επεκταθούν για μεγαλύτερη επαναχρησιμοποίηση του κώδικα.

Για να δημιουργήσουμε ένα address entry class που περιέχει το όνομα ενός ανθρώπου και το τηλέφωνό του :

```
class address_book_entry {
        var $first;
        var $last;
        var $number;
        function set_name($first, $last) {
                $this->first = $first;
                $this->last = $last;
        }
        function set_number($number) {
                $this->number = $number;
        }
        function display_entry() {
                echo "<p>Όνομα : " . $this->first . " " . $this->last;
                echo "<br>Αριθμός : " . $this->number;
        }
}
```

```
// Χρήση :
$entry = &new address_book_entry;
$entry->set_name("Jane","Smith");
$entry->set_number("555-555-5555");
$entry->display_entry();
// εμφανίζει :
Όνομα : Jane Smith
Αριθμός : 555-555-5555
```

Επίσης μπορείτε να επεκτείνετε ένα υπάρχον class για να δημιουργήσετε ένα class που έχει την ίδια λειτουργία μ' ένα παλιό class, συν κάθε νέα δυνατότητα που προσθέτετε :

```
class address_book_entry2 extends address_book_entry {
	var $email;
	function set_email($email) {
		$this->email = $email;
	}
	function display_entry2() {
		echo "<p>Όνομα : " . $this->first . " " . $this->last;
		echo "<br>Αριθμός : " . $this->number;
		echo "<br>Email : " . $this->email;
	}
}

// Χρήση :
$entry = &new address_book_entry2;
$entry->set_name("Jane","Smith");
$entry->set_number("555-555-5555");
$entry->set_email("jsmith@com.com");
$entry->display_entry();
// εμφανίζει :
Όνομα : Jane Smith
Αριθμός : 555-555-5555
Email : smith@com.com
```

Η Συνάρτηση phpinfo()

Η συνάρτηση (function) phpinfo() είναι πολύ χρήσιμη γιατί σας επιτρέπει να δείτε την έκδοση της PHP που είναι εγκατεστημένη στον server σας όπως και άλλες πληροφορίες σχετικά με την εγκατάσταση της PHP σας (όπως το τι ρυθμίσεις είναι ενεργοποιημένες).

Ένας τρόπος για να εμφανίσετε αυτές τις πληροφορίες είναι να δημιουργήσετε ένα script σαν το ακόλουθο :

```
<?php
	phpinfo();
?>
```

Κατά την εκτέλεση του script, ο browser εμφανίζει τις ρυθμίσεις της PHP. Αυτή η function είναι ιδιαίτερα χρήσιμη όταν δουλεύετε στον server κάποιου άλλου και δεν είστε βέβαιος για τις δυνατότητες του server ή τις ρυθμίσεις.

ΠΕΡΙΕΧΟΜΕΝΑ