Σχεδίαση Εφαρμογών και Υπηρεσιών Διαδικτύου
7η Διάλεξη: Προγραμματισμός στην πλευρά του εξυπηρετητή: Τεχνολογία Servlet

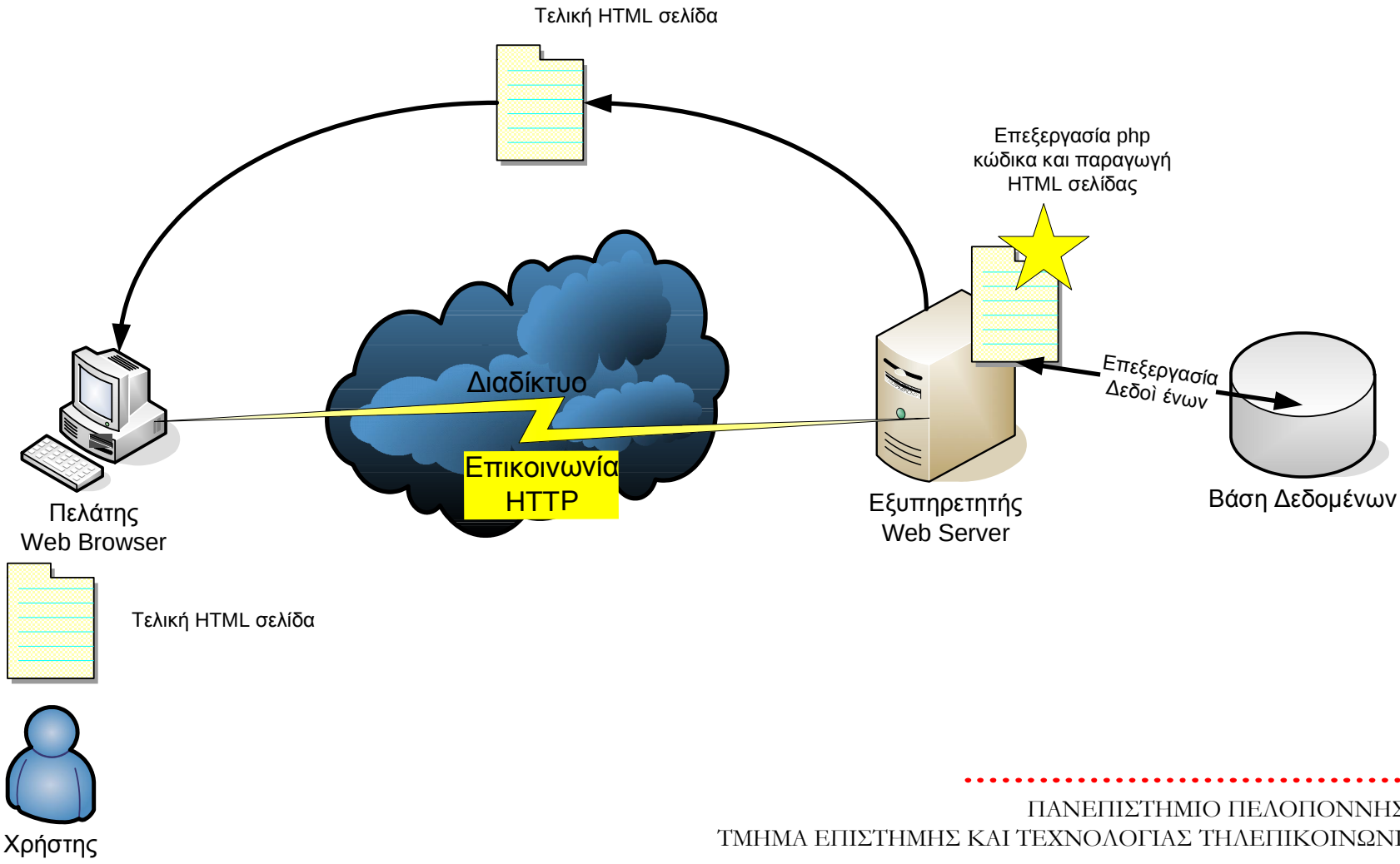
Δρ. Απόστολος Γκάμας

Λέκτορας (407/80)

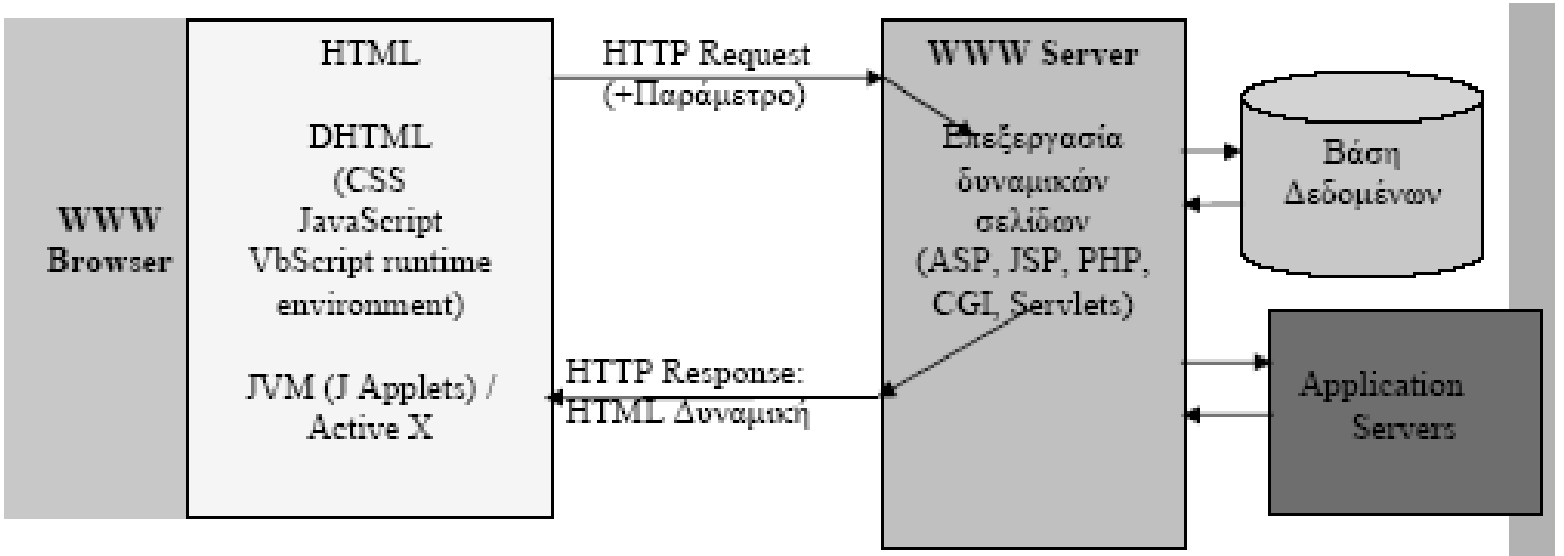
gkamas@uop.gr

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Προγραμματισμός στην πλευρά του εξυπηρετητή (server side)



Προγραμματισμός στην πλευρά του εξυπηρετητή (server side)



Server Side: Καταλληλότητα, Πλεονεκτήματα, Μειονεκτήματα

- Καταλληλότητα:
  - Δυναμική / Παραμετρική εμφάνιση περιεχομένου
  - Απαραίτητο όταν απαιτείται επικοινωνία (αλληλεπίδραση) με τον Server
  - Δυνατότητα ελέγχου των πελατών, π.χ. μετρητές επισκέψεων (hit counters), ελεγχόμενη πρόσβαση σε κάποιες σελίδες
- Πλεονεκτήματα:
  - Η επεξεργασία μεταφέρεται στο server, χρησιμοποιείται η ισχύς του server
  - Ο κώδικας είναι κρυφός
  - Η εκτέλεση του κώδικα είναι ανεξάρτητη του browser: στέλνεται «καθαρό»
  - HTML που εμφανίζεται πανομοιότυπο σε κάθε browser
  - Η μοναδική λύση για πρόσβαση στο file system του server
- Μειονεκτήματα:
  - Χρησιμοποιεί πολύτιμη επεξεργαστική ισχύ του server.
  - Κλιμάκωση (scalability);

Server Side: Τεχνολογίες

Τεχνολογία	Server	Πλεονεκτήματα
Active Server Pages (.asp)	MS IIS (Internet Information Server)	Σχετικά εύκολο στη χρήση, καλή ενσωμάτωση/ ολοκλήρωση (integration), καλή υποστήριξη (support) από Microsoft
ASP.NET	IIS	Κλιμάκωση (Scalability), web services.
Java Server Pages (.jsp)		Κλιμάκωση, αξιοπιστία
PHP (.php)	Apache (open source)	Δωρεάν, εύκολη στη χρήση
Java Servlets	Any web server	Προγραμματισμός σε Java, Εύκολη πρόσβαση σε ΒΔ

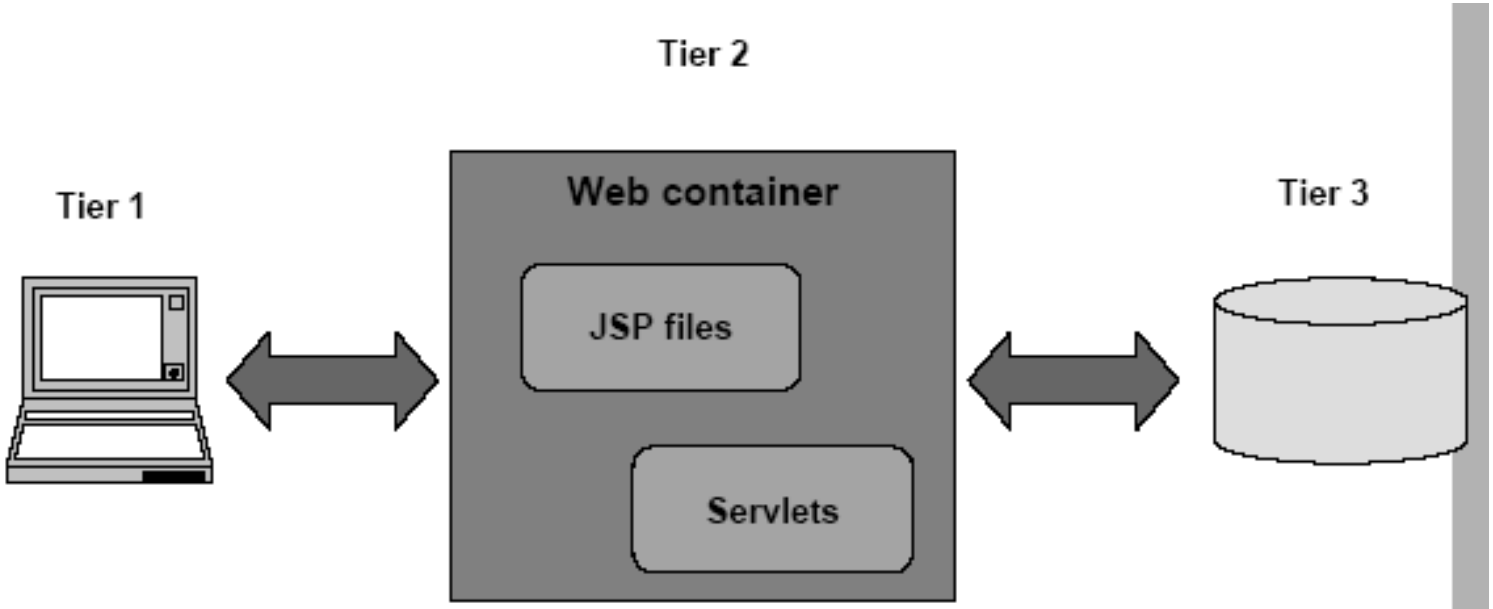
Τι είναι Servlet (SERVer appLET)

- Τα Servlets είναι μικρά προγράμματα γραμμένα στη γλώσσα Java που λειτουργούν στον server και επεκτείνουν τις λειτουργίες ενός Web Server.
- Όπως και οι άλλες αντίστοιχες τεχνολογίες (CGIs, ASP, PHP, ..), χρησιμοποιείται για την δημιουργία Web σελίδων που το περιεχόμενό τους δεν είναι στατικό αλλά μπορεί να εξαρτάται από τα δεδομένα που εισαγάγει ο χρήστη και χρειάζεται να ανακτηθεί από βάσεις δεδομένων ή από άλλα συστήματα

Τι είναι Servlet (SERVer appLET)

- Εκτελούνται σε ένα Web Server (Servlets: server-side πρόσωπο της Java – Applets: client-side πρόσωπο της Java, δηλ. εκτελούνται σε Web Browsers)
- Τα Servlets είναι εγκατεστημένα σε Web Server, δέχονται δεδομένα μέσω του πρωτοκόλλου HTTP και απαντούν στέλνοντας στον Web Browser αρχεία τύπου HTML.
- Για να προγραμματίσουμε Servlets είναι απαραίτητο το JSDK (Java Servlet Development Kit) ή Servlets API (Application Programming Interface) που είναι ενσωματωμένο σε αρκετά εργαλεία προγρ/μού Java (π.χ. NetBeans)
- Το Servlets API αποτελεί πλέον μέρος του JDK
- Τα Servlets υποστηρίζονται από (μπορούν να τρέξουν σε) σε πολλους web servers

Οι Java Servlets σε μια 3-tier αρχιτεκτονική



Η δύναμη των Servlet

- Είναι γραμμένα σε γλώσσα Java κατά συνέπεια είναι platform independent: “Write once Serve Everywhere”
- Εκμεταλλεύονται πλήρως το Java API, RMI, CORBA, Database Connectivity.
- Αποδοτικότητα & Αντοχή - Μένουν στην μνήμη μεταξύ διαδοχικών καλεσμάτων
- Κομψότητα (Elegance), Object-Oriented, Clean Code, Modular, Simple
- Λειτουργούν με το πρωτόκολλο HTTP

PHP vs. Servlets

- Τα servlets και τα PHP scripts αποτελούν εναλλακτικές για server-side προγραμματισμό.
- Τα servlets «φορτώνονται» μία φορά και όχι κάθε φορά που καλούνται, αντίθετα με τα PHP scripts
- Τα servlets είναι τεχνολογία που βασίζεται σε μια πλήρη αντικειμενοστραφή γλώσσα (Java), η PHP είναι γλώσσα σεναρίου (script)
- Υπάρχει διαφορά στη λογική: η PHP μοιάζει περισσότερο με την τεχνολογία JSP, ο PHP κώδικας είναι ενσωματωμένος σε HTML κώδικα, το στατικό HTML διακρίνεται από το HTML που παράγει δυναμικά η PHP. Οι servlets αποτελούν κλάσεις Java που όταν εκτελούνται παράγουν HTML κώδικα (στατικό & δυναμικό)

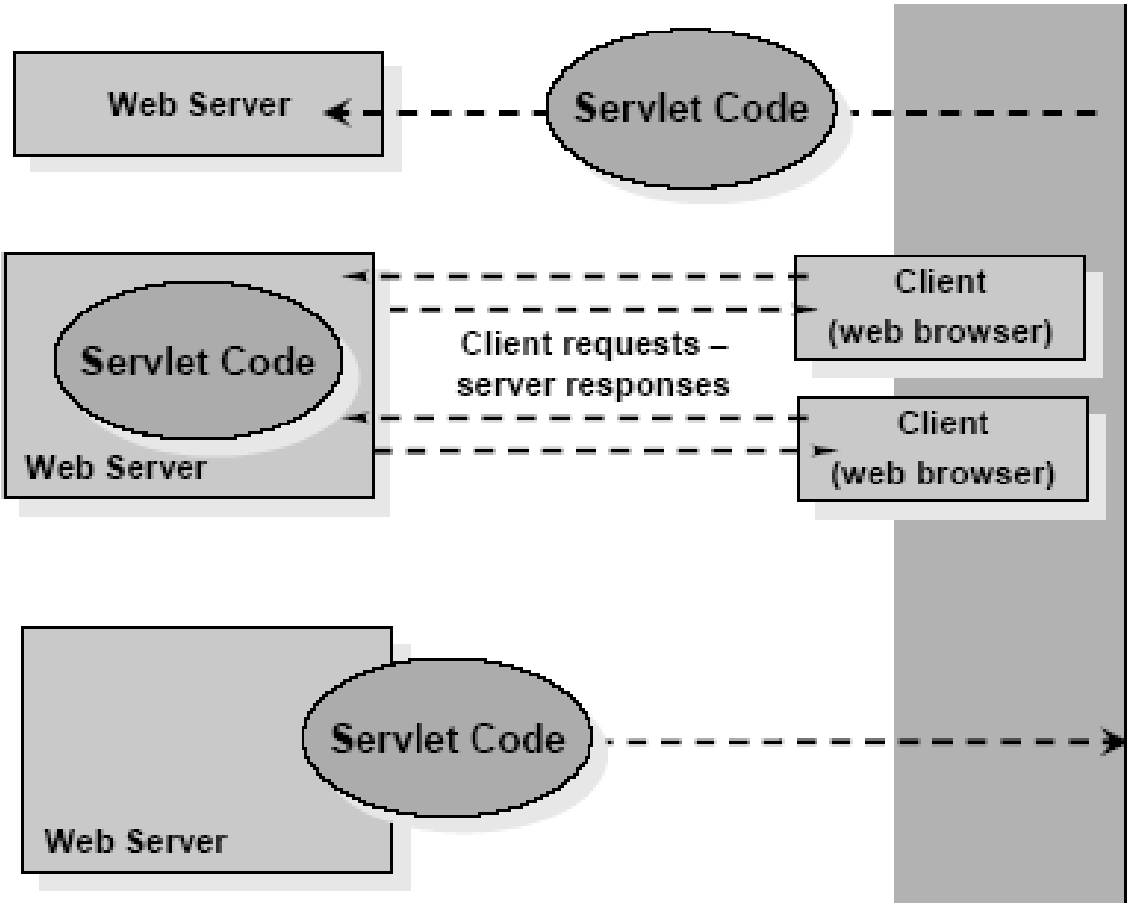
PHP vs. Servlets

- Οι κλήσεις σε «έτοιμες» (built-in) συναρτήσεις της PHP (που περιλαμβάνονται στις βιβλιοθήκες της PHP) είναι συνήθως γρηγορότερες από κλήσεις σε συναρτήσεις των Servlets. Το αντίστροφο όμως ισχύει για τον επιπλέον κώδικα που γράφει ο προγραμματιστής PHP.
- Η PHP προσφέρεται για γρήγορη ανάπτυξη κώδικα λόγω απλής σύνταξης
- Τα servlets προσφέρονται για μεγαλύτερης κλίμακας έργα λόγω της εκμετάλλευσης των πλούσιων βιβλιοθηκών αλλά και της αντικειμενοστραφούς (object-oriented) φύσης της Java.
- Με τους servlets είναι εύκολη η μετάβαση από μια ΒΔ σε άλλη (με αλλαγή λίγων γραμμών κώδικα)

Ο κύκλος ζωής του Servlet

- Κάθε servlet έχει τον ίδιο κύκλο ζωής:
  - Ο server το κάνει load και το αρχικοποιεί: «τρέχει» η μέθοδος init()
  - Το servlet δέχεται μηδέν ή και περισσότερα client requests: «τρέχουν» οι μέθοδοι service() ή doGet()/doPost()
  - Ο server το κάνει remove (ορισμένοι servers εκτελούν αυτό το βήμα μόνο όταν κάνουν shut down): «τρέχει» η μέθοδος destroy()

Ο κύκλος ζωής του Servlet



Αλληλοεπίδραση του servlet με τον πελάτη

- Όταν ένα servlet δέχεται ένα κάλεσμα από τον πελάτη (client), λαμβάνει δύο αντικείμενα (objects):
  - Ένα ServletRequest, που εξασφαλίζει την επικοινωνία από τον πελάτη προς τον server.
  - Ένα ServletResponse, που εξασφαλίζει την επικοινωνία από το servlet πίσω στον πελάτη.
  - (Τα ServletRequest και ServletResponse είναι interfaces ορισμένα στο javax.servlet package)

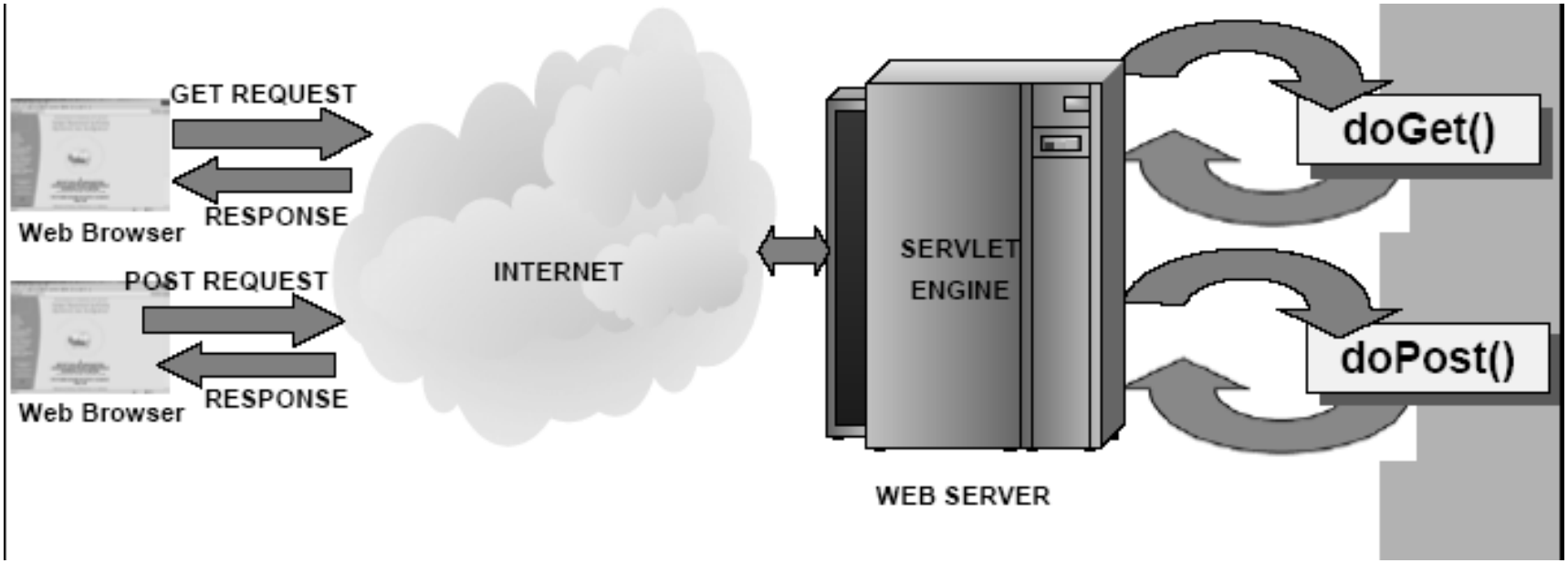
Αλληλοεπίδραση του servlet με τον πελάτη

- Βασικές μέθοδοι:
  - init (καλείται από τον servlet container στην αρχικοποίηση του servlet)
  - doGet (καλείται με HTTP Get request)
  - doPost (καλείται με HTTP Post request)
  - doPut (καλείται με HTTP Put request)
  - service (καλείται από τον servlet container σε κάθε HTTP request)
  - destroy (καλείται από τον servlet container όταν ο servlet πρόκειται να 'σβηστεί' από τη μνήμη)

Το Servlet API

- Όλα τα servlets υλοποιούν τη διασύνδεση (interface) Servlet μέσω μιας εκ των δύο βασικών κλάσεων:
- GenericServlet (javax.servlet package) - γενικά servlets ανεξάρτητα πρωτοκόλλου
- HttpServlet (javax.servlet.http package) - http servlets

Το Servlet API



Ένα απλό servlet

```
package mypackage;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;
public class Servlet1 extends HttpServlet
{
 private static final String CONTENT_TYPE = "text/html; charset=windows-1253";
 public void init(ServletConfig config) throws ServletException
 {
  super.init(config);
 }
```

Ένα απλό servlet

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
 {
   response.setContentType(CONTENT_TYPE);
   PrintWriter out = response.getWriter();
   out.println("<html>");
   out.println("<head><title>Servlet1</title></head>");
   out.println("<body>");
   out.println("<p>The servlet has received a GET. This is the reply.</p>");
   out.println("</body></html>");
   out.close();
  }
}
```

Προϋποθέσεις για να «τρέξουμε» ένα servlet

- Κάνουμε compilation του java file: SimpleServlet.java -> SimpleServlet.class (πρέπει να είναι εγκατεστημένο το Java Servlet API που περιέχει τις κλάσεις javax.servlet και javax.servlet.http)
- Απαιτείται ένας Application Server (ή Servlet Engine ή Servlet Container), π.χ. οι Oracle OC4J, Tomcat, JavaServer, ...
- Μπορούμε να «τρέξουμε» το servlet απ'ευθείας από κάποιο περιβάλλον προγραμματισμού σε Java (π.χ. ο NetBeans έχει ενσωματώσει το Servlet API και τον Application Server Tomcat)

Πως «τρέχουμε» ένα servlet

- Αν ο servlet ανήκει σε μια εφαρμογή που ονομάσαμε "ServletApp" και αποθηκεύσαμε το class αρχείου του servlet (αυτό που προκύπτει από το compilation) σε ένα directory, π.χ. το: \Project\classes\ServletPackage\
- Κλήση του servlet:
  http://localhost:8084/ServletApp/TestServlet
- Γενικά:
  http://<ServerName>:8084/<ProjectName>/<ServletName>

Εισαγωγή στο NetBeans

- Ένα IDE ανεπτυγμένο από την Sun
- Χρησιμότητα των IDEs (Integrated Development Environment): Ολοκληρωμένα περιβάλλοντα για ανάπτυξη εφαρμογών
  - Εύκολη και γρήγορη συγγραφή κώδικα
  - Ενσωματωμένος compiler
  - Τα IDEs για Java ενσωματώνουν και interpreter, applet viewer
  - Εύκολος εντοπισμός και διόρθωση λαθών
  - Ενσωματωμένος debugger
  - Ενσωματωμένο documentation, help
- Άλλα IDEs για ανάπτυξη εφαρμογών Java (και Servlets): JBuilder (Borland), Java Sun One (Sun), Visual Cafe (Symantec), VisualAge for Java (IBM)…
- Η έννοια του project

Περνώντας παραμέτρους στο servlet: η HTML φόρμα

```
<html>
<head>
<title>Servlet Example - Passing Parameters</title>
</head>
<body>
<form method="GET" action="Servlet2">
<p>Give your name: <input type="text" name="username" «size="20">
      <input type="submit" value="Try it"></p>
</form>
</body>
</html>
```

Περνώντας παραμέτρους στο servlet: Ο κώδικας του servlet

```
package mypackage;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.IOException;
public class Servlet2 extends HttpServlet
{
  private static final String CONTENT_TYPE = "text/html; charset=windows-1253";
  public void init(ServletConfig config) throws ServletException
  {
    super.init(config);
  }
```

Περνώντας παραμέτρους στο servlet: Ο κώδικας του servlet

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
 {
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();
  //Store the parameter value passed by the form
  String user = request.getParameter("username");
  // then write the data of the response
  out.println("<html>");
  out.println("<head><title>Servlet talking to an HTML form!...</title></head>");
  out.println("<body>");
  out.println("<h2>Hello " + user + "</h2>");
  out.println("<h5>The time is: " + new java.util.Date() + "</h5>");
  out.println("</body></html>");
  out.close();
 }
}
```

Πως ένα servlet μπορεί να απαντάει σε Get και Post requests με τον ίδιο ακριβώς τρόπο;

- Έχουμε έναν Servlet στον οποίο κάποια HTML φόρμα μπορεί να στείλει είτε GET είτε POST request.
- Θέλουμε η απόκριση του Servlet να είναι πανομοιότυπη ανεξαρτήτως του τύπου του request, χωρίς όμως να αντιγράψουμε τον κώδικα της doGet() στην doPost()
- Λύση: Απλά, η doPost() καλεί την doGet():

```
public void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
doGet(request, response);
}
```

Επικοινωνία μεταξύ Servlets

- Υπάρχουν πολλοί τρόποι με τους οποίους ένα servlet μπορεί να επικοινωνήσει με ένα άλλο servlet:
- Servlet Chaining (Κάνοντας διαδοχικά Post, Get)
  - Μπορούμε να αλισιδώσουμε μια σειρά από servlets
  - Θα μπορούσε πχ, να είναι μια διαδοχική καταχώρηση δεδομένων.
- Servlet Interface through a Database (πχ για ένα Chat Room)
  - Μπορούμε να καταχωρούμε στοιχεία από ένα servlet σε μια βάση δεδομένων όπου, μπορεί να τα ανακτήσει ένα άλλο servlet ταυτόχρονα και να δίνεται η εικόνα ότι επικοινωνούν ταυτόχρονα.
- Socket, RMI

ODBC (Open DataBase Connectivity)

- Η διεπαφή ODBC της Microsoft επιτρέπει σε εφαρμογές πρόσβαση σε συστήματα ΒΔ μέσω SQL
- Χρησιμοποιώντας ODBC ένας application developer μπορεί να αναπτύξει, μεταγλωτίσσει και προωθήσει μια εφαρμογή ανεξάρτητη του DBMS
- Διαφορετικά, η εφαρμογή δεν έχει μεταφερσιμότητα (non-portable) -> δύσκολη συντήρηση (δεν υποστηρίζονται άλλα DBMS ή άλλες εκδόσεις του ίδιου DBMS)
- Συστατικά μιας ODBC αρχιτεκτονικής:
- Application: καλεί ODBC συναρτήσεις
- Driver Manager: «φορτώνει» drivers για την εφαρμογή
- Driver: επεξεργάζεται και εκτελεί τις κλήσεις ODBC συναρτήσεων, στέλνει SQL requests και επιστρέφει τα αποτελέσματα στην εφαρμογή

JDBC (Java DataBase Connectivity)

- Το JDBC API της Sun Microsystems παρέχει ένα τυποποιημένο τρόπο πρόσβασης σε DBMS μέσω της γλώσσας Java. Με το JDBC, μια εφαρμογή έχει ομοιόμορφη πρόσβαση (με SQL ερωτήματα) στα δεδομένα ανεξαρτήτως του DBMS και τρέχει πανομοιότυπα σε οποιαδήποτε πλατφόρμα υποστηρίζει
- Το JDBC API ορίζει ένα σύνολο διεπαφών Java που ενσωματώνουν την κύρια λειτουργικότητα ΒΔ (εκτέλεση queries, επεξεργασία αποτελεσμάτων, ...)

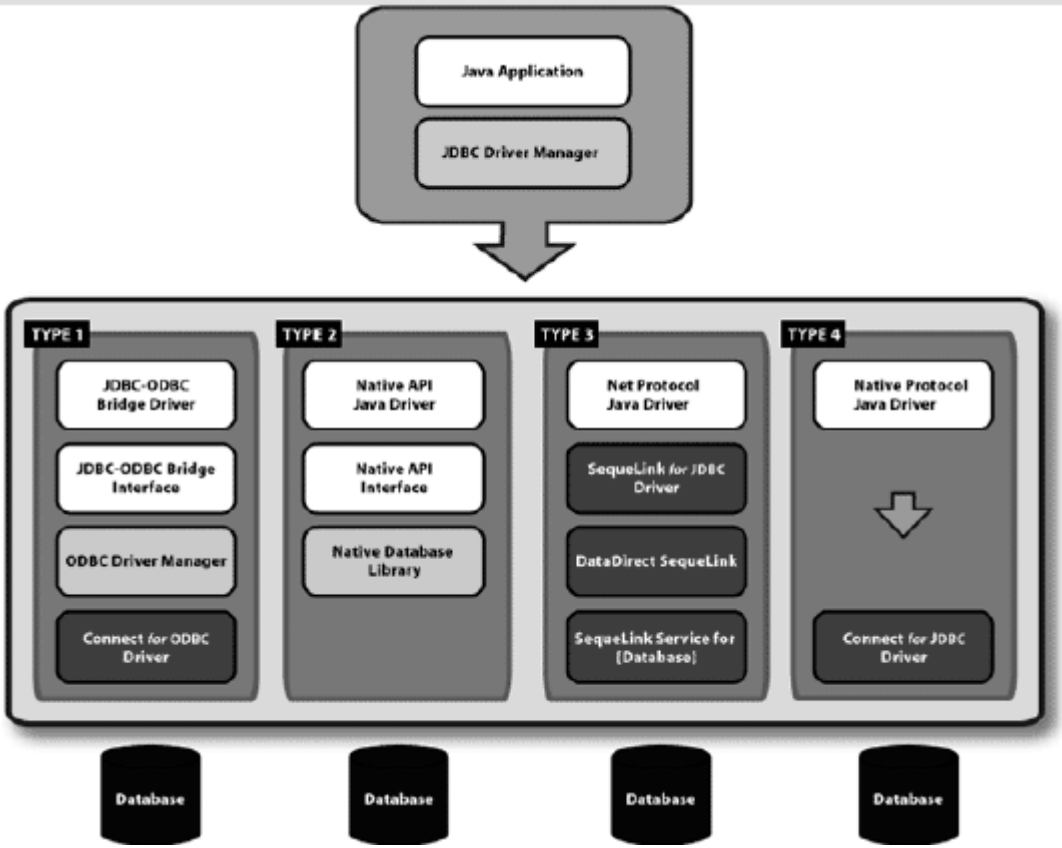
JDBC (Java DataBase Connectivity)

- Το JDBC API υλοποιείται μέσω του JDBC driver: ένα σύνολο κλάσεων που αποτελούν διεπαφές για επεξεργασία JDBC κλήσεων και επιστροφή αποτελεσμάτων στην εφαρμογή Java
- Γιατί χρειαζόμαστε το JDBC και δεν αρκούμαστε στο ODBC που είναι ένα API τυποποίησης της πρόσβασης σε ΒΔ;
  - Το ODBC δεν είναι κατάλληλο για άμεση χρήση από εφαρμογές Java γιατί είναι μια διεπαφή γραμμένη σε C
  - Το JDBC προσφέρει μια λύση για μια φυσική διεπαφή Java, δηλαδή μια «αμιγώς Java» στον application development

Τύποι JDBC drivers

- 4 είδη JDBC drivers σε χρήση:
  - Type 1: JDBC-ODBC bridge
  - Type 2: «μερικός» Java driver
  - Type 3: «αμιγής» Java driver προς ενδιάμεσο λογισμικό (middleware) ΒΔ
  - Type 4: «αμιγής» Java driver για άμεση πρόσβαση σε ΒΔ
- Οι «αμιγείς» λύσεις προσφέρουν ανώτερη απόδοση
- Το JDK περιέχει μόνο έναν JDBC driver, το jdbc-odbc bridge
- Για ΒΔ που δεν υποστηρίζονται από το ODBC, χρειαζόμαστε έναν JDBC driver για τη συγκεκριμένη ΒΔ (συνήθως αυτοί drivers πωλούνται)

Τύποι JDBC drivers



Διασύνδεση Servlets με Βάσεις Δεδομένων

- Τα Servlets όπως όλα τα αλλά προγράμματα Java μπορούν να συνδεθούν με βάσεις δεδομένων με την χρήση JDBC driver
- Το JDBC είναι database-independent. Πχ με αλλαγή 2 γραμμών κώδικα μπορούμε να αλλάξουμε τη βάση μας από Microsoft Access σε MySQL, χωρίς αλλαγή του υπόλοιπου κώδικα
- Κύριο πλεονέκτημα είναι ότι τα Servlets μπορούν να διατηρούν Open Database Connections, με αποτέλεσμα να μπορούν πολλά requests να εξυπηρετηθούν από ένα μόνο κάλεσμα, σε αντίθεση με τα CGI scripts
- Που βρίσκω τον κατάλληλο driver;
  - Η επιλογή εξαρτάται από την πλατφόρμα (λειτουργικό σύστημα) όπου τρέχει ο Servlet και από το RDBMS (ΒΔ) με την οποία θα επικοινωνήσει:
  - http://servlet.java.sun.com/products/jdbc/drivers

Εγκατάσταση JDBC driver

- Κατεβάζουμε (download) τον κατάλληλο JDBC driver (διαφορετικός για κάθε Βάση Δεδομένων)
  - Για MySQL: http://www.mysql.com/downloads/api-jdbc.html
- Αποσυμπιέζουμε (unzip) το αρχείο που κατεβάσαμε σε κάποιο φάκελο (directory)
- «Φόρτωμα» JDBC driver:
  - Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  - Class.forName("postgresql.Driver");
  - Class.forName("oracle.jdbc.driver.OracleDriver");

Διασύνδεση Servlets με Βάσεις Δεδομένων: Μεθοδολογία

```
import java.sql.*;
Class.forName("org.gjt.mm.mysql.Driver");
String url = "jdbc:mysql://host:port/db";
```

π.χ. String url = "jdbc:mysql://localhost:3306/books";

```
con = DriverManager.getConnection(url, "root", "");
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT …");
while(rs.next()) { … }
```

Γενική σύνταξη ενός connection URL:<url> ::=

jdbc:easysoft:[<server spec>]:[<database>]{:<attribute>=<value>}*<server spec> ::= //[<host name>][:<port>]/<database> ::= <dsn> | DSN=<dsn> | FILEDSN=<filedsn><DSNlessconnection string >

Εξαγωγή δεδομένων από βάση

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
 {
   response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();
  out.print("<html><head>");
  out.print("</head><body>");
  out.print("<code><pre>");
  out.print("<font color=green>ID\t Name\t\t Title\n</font>");
  // debugging info
  long time1 = System.currentTimeMillis();
  // connecting to database
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  try
  {
    // Load the JDBC-ODBC Bridge driver
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    // Get a connection to the database
    con = DriverManager.getConnection("jdbc:odbc:FPNWIND", "", "");
    stmt = con.createStatement();
    rs = stmt.executeQuery("SELECT customerId, contactName, contactTitle FROM customers");
    // displaying records
```

Εξαγωγή δεδομένων από βάση

```
while(rs.next())
    {
      out.print(rs.getObject(1).toString() + "\t");
      out.print(rs.getObject(2).toString() + "\t");
      out.print(rs.getObject(3).toString() + "\n");
    }
  }
  catch (SQLException e)
  {
    throw new ServletException("Servlet could not display records: " + e.toString(), e);
  }
  catch (ClassNotFoundException e)
  {
    throw new ServletException("JDBC Driver not found.", e);
  }
  // debugging info
  long time2 = System.currentTimeMillis();
  out.print("</pre></code>");
  out.print("<p>Search took : ");
  out.print( (time2 - time1) );
  out.print(" ms.</p>");
  out.print("<p\"><a href=\"");
  out.print(request.getRequestURI());
  out.print("\">Back</a></p>");
  out.print("</body></html>");
  out.close();
}
```

Εισαγωγή δεδομένων σε βάση


```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
 {
  response.setContentType(CONTENT_TYPE);
  PrintWriter out = response.getWriter();
  // Διάβασμα παραμέτρων
  String id = request.getParameter("id").trim();
  String name = request.getParameter("name").trim();
  String price = request.getParameter("price").trim();
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  try
  {
   // «Φόρτωμα» JDBC driver και σύνδεση στη ΒΔ
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   con = DriverManager.getConnection("jdbc:odbc:FPNWIND", "", "");
   String sql = "INSERT INTO Products(ProductID,ProductName, UnitPrice) VALUES (?,?,?)";
   PreparedStatement ps = con.prepareStatement(sql);
   stmt = con.createStatement();
   // Εισαγωγή εγγραφής
```

Εισαγωγή δεδομένων σε βάση

```
ps.setString(1, id);
    ps.setString(2, name);
    ps.setString(3, price);
    ps.executeUpdate();
  }
  catch (SQLException e)
  {
    throw new ServletException("Servlet could not insert records: " + e.toString(), e);
  }
  catch (ClassNotFoundException e)
  {
    throw new ServletException("JDBC Driver not found.", e);
  }
  out.print("<html><head>");
  out.print("</head><body>");
  out.print("<p> Data added </p>");
  out.print("</body></html");
  out.close();
}
```