# Server Side web programming

2010

Οικονομικό Πανεπιστήμιο Αθηνών

# Η δημιουργία ενός προγράμματος που εκτελείται στον Web Server



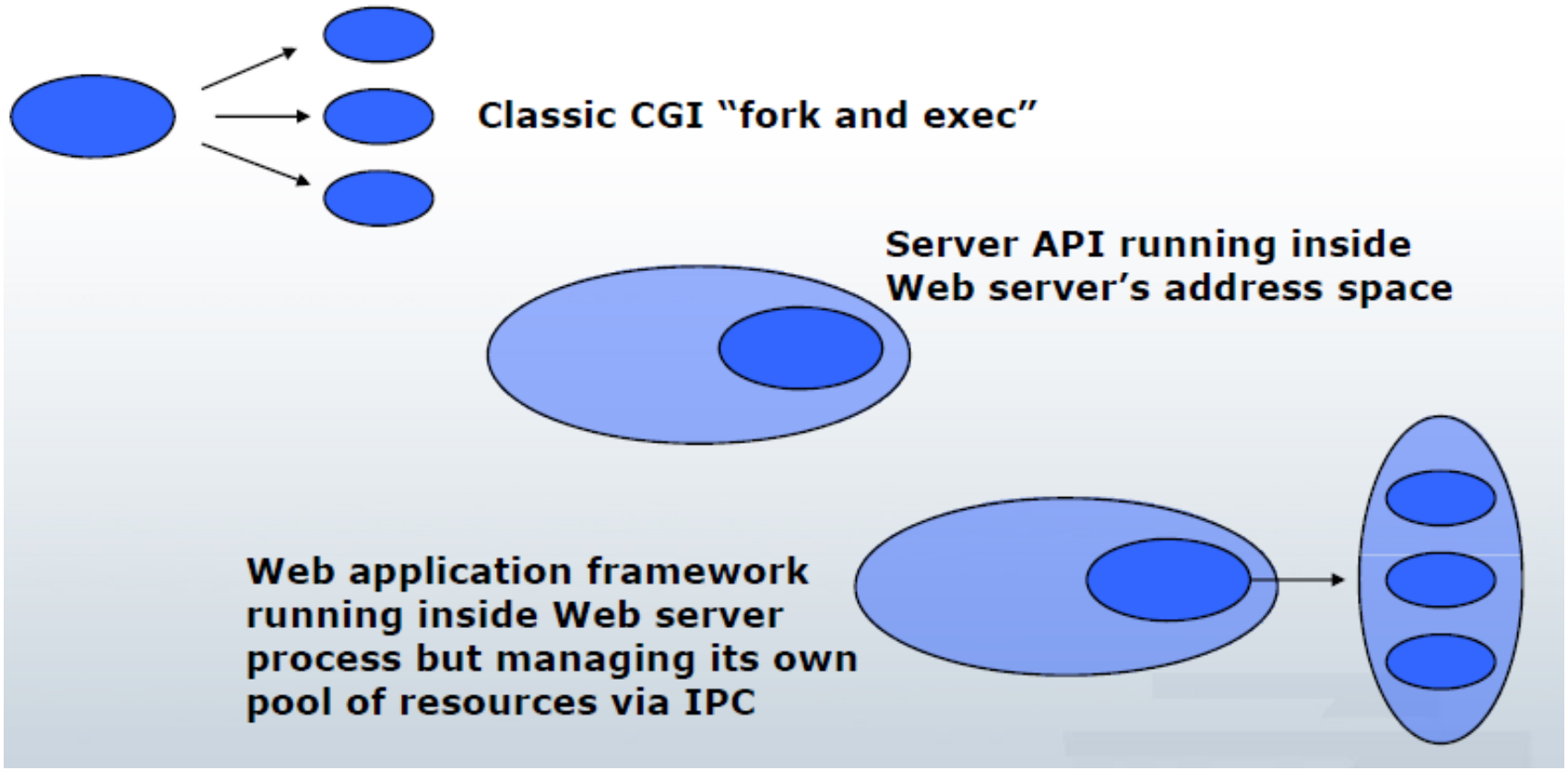
- Αρχικά περιλαμβάνει την ανάκτηση δεδομένων (από τον user-agent στον εξυπηρετητή)
- Έπειτα επεξεργασία τους, επιτέλεση κάποιας εργασίας κ.ο.κ
- Επιστροφή δεδομένων (από τον εξυπηρετητή στον user-agent)
  - Είσοδος: Δεδομένα από φόρμες, URL, HTTP headers (τύπος φυλλομετρητή, IP, cookie, κτλ)
  - Έξοδος: Δεδομένα συνήθως HTTP, GIF, JPEG κτλ και το αντίστοιχο header (τύπος MIME, cookie, κτλ)
- Πρέπει να αντιμετωπιστεί η stateless φύση του HTTP για να δημιουργηθεί ένα ενδιαφέρον πρόγραμμα
  - Δημιουργία συνόδων (session) με χρήση cookie, URLs, κρυμμένα πεδία κτλ
- Θα πρέπει οι επιλογές κατά την υλοποίηση εκτός από τον εύκολο προγραμματισμό να παρέχουν διασύνδεση με ΒΔ, HTML έξοδο, Web Site widgets, κτλ

# Μοντέλα Server-Side Προγραμματισμού



- Υπάρχουν 3 μοντέλα Server-Side προγραμμα-τισμού
- Μοντέλο CGI – "fork and exec"
  - Ο εξυπηρετητής Web δημιουργεί ένα child process και του «περνάει» τα δεδομένα από το request σαν μεταβλητές περιβάλλοντος (environment vars)
  - Τα CGI scripts απαντούν χρησιμοποιώντας μηχανισμούς I/O streams
- Μοντέλο Server API
  - Ο εξυπηρετητής Web εκτελεί κώδικα, μέσα στο process space του, που χειρίζεται αιτήματα (request)
- Web application frameworks
  - Ο εξυπηρετητής Web καλεί μια εφαρμογή, η οποία διαχειρίζεται τα αιτήματα χρησιμοποιώντας native αντικείμενα που καταναλώνουν τους δικούς της πόρους (resources)

# Μοντέλα Server-Side Προγραμματισμού



# Μοντέλα Server-Side Προγραμματισμού



- Το κάθε μοντέλο έχει τα πλεονεκτήματα και τα μειονεκτήματα του
  - Μοντέλο CGI
    - Πλεονεκτήματα: Η απομόνωση σημαίνει ότι είναι εύκολη η προστασία, λιγότερο καταστρεπτικές συνέπειες όταν κάτι πάει στραβά
    - Μειονεκτήματα: Η απομόνωση απαιτεί πόρους και προκαλεί καθυστερήσεις
  - Μοντέλο Server API
    - Πλεονεκτήματα: μεγάλη ταχύτητα και μικρή επιβάρυνση αν η υλοποίηση είναι σωστή
    - Μειονεκτήματα: δυσκολία στην υλοποίηση, ο εξυπηρετητής καταρρέει αν κάτι πάει στραβά
  - Web application frameworks
    - Πλεονεκτήματα: συνδυάζει την αποδοτικότητα του δεύτερου μοντέλου και την ασφάλεια του πρώτου και επιπλέον παρέχει χρήσιμες δραστηριότητες που απαιτούνται συχνά, όπως state management
    - Μειονεκτήματα: Τα παρεχόμενα εργαλεία μπορεί να προκαλέσουν κακή χρήση των πόρων εξαιτίας της απροσεξίας αυτών που τα χρησιμοποιούν

# Μοντέλα Server-Side Προγραμματισμού



- Πολλά παραδείγματα χρήσης για κάθε μοντέλο
  - CGI
    - Scripts γραμμένα σε Perl
    - Προγράμματα γραμμένα σε C
  - Server API
    - Apache modules
    - ISAPI filters και extensions
  - Web application frameworks
    - Όλα προέρχονται από τα Server Side Includes (SSI), μια "parsed HTML" λύση που επιτρέπει την διερμηνεία εκτελέσιμου κώδικα μέσα σε markup
    - ASP, ASP.NET, Cold Fusion, JSP/Servlets, Python, PHP, κτλ

# Θεωρητικές αντισταθμίσεις (trade-offs)



- Όπως έχει αναφερθεί υπάρχει διαφορετικοί τρόποι προγραμματισμού στο Web
  - Κανένας όμως δεν επαρκεί για όλες τις περιπτώσεις
    - Ταχύτητα του ISAPI vs απλότητα ενός PHP script
  - Κάποιοι παρέχουν πρόσβαση σε 'low-level' πληροφορία όπως headers, query strings, κτλ
  - Κάποιοι είναι πιο δύσκολοι στον προγραμματισμό
    - Trade-off: Απλότητα vs Παραμετροποίηση (Configurability)
  - Κάποια high level frameworks παρέχουν ευκολία στον προγραμματισμό θυσιάζοντας την ταχύτητα και την επεκτασιμότητα
  - Κάποιοι είναι μεταφέρσιμοι (portability) ενώ κάποιοι άλλοι συνδέονται με κάποια πλατφόρμα
- Ξεκινάμε με low level CGI για να φανούν οι ομοιότητες στον Server –Side προγραμματισμό ιστού

# CGI (Common Gateway Interface)



- Απλό πρότυπο που ορίζει τον τρόπο εκτέλεσης εξωτερικών προγραμμάτων σε ένα εξυπηρετητή Web
- Χρήσιμοι σύνδεσμοι για CGI
  - http://www.cgi-resources.com
  - http://www.w3.org/CGI/
  - O'Reilly's CGI Book - http://www.oreilly.com/catalog/cgi2/

# CGI & Perl



- Συνήθως με την χρήση CGI κάποιοι εννοούν την χρήση της Perl,
- Το πρότυπο του CGI δεν ορίζει όμως την χρήση κάποιας γλώσσας
- Γιατί όμως συμβαίνει αυτή η συσχέτιση;
  - Η Perl βρίσκεται συνήθως σε UNIX μηχανήματα που ήταν οι πρώτοι εξυπηρετητές Web
  - Η Perl παρέχει πολύ καλή επεξεργασία συμβολοσειρών, που είναι ένα από σημαντικότερα προβλήματα (task) που πρέπει να λυθούν σε ένα πρόγραμμα CGI
  - Η Perl μπορεί να χρησιμοποιηθεί για να δημιουργηθούν ενδιαφέροντα προγράμματα
  - Το γεγονός ότι είναι διερμηνεύσιμη γλώσσα εξηγεί τα προβλήματα ταχύτητας των προγραμμάτων CGI
    - Λύση: Χρήση της mod_perl
    - Βέλτιστη Λύση: Μεταγλώττιση σαν ένα πρόγραμμα σε C

# Πρώτο παράδειγμα σε Perl



```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "<html><head><title>Hello!</title>";
print "</head><body bgcolor='yellow'>\n";
print "<h1>Hello from CGI</h1>\n";
print "</body>\n</html>";
```

- Παρατηρήστε το Content-type: header. Αυτό είναι το 50% της "μαγείας" της Perl
- Σε αυτή τη περίπτωση η κατάληξη του αρχείου θα είναι .pl, αλλά μπορείτε να χρησιμοποιήσετε .cgi ή ακόμη καλύτερα κάποια άλλη δική σας κατάληξη ή καμία
- Συνηθίζεται να μπαίνει ο κώδικας σε ένα κατάλογο με όνομα cgi-bin
  - Μπορεί να θεωρηθεί βέλτιστη πρακτική, όμως αλλάξτε το όνομα

Σημείωση: Είναι πολύ πιθανόν ότι πρέπει να χρησιμοποιήσουμε το #!/usr/bin/perl –wT για να εκκινήσουμε την Perl. Η παράμετρος w εμφανίζει τις ειδοποιήσεις (warnings) και η T ελέγχει τα δεδομένα με μεγαλύτερη προσοχή

# Πρώτο παράδειγμα σε C



- Όπως προαναφέραμε η γλώσσα δεν παίζει κανένα ρόλο στο CGI, για αυτό ξαναγρά-φουμε τον κώδικα σε C

```
#include <stdio.h>
main()
{
  printf("Content-type: text/html \n\n");
  printf("<html><head><title>Hello World from CGI
  in C</title></head><body>");
  printf("<h1 align='center'>Hello World!</h1>");
  printf("</body></html>");
}
```

# Μην επανεφεύρετε τον τροχό



- Δεν υπάρχουν πολλά πράγματα για να γίνουν με CGI
  - Συνήθως λαμβάνονται δεδομένα που υποβάλλονται από τους χρήστες και δίνεται ως έξοδος HTML και headers
- Μεγάλη πιθανότητα για λάθη
  - Υπόθεση ότι αυτό που στάλθηκε είναι σωστό κα ασφαλές
  - Δεν υπάρχουν μηχανισμοί για διαχείριση σφαλμάτων ή να μην αφήνουν εκτεθειμένους ευαίσθητους μηχανισμους
- Για την αποφυγή της επανεφεύρεσης του τροχού χρησιμοποιήστε τις διαθέσιμες βιβλιοθήκες CGI όπως η παρακάτω
  - http://search.cpan.org/dist/CGI.pm/

# Είσοδος/Έξοδος με CGI



- Τα προγράμματα CGI λαμβάνουν δύο είδη εισόδου
  - Δεδομένα που υποβάλει ο χρήστης
  - Δεδομένα από το περιβάλλον εκτέλεσης
- Τα δεδομένα από τον χρήστη συνήθως προέρχονται από φόρμες, γεγονότα από το ποντίκι κτλ, με τις μεθόδους POST & GET
- Τα δεδομένα περιβάλλοντος εκτέλεσης σχετίζονται με τα HTTP headers που στέλνονται από τον user-agent σε συνδυασμό με τοπικές μεταβλητές του εξυπηρετητή
- Στο προηγούμενο παράδειγμα είδαμε ότι η έξοδος του CGI ήταν Content-type: header με τον κατάλληλο τύπο (συνήθως text/html) ακολουθούμενη με το προσδιορισμένο περιεχόμενο (συνήθως html)

# Μεταβλητές Περιβάλλοντος



- Αυτές οι μεταβλητές αντιστοιχούν στα HTTP headers που υποβάλλονται στο πρόγραμμα
  - HTTP_ACCEPT, HTTP_USER_AGENT, HTTP_REFERER

  REMOTE_HOST, REMOTE_ADDR
- Δεδομένα που υποβάλλονται
  - QUERY_STRING, CONTENT_TYPE, CONTENT_LENGTH, REQUEST_METHOD
- Δεδομένα του εξυπηρετητή ή του προγράμματος
  - SERVER_NAME, SERVER_SOFTWARE, SERVER_PROTOCOL, SERVER_PORT, DOCUMENT_ROOT, SCRIPT_NAME
- Υπάρχουν πολλές άλλες ανάλογα με τον εξυπηρετητή που χρησιμοποιείται
- Όταν χρησιμοποιείται Perl, οι μεταβλητές περιβάλλοντος αποθηκεύονται στο %ENV, έτσι μπορούμε να τις προσπελάσουμε ή να τις τυπώσουμε εύκολα, όπως φαίνεται στο επόμενο παράδειγμα

# Παράδειγμα CGI Μεταβλητών Περιβάλλοντος



```
#!/usr/bin/perl
# print HTTP response header(s)
print "Content-type: text/html \n\n";
# print HTML file top
print <<END;
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html><head><title>Environment Variables</title>
</head><body><h1 align="center">Environment Variables</h1><hr />
END
# Loop over the environment variables
foreach $variable (sort keys %ENV) {
print "<b>$variable:</b> $ENV{$variable}<br />\n";
}
# Print the HTML file bottom
print <<END;
</body></html>
END
```

# Παράδειγμα CGI Μεταβλητών Περιβάλλοντος (βιβλιοθήκη cgi.pm)



```
#!/usr/bin/perl -wT
use strict;
use CGI qw(:standard);
print header;
print start_html("Environment Variables");
print "<h1 align='center'>Environment
  Variables</h1><hr
/>";
foreach my $key (sort(keys(%ENV))) {
print "$key = $ENV{$key}<br />\n";
}
print end_html
```

# Διαβάζοντας δεδομένα από τους χρήστες



- Βιβλιοθήκες όπως η CGI.pm διευκολύνουν την ανάγνωση δεδομένων παρέχοντας ζευγάρια <όνομα, τιμή> για τα δεδομένα που διαβάστηκαν
- Θα δούμε ότι σε scripting γλώσσες όπως η PHP αυτό γίνεται ακόμα πιο εύκολο.
- Έστω η παρακάτω φόρμα:

```
<html>
<head><title>Simple Form</title></head>
<body><h1>Form Test</h1><hr>
<form action="/cgi-bin/getdata.cgi" method="get">
Name: <input type="text" name="username"><br>
Password: <input type="password" name="password"><br>
Magic Number: <input type="text" name="magicnum" size="2"
maxlength="2"><br>
<input type="submit" value="send"></form>
</body>
</html>
```

# Διαβάζοντας δεδομένα από τους χρήστες (Συνέχεια)



- Διαβάζουμε τα δεδομένα ως εξής:

```
#!/usr/bin/perl -wT
use CGI qw(:standard);
use strict;
print header;
print start_html("Form Result");
print "<h1 align='center'>Form Result</h1><hr>";
my %form;
foreach my $p (param()) {
$form{$p} = param($p);
print "$p = $form{$p}<br>";
}
print end_html;
```

# Διαβάζοντας δεδομένα από τους χρήστες (Συνέχεια)



- Όταν χρησιμοποιείτε βιβλιοθήκες σαν την CGI.pm η επεξεργασία δεδομένων από το χρήστη γίνεται με τον ίδιο τρόπο
- Μπορούμε να ζητήσουμε απευθείας δεδομένα των οποίων ξέρουμε τις ονομασίες τους αντί να επεξεργαζόμαστε ότι μας παρέχεται
  - Ασφαλώς το γεγονός ότι επεξεργαζόμαστε **οτιδήποτε** μας παρέχει ο χρήστης είναι **ανασφαλές.** Γενικά πρέπει να επεξεργαζόμαστε αναμενόμενα (σε μέγεθος και ονομασία) ζευγάρια ονόματος-τιμής
  - Αφήνοντας το περιβάλλον εκτέλεσης να δημιουργεί global μεταβλητές (ή να επαναφέρει (reset) υπάρχουσες μεταβλητές) μπορεί να αποδειχθεί αρκετά επικίνδυνο

# CGI προβλήματα



- Όσον αφορά το θέμα της ασφάλειας γιατί το CGI θεωρείται ανασφαλές;
  - Συνήθως τα προγράμματα CGI εκτελούνται με περισσότερα δικαιώματα εκτέλεσης από ότι χρειάζονται
  - Κάποιοι χρησιμοποιούν το τερματικό!
    - Φανταστείτε δεδομένα που προέρχονται από το χρήστη να εκτελούνται σε τερματικό!!
  - Οι προγραμματιστές εμπιστεύονται τα δεδομένα εισόδου περισσότερο από ότι θα έπρεπε
    - Επιθέσεις cross site scripting, code injection
    - Αυτές οι επιθέσεις δεν γίνονται μόνο σε CGI όμως η αρχιτεκτονική που χρησιμοποιείται, ειδικά αν εκτελούνται scripts στο τερματικό, επιβαρύνουν περισσότερο τα προβλήματα

# CGI προβλήματα



- Προβλήματα ταχύτητας
  - Σε κάθε εκτέλεση ενός προγράμματος CGI ο εξυπηρετητής ρυθμίζει το περιβάλλον εκτέλεσης, φορτώνει το πρόγραμμα/script, το εκτελεί και μετά το καταστρέφει όταν τελειώσει
  - Πολλαπλοί χρήστες εκτελούν πολλαπλά προγράμματα CGI
  - Τα προγράμματα συνήθως υλοποιούνται σε διερμηνεύσιμες γλώσσες όπως η Perl
- Λύσεις:
  - Διατήρηση του script φορτωμένου στην μνήμη, εκτελούμενου ως co-process (πχ FastCGI - www.fastcgi.com)
  - Χρήση ενός embedded διερμηνέα για script (mod_perl)
  - Χρήση ενός προγράμματος server API (πχ ISAPI extension, Apache module)

# CGI προβλήματα



- Προβλήματα με την πολυπλοκότητα συγγραφής κώδικα
  - Ο προγραμματιστής πρέπει να ασχοληθεί με πολλές λεπτομέρειες
  - Με την χρήση βιβλιοθηκών, όπως η CGI.pm κάποιες από αυτές τις λεπτομέρειες αποκρύπτονται, όμως ο προγραμματιστής θα πρέπει να αντιλαμβάνεται τα headers, τις μεταβλητές περιβάλλοντος κτλ, πολύ περισσότερο από άλλα περιβάλλοντα προγραμματισμού
  - Κάποια θέματα όπως διαχείριση συνόδων (session management) δεν παρέχουν προγραμματιστική διαφάνεια στον προγραμματισμό με CGI ούτε σε ποιο μοντέρνα frameworks
  - Λύση: Προγραμματισμός σε περιβάλλοντα scripting όπως PHP, όπου αυτές οι λεπτομέρειες αποκρύπτονται καλύτερα
- Παρόλα αυτά, αρκετοί προγραμματιστές εξακολουθούν να χρησιμοποιούν αρκετά προγραμματισμό CGI και σε ορισμένες περιπτώσεις είναι χρήσιμος γιατί τους παρέχει περισσότερες λεπτομέρειες για τον εξυπηρετητή

# 23 Server Scripting Environments

# Εισαγωγή σε Server Side Scripting



- Το server-side scripting παρέχει μια ισορροπία ανάμεσα στην απόδοση και στην πολυπλοκότητα γραφής κώδικα
  - Λιγότερο δύσκολη υλοποίηση από server-modules (και ίσως CGI), αλλά με λιγότερη απόδοση
- Η γενική ιδέα είναι ή προσθήκη εντολών scripting στα αρχεία template (τα οποία συνήθως τα σκαφτόμαστε σαν τροποποιημένα αρχεία HTML)
  - Οι σελίδες που περιέχουν scripts, διερμηνεύονται από τον εξυπηρετητή Web και η έξοδος τους μετατρέπεται κατάλληλα – συνήθως σε HTML

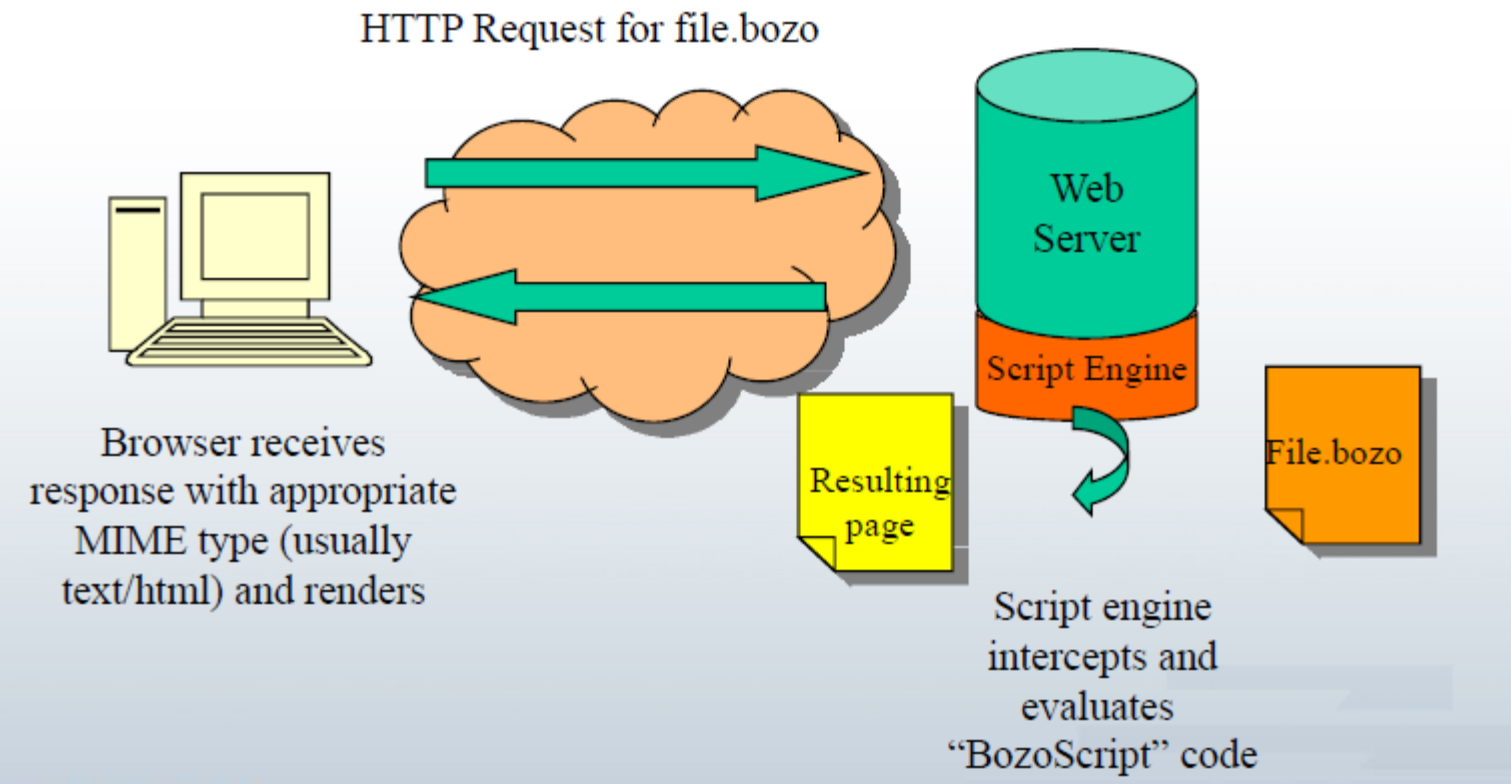
# Υλοποιώντας μια Server-Side Scripting γλώσσα



- Υποθέστε ότι θέλουμε να δημιουργήσουμε ένα server module, το BozoScript, το οποίο εξετάζει αν ζητείται ένα αρχείο με κατάληξη .bozo
  - Το server module είτε συλλαμβάνει τη σελίδα, την επεξεργάζεται και παράγει κατάλληλη έξοδο ή την προσπερνά χωρίς επεξεργασία
  - Η σελίδα συλλαμβάνεται σύμφωνα με την κατάληξη του αρχείου, όπου η επιλογή της κατάληξης .html σαν κατάληξη σύλληψης  είναι λανθασμένη επιλογή
    - Γιατί συλλαμβάνονται όλες οι σελίδες, κάτι που δεν χρειάζεται
    - Όμως αποκρύπτει τις λεπτομέρειες υλοποίησης

# Υλοποιώντας μια Server-Side Scripting γλώσσα





# Χαρακτηριστικά γλωσσών Server-Side Scripting



- Οι γλώσσες Server-Side Scripting έχουν κοινά χαρακτηριστικά
  - Διαφοροποίηση στην σύνταξη
    - Χρήση tag ή script
  - Εστιάζονται στα προβλήματα του προγραμματισμού στον ιστό
    - Session management, διαχείριση φορμών, προγραμματισμός σε βάσεις δεδομένων
  - Περιστασιακά ασχολούνται με τον διαχωρισμό του προγραμματιστή, σχεδιαστή (designer) και markup specialist
    - Μέσω της σύνταξης, διαχωρισμού της παρουσίασης και της λογικής του κώδικα κτλ
  - Στήριξη σε αντικείμενα για τις "απαιτητικές εργασίες"

# Χαρακτηριστικά γλωσσών Server-Side Scripting



- Γενιές Server-Side Scripting
  - Γενιά μηδέν: SSI (server-side includes)
  - Πρώτη Γενιά: ASP, ColdFusion, PHP
  - Δεύτερη Γενιά : JSP, ASP.NET
  - Τρίτη Γενιά : Declarative style- Flex –ή- 4GL σαν - Ruby on Rails
- Όλα τα περιβάλλοντα server-side scripting έχουν προβλήματα επεκτασιμότητας και απόδοσης, άλλα έχουν προβλήματα με την φορητότητα αφού συνδέονται στενά με μια πλατφόρμα
- Η δεύτερη γενιά επικεντρώνεται στην βελτίωση της απόδοσης ή/και στην υποστήριξη υλοποιήσεων μεγαλύτερων συστημάτων
- Η τρίτη γενιά επικεντρώνεται, ξανά, στην ευκολία προγραμματισμού που παρείχε η πρώτη γενιά κάνοντας την πολύ δημοφιλή

# Parsed Script – Εισαγωγή



- Παρουσιάζουμε μερικά από τα θέματα της πρώτης γενιάς γλωσσών server-side scripting
- Server Side Includes (SSI)
  - .αρχεία shtml
  - Σε μορφή σχολίου
  - <!--#include file="footer.html"-->
  - SSI υπάρχει σε κάθε σχεδόν εξυπηρετητή, παρόλο που συνήθως δεν είναι ενεργοποιημένο. Κάποιοι εξυπηρετητές μπορεί να έχουν μοναδικά χαρακτηριστικά
- Ακόμη και με τα SSI φαίνονται τα πρώτα ζητήματα
  - Πώς φαίνεται ότι το αρχείο έχει script στο εσωτερικό του (κατάληξη αρχείου)
  - Μέσα στο αρχείο πώς διαχωρίζεται ή οριοθετείται το script από το template (σε αυτή τη περίπτωση με τη μορφή σχολίου)
  - Αποδοτηκότα vs παραγωγικότητα (σε κώδικα)
- Κάποιοι editors όπως ο Dreamweaver παρέχουν αυτό το στυλ για template χωρίς άσχετα να πραγματοποιείται σύνθεση μιας server-side σελίδας

# ColdFusion Markup - Σύνοψη



- Η ColdFusion δημιουργήθηκε από την Allaire, συγχωνεύτηκε με την Macromedia, η οποία τώρα ανήκει στην Adobe
  - Είναι μια απλή server-side scripting γλώσσα, βασισμένη σε tags, η οποία στοχεύει στην σύνδεση ΒΔ με σελίδες
  - Για την λειτουργία της απαιτεί συγκεκριμένο Application Server, που είναι συμβατός με Windows, Linux και Solaris
    - Ο Application Server ενεργοποιείται όταν ένα αρχείο με κατάληξη .cfm ζητείται
- Η γλώσσα ColdFusion μοιάζει με HTML και για αυτό είναι δημοφιλής σε όσους είναι αρχάριοι στον προγραμματισμό
- Κύριος σκοπός της CF είναι η χρήση της για σύνδεση με ΒΔ
  - Χρησιμοποιώντας ODBC ή άλλους native drivers για βάσεις δεδομένων

# ColdFusion Markup - Σύνοψη



- Οι εντολές της CF είναι εντολές SQL (Structured Query Language)
- Έστω μια ΒΔ Positions, με πεδία Position-Num, JobTitle, Location, Description, Hiring Manager και PostDate. Είναι εύκολη η πρόσβαση στη ΒΔ με εντολές όπως οι παρακάτω
  - `SELECT * FROM Positions`
  - `SELECT * FROM Positions WHERE Location="Austin"`
  - 
```
SELECT *
  FROM Positions
  WHERE ((Location="Austin" OR
  (Location="Los Angeles") AND
  (Position="Game Tester"))
```

# <CFQUERY>



```
<CFQUERY NAME="ListJobs"
DATASOURCE="CompanyDataBase">
SELECT * FROM Positions
</CFQUERY>
```

- Σημείωση: Το γνώρισμα DATASOURCE είναι ίσο με CompanyDataBase που είναι η ODBC πηγή δεδομένων (data source) που περιέχει μια βάση δεδομένων που ονομάζεται Company, που με την σειρά της περιλαμβάνει τον πίνακα Positions από όπου εξάγουμε τα δεδομένα
- Το <CFQUERY> υποστηρίζει γνωρίσματα όπως NAME, DATASOURCE, MAXROWS, USERNAME, PASSWORD, TIMEOUT, και DEBUG

# <CFOUTPUT>



- Μόλις δεδομένα εξόδου είναι διαθέσιμα (από μεταβλητές, υπολογισμούς ή επερωτήσεις σε ΒΔ) χρησιμοποιείται το <CFOUTPUT>

```
<CFOUPUT QUERY="ListJobs">
<hr noshade><br>
Position Number: <b>#PositionNum#</b><br><br>
Title: #JobTitle#<br><br>
Location: #Location#<br><br>
Description: #Description#
</CFOUPUT>
```

- Παρατηρήστε τη χρήση του συμβόλου # για την συσχέτιση των πεδίων της ΒΔ με τα "slots" εξόδου. Επίσης παρατηρήστε την μίξη της HTML με τα πεδία της ΒΔ

# ColdFusion Παράδειγμα



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<CFQUERY NAME="ListJobs" DATASOURCE="CompanyDataBase">
    SELECT * from Positions
</CFQUERY>
<html>
    <head><title>Demo Company Job Listings</title></head>
    <body bgcolor="#FFFFFF">
           <h2 align="center">Job Listings</h2>
           <hr>
           <CFOUTPUT QUERY="ListJobs">
                      <hr noshade><br>
                      Position Number: #PositionNum#<br><br>
                      Title: #JobTitle#<br><br>
                      Location: #Location#<br><br>
                      Description: #Description#
           </CFOUTPUT>
           <hr>
           <address>
                      Demo Company, Inc.
           </address>
    </body>
</html>
```

# Συνήθη CFML Elements



- <CFABORT>
- <CFAPPLICATION>
- <CFCOL>
- <CFCONTENT>
- <CFCOOKIE>
- <CFERROR>
- <CFFILE>
- <CFHEADER>
- <CFIF>
- <CFINCLUDE>
- <CFINSERT>
- <CFLOCATION>
- <CFLOOP>
- <CFMAIL>
- <CFOUTPUT>
- <CFPARAM>
- <CFQUERY>
- <CFREPORT>
- <CFSET>
- <CFTABLE>
- <CFUPDATE>

# CFM Περίληψη



- Επικεντρώνεται στην δημιουργία ιστοσελίδων που συνδέονται με ΒΔ
- Πολύ υψηλού επιπέδου (σαν μια γλώσσα τέταρτης γενιάς)
- Πολύ προγραμματιστές θεωρούν την σύνταξη της περίεργη, χωρίς να μοιάζει με script
  - Παρόλα αυτά, νεότερες εκδόσεις το άλλαξαν αυτό, χωρίς όμως να χαθεί η αίσθηση του "baby coding"
- Είναι ακόμη προσανατολισμένη στα Windows παρόλο που μοιάζει να είναι ανεξάρτητη πλατφόρμας
- Η πρόσφατη χρήση συστημάτων που βασίζονται σε tags πχ XUL, XAML κτλ υποδηλώνει ότι τελικά η CF ως ιδέα είχε βάση

# Εισαγωγή σε ASP



- Η τεχνολογία Microsoft Active Server Pages (ASP) ήταν (και ως ένα βαθμό είναι ακόμη) μια ένα δημοφιλές server parsed scripting framework
  - Ενσωματωμένη στον IIS
  - Συνδεόμενη με αρχεία κατάληξης .asp
  - Δεν ορίζει κάποια γλώσσα είναι ένα framework
    - Αυτό συνήθως παρερμηνεύεται και συνήθως μπερδεύεται με την γλώσσα VBScript ή Jscript (κλώνος της JavaScript)
  - Ο κώδικας μοιάζει με την PHP αλλά είναι λιγότερο φιλικός προς τον προγραμματιστή
  - Με την εμφάνιση της ASP.NET πολλοί προγραμματιστές προσχώρησαν στην PHP ή και στη Ruby λόγω πολυπλοκότητας του κώδικα

# Παράδειγμα ASP



- Παρατηρήστε το tag <script> ορίζει το που θα γίνει η επεξεργασία και τη γλώσσα που χρησιμοποιείται και το script ορίζεται σε <% %>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<script language="VBScript" runat="Server"></script>
<html>
<head>
   <title>ASP Example</title>
</head>
<body>
   <h1>Breaking News</h1>
   <% = date() %>
   <p>Today the stock of a major software company <br>
   reached an all time high, making the Demo Company CEO<br>
   the world's first and only trillionaire.</p>
</body>
</html>
```

# Σύνοψη ASP



- Η σύνταξη της αφομοιώνεται ευκολότερα από τους προγραμματιστές, για παράδειγμα

```
<%
    Set Conn = Server.CreateObject("ADODB.Connection")
    Conn.Open ODBCPositions
    SQL = "SELECT JobTitle, Location, Description, HiringManager,
    PostDate FROM Positions"
    Set RS = Conn.Execute(SQL)
    Do While Not RS.EOF
%>
```

- Όμως είναι σωστό το να κάνουμε πολύπλοκες ενέργειες με script; Αντίθετα πρέπει να κάνουμε απλές ενέργειες με scripts και πολύπλοκες με αντικείμενα;
  - Τα scripts & objects έχουν νόημα με αυτό τον τρόπο και τελικά φαίνεται ότι οι διαφορές ανάμεσα στην CFM και στην ASP είναι κυρίως σύνταξης