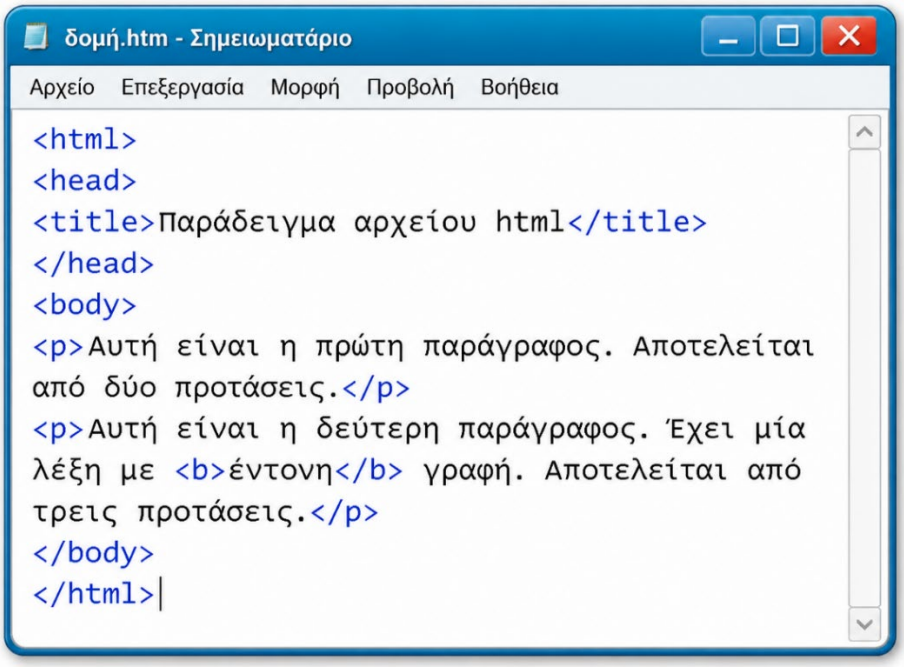
Κεφάλαιο 12 Servlets

Θα ξεκινήσουμε από βασικές έννοιες που θα βοηθήσουν στην κατανόηση της έννοιας (όρου) **Servlet** που είναι νέος όρος / τεχνολογία στη βιομηχανία της Πληροφορικής.

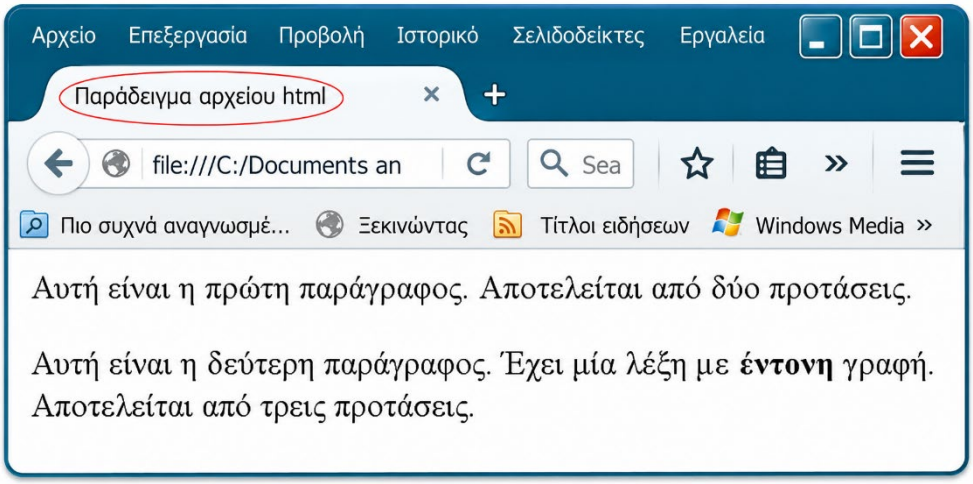
HTML

Πρόκειται για έγγραφα (αρχεία) κειμένου. Η κατάληξη του αρχείου είναι συνήθως **htm** ή **html**. Μέσα στο αρχείο καταγράφονται ετικέτες (**tags**). Οι ετικέτες είναι δεσμευμένες λέξεις μέσα σε γωνιακές αγκύλες. Κάθε ετικέτα έχει και τη συμμετρική της ετικέτα κλεισίματος. Η κωδικοποίηση του αρχείου κειμένου μπορεί να είναι συνήθως **ANSI** (**ISO-8859-1**, **ISO-8859-7**, κλπ) ή **UTF8**. Η επόμενη εικόνα δείχνει έναν εκδότη κειμένου (το notepad) για τη συγγραφή μιας html σελίδας (αρχείο **δομή.htm**).



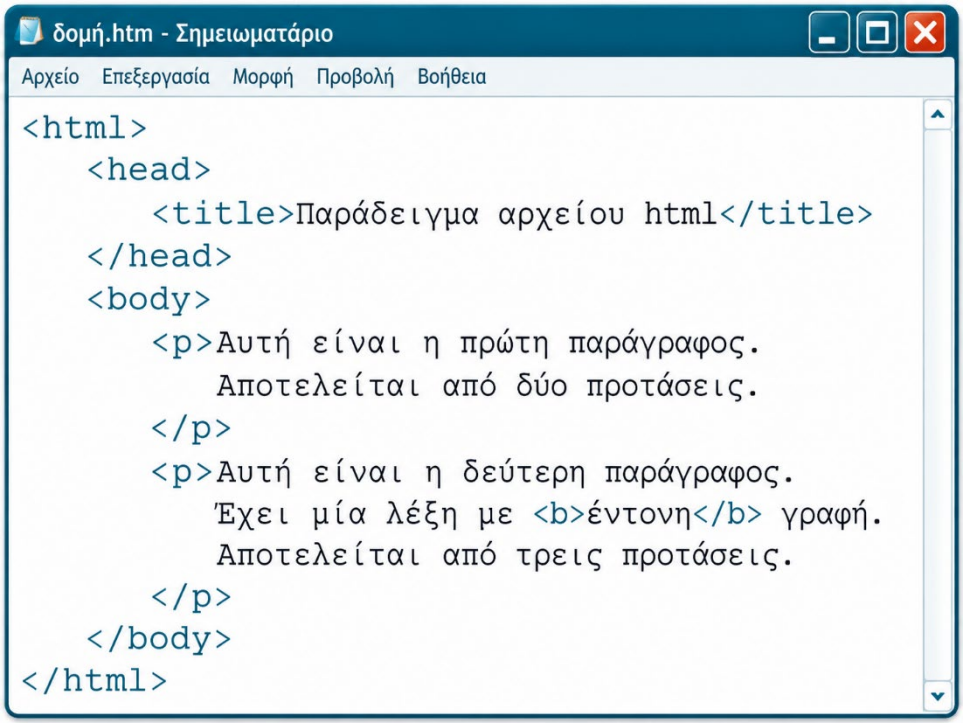
Εικόνα 12.1 Παράδειγμα αρχείου **html** σε σημειωματάριο (notepad)

Αυτό το αρχείο (αυτή η **html** σελίδα) εμφανίζεται με τον ακόλουθο τρόπο (επόμενη εικόνα) μέσα σε ένα browser:



Εικόνα 12.2 Εμφάνιση της **html** σελίδας σε φυλλομετρητή

Οι ετικέτες μέσα σε ένα **html** αρχείο είναι συμμετρικές και μπορεί να είναι και εμφωλευμένες. Ένα **html** αρχείο μπορούμε να το στοιχειοθετήσουμε με εσοχές (indentation) για να έχουμε καλύτερη ανάγνωση, χωρίς να διαφοροποιείται το αποτέλεσμα που εμφανίζεται στο browser. Στην επόμενη εικόνα βλέπουμε ξανά το αρχείο **δομή.htm** με εσοχές (indentation) προκειμένου να είναι ευκρινέστερες οι εμφωλεύσεις ετικετών:



Εικόνα 12.3 Παράδειγμα αρχείου **html** σε σημειωματάριο (notepad) με εσοχές

Πρώτο Παράδειγμα Servlet – Hello World

Πάμε κατευθείαν στα βαθιά και βλέπουμε τον πηγαίο κώδικα ενός Servlet που δημιουργεί δυναμικό κώδικα **html** που εμφανίζει ένα μήνυμα καλωσορίσματος και ορισμένες άλλες πληροφορίες. Το πηγαίο μας πρόγραμμα περιέχει μια κλάση **HelloServlet** και είναι αποθηκευμένο στο αρχείο **HelloServlet.java** (έστω στον κατάλογο C:\Users\nnk\Documents\JavaProjects\helloservlet):

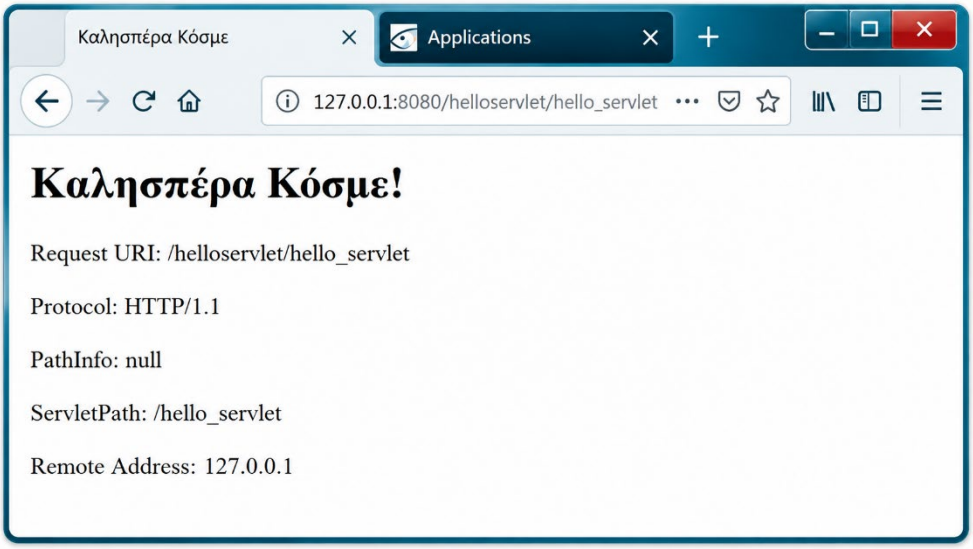
Κώδικας 12.1 – Παράδειγμα Servlet

```
1. import java.io.*;
2. import javax.servlet.*;
3. import javax.servlet.http.*;
4.
5. public class HelloServlet extends HttpServlet {
6.    public void doGet(HttpServletRequest request, HttpS-
                        ervletResponse response)
7.       throws IOException, ServletException {
8.       // Set the response message's MIME type
9.       response.setContentType("text/html;charset=UTF-8");
10.       // Allocate an output writer to write the Response Message Body
11.       PrintWriter out = response.getWriter();
12.
13.       // Write the Response Message Body, which is a dynamic HTML page
14.       try {
15.          out.println("<html><head>");
16.          out.println("<meta http-equiv='Content-Type' content=
                          'text/html; charset=UTF-8'>");
17.          out.println("<title>Καλησπέρα Κόσμε</title></head>");
18.          out.println("<body>");
19.          out.println("<h1>Καλησπέρα Κόσμε!</h1>");
20.          // Echo client's request information
21.          out.println("<p>Request URI: " + request.getRequestURI
                         () + "</p>");
22.          out.println("<p>Protocol: " + request.getProtocol() + "</p>");
23.          out.println("<p>PathInfo: " + request.getPathInfo() + "</p>");
24.          out.println("<p>ServletPath: " + request.getServletPath() + "</p>");
25.          out.println("<p>Remote Address: " + request
                          .getRemoteAddr() + "</p>");
26.          out.println("</body>");
27.          out.println("</html>");
28.       } finally {
29.          out.close();  // Always close the output writer
30.       }
31.    }
32. }
```

Η κλάση μας είναι παιδί (extends) της κλάσης **HttpServlet**. Η κλάση HttpServlet είναι βασική στο Java Web Programming και διαθέτει τις μεθόδους **doDelete**, **doGet**, **doHead**, **doOptions**, **doPost**, **doPut**, **doTrace**, κλπ. Όλες αυτές οι μέθοδοι καλούνται με δύο ορίσματα. Το πρώτο όρισμα είναι ένα αντικείμενο που ακολουθεί τη διασύνδεση **HttpServletRequest** και το δεύτερο όρισμα είναι ένα αντικείμενο που ακολουθεί τη διασύνδεση **HttpServletResponse**. Εμείς εδώ υποκαθιστούμε την κληρονομημένη (από την κλάση **HttpServlet**) μέθοδο **doGet** για να χειριστούμε με τον τρόπο που επιθυμούμε ένα **GET Request** που φτάνει στην κλάση **HelloServlet**. Όπως αναφέραμε, η μέθοδος **doGet** λαμβάνει δύο ορίσματα: το request (που ακολουθεί τη διασύνδεση **HttpServletRequest** και επομένως διαθέτει όλες τις μεθόδους που ορίζει η διασύνδεση **HttpServletRequest**) και το response (που ακολουθεί τη διασύνδεση **HttpServletResponse** και επομένως διαθέτει όλες τις μεθόδους που ορίζει η διασύνδεση **HttpServletResponse**). Το όρισμα request της μεθόδου **doGet** είναι μια κλάση που γνωρίζει και μπορεί (με τις μεθόδους που διαθέτει από τη διασύνδεση **HttpServletRequest**) να μας επιστρέψει όλες τις πληροφορίες ενός **HTTP Request**. Το όρισμα response της μεθόδου **doGet()** είναι μια κλάση που μπορεί (με τις μεθόδους που διαθέτει από τη διασύνδεση **HttpServletResponse**) να σχηματίσει (μορφοποιήσει κατάλληλα) ένα **HTTP Response** που θα αποσταλεί στον αποδέκτη του (συνήθως στον browser).

Στη γραμμή 9 χρησιμοποιούμε τη μέθοδο **setContentType()** για να ορίσουμε έναν από τους Response Headers του **HTTP Response**. Στη γραμμή 11 με τη μέθοδο **getWriter()** αποκτάμε ένα character stream μέσω του οποίου μπορούμε να σχηματίσουμε / μορφοποιήσουμε το Response Message Body του **HTTP Response**. Στις γραμμές 15 έως 19 μορφοποιούμε το Response Message Body έτσι ώστε να είναι μια **html** σελίδα με ένα καλωσόρισμα. Στις γραμμές 21 έως 25 χρησιμοποιούμε τις μεθόδους **getRequestURI()**, **getProtocol()**, **getPathInfo()**, **getServletPath()** και **getRemoteAddr()** για να αντλήσουμε πληροφορίες από τη **Request Line** και τα **Request Headers** του **HTTP Request**. Στη συνέχεια αυτές τις πληροφορίες τις κάνουμε **echo** (τις προσθέτουμε – με **println()** – στο **Response Message Body** του **HTTP Response**). Στις γραμμές 26 και 27 ολοκληρώνουμε (με τα html tag **</body>** και **</html>**) την **html** σελίδα που «ταξιδεύει» στον browser ως **Response Message Body**. Ο browser λαμβάνει το HTTP Response και από

αυτό αντλεί τη δυναμικά δημιουργημένη HTML σελίδα και την εμφανίζει όπως φαίνεται στην επόμενη εικόνα:

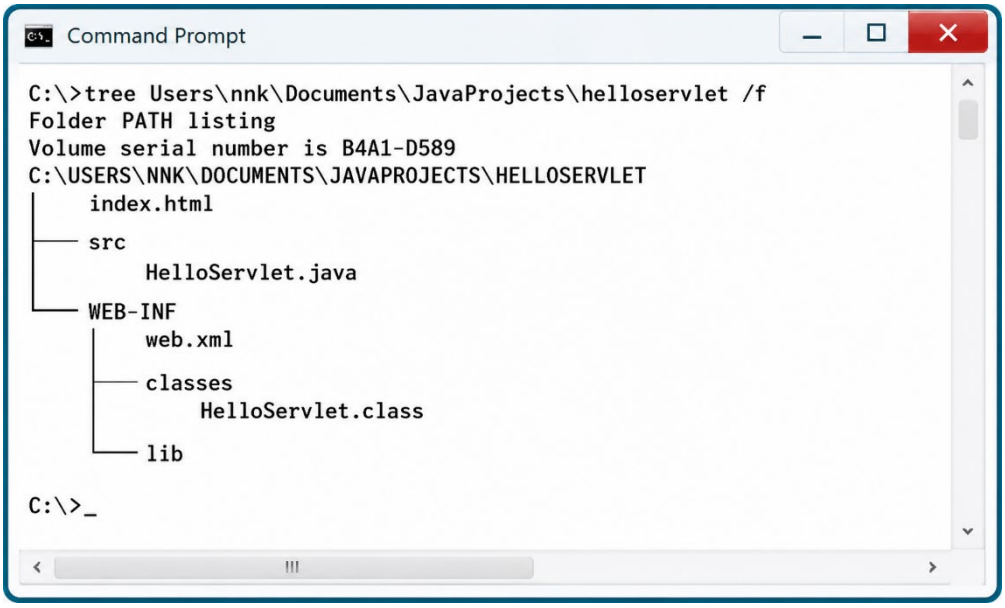


Το παραπάνω πρόγραμμα αρχικά δοκιμάστηκε σε Glassfish 4.0 με Java 7 και Servlet API 3.0.1. Σε νεότερες εκδόσεις της Java και κυρίως του Servlet API θα πρέπει να αντικαταστήσουμε τις γραμμές 2 και 3 με τις επόμενες:

```
2. import jacarta.servlet.*;
3. import jacarta.servlet.http.*;
```

Δομή Καταλόγων Εφαρμογής Servlet και Deploy

Μια εφαρμογή Servlet πρέπει να ακολουθεί μια συγκεκριμένη δομή καταλόγων για την αποθήκευση των αρχείων της εφαρμογής. Η δομή αυτή έχει καθοριστεί από τη Sun και ακολουθείται υποχρεωτικά. Πριν περιγράψουμε αυτή τη δομή θα δούμε ένα παράδειγμα. Το παράδειγμα του helloservlet του οποίου το πηγαίο (HelloServlet.java) είδαμε πριν:



Ο κατάλογος «C:\Users\nnk\Documents\JavaProjects\helloservlet» είναι ο «root folder» και είναι δική μας επιλογή. Μέσα σε αυτόν πρέπει να υπάρχει ο κατάλογος «WEB-INF». Μέσα στον «WEB-INF» πρέπει να υπάρχουν οι κατάλογοι «classes» και «lib». Στον κατάλογο «classes» μπαίνουν τα **.class** αρχεία μας (τα εκτελέσιμα του servlet). Στον κατάλογο «lib» μπαίνουν τα **.jar** αρχεία μας (οι βιβλιοθήκες που χρησιμοποιεί το servlet). Το αρχείο **web.xml** έχει ιδιαίτερη σημασία και θα το αναπτύξουμε διεξοδικά παρακάτω. Αν συντηρούμε τα πηγαία μας προγράμματα μέσα στο «root folder» τότε αυτά θα βρίσκονται στον κατάλογο «src». Όλα τα άλλα (αν υπάρχουν), για παράδειγμα αρχεία **.htm**, **.html**, **.jsp**, **.gif**, **.js** θα πρέπει να βρίσκονται στο «root folder».

Στο προηγούμενο παράδειγμα έχουμε ένα άδειο «lib», έχουμε το πηγαίο (HelloServlet.java) στον «src» κατάλογο. Επίσης έχουμε το αρχείο index.html στον «root folder». Τα περισσότερο ουσιαστικά (και αναγκαία) είναι το **web.xml** στον κατάλογο «WEB-INF» και το εκτελέσιμο **HelloServlet.class** στον κατάλογο «classes» (μέσα στον κατάλογο «WEB-INF»).

Με βάση τα παραπάνω μπορούμε να δούμε τα βήματα που κάναμε για να δημιουργήσουμε, μεταγλωττίσουμε και να θέσουμε σε λειτουργία (να κάνουμε deploy) το **helloservlet**. Ας το δούμε σε βήματα:

1. Δημιουργία του «root folder» και της υπόλοιπης δομής

```
C:\Users\nnk\Documents\JavaProjects> mkdir helloservlet
C:\Users\nnk\Documents\JavaProjects> cd helloservlet
C:\Users\nnk\Documents\JavaProjects\helloservlet> mkdir src
C:\Users\nnk\Documents\JavaProjects\helloservlet> mkdir WEB-INF
C:\Users\nnk\Documents\JavaProjects\helloservlet> mkdir WEB-INF\classes
C:\Users\nnk\Documents\JavaProjects\helloservlet> mkdir lib
```

2. Λήψη της βιβλιοθήκης Servlet API library

Κατεβάζουμε την Servlet API library και την αποθηκεύουμε τοπικά σε έναν κατάλογο που συνηθίζουμε να αποθηκεύουμε τις βιβλιοθήκες Java. Εμείς έχουμε επιλέξει να φυλάμε τις βιβλιοθήκες στον κατάλογο C:\Users\nnk\Documents\JavaProjects\

i. Στην αρχική έκδοση

Κατεβάσαμε την Servlet API library 3.0.1 (javax.servlet-api-3.0.1) από τη διεύθυνση http://repo1.maven.org/maven2/javax/servlet/javax.servlet-api/3.0.1/

ii. Στην πρόσφατη έκδοση

Κατεβάσαμε το Jakarta Servlet API 6.0 από τη διεύθυνση https://repo1.maven.org/maven2/jakarta/servlet/jakarta.servlet-api/6.0.0/jakarta.servlet-api-6.0.0.jar

3. Δημιουργούμε το πηγαίο πρόγραμμα του Servlet

Φτιάχνουμε το πηγαίο πρόγραμμα (που είδαμε παραπάνω) με κωδικοποίηση **UTF-8** without BOM και το αποθηκεύουμε στο src\HelloServlet.java κάτω από το «root folder». Προσοχή στα import της 2ης και 3ης γραμμής ανάλογα το Servlet API.

4. Κάνουμε compilation

i. Στην αρχική έκδοση

```
C:\Users\nnk\Documents\JavaProjects\helloservlet>
javac -encoding utf8 -cp
"C:\Users\nnk\Documents\JavaProjects\javax.servlet-api-3.0.1.jar"
src\HelloServlet.java
```

ii. Στην πρόσφατη έκδοση

```
C:\Users\nnk\Documents\JavaProjects\helloservlet>
javac -encoding utf8 -cp
"C:\Users\nnk\Documents\JavaProjects\jakarta.servlet-api-6.0.0.jar"
src\HelloServlet.java
```

Μετά από αυτό (το κατάλληλο από τα 2 εναλλακτικά compilations) το πηγαίο (.java) και το εκτελέσιμο (**.class**) βρίσκονται στον υποκατάλογο «src».

5. Μεταφέρουμε το εκτελέσιμο στον κατάλογο «classes»

```
C:\Users\nnk\Documents\JavaProjects\helloservlet>
MOVE src\HelloServlet.class WEB-INF\classes
```

6. Δημιουργούμε το web.xml

Με έναν εκδότη κειμένου (notepad ή notepad++) δημιουργούμε το επόμενο (xml) κείμενο:

```
<web-app>

   <servlet>
      <servlet-name>HelloServletByNNK</servlet-name>
      <servlet-class>HelloServlet</servlet-class>
   </servlet>

   <servlet-mapping>
      <servlet-name>HelloServletByNNK</servlet-name>
      <url-pattern>/hello_servlet</url-pattern>
   </servlet-mapping>

</web-app>
```

και το αποθηκεύουμε με όνομα **web.xml** και με κωδικοποίηση **ISO-8859-1** στον κατάλογο WEB-INF.

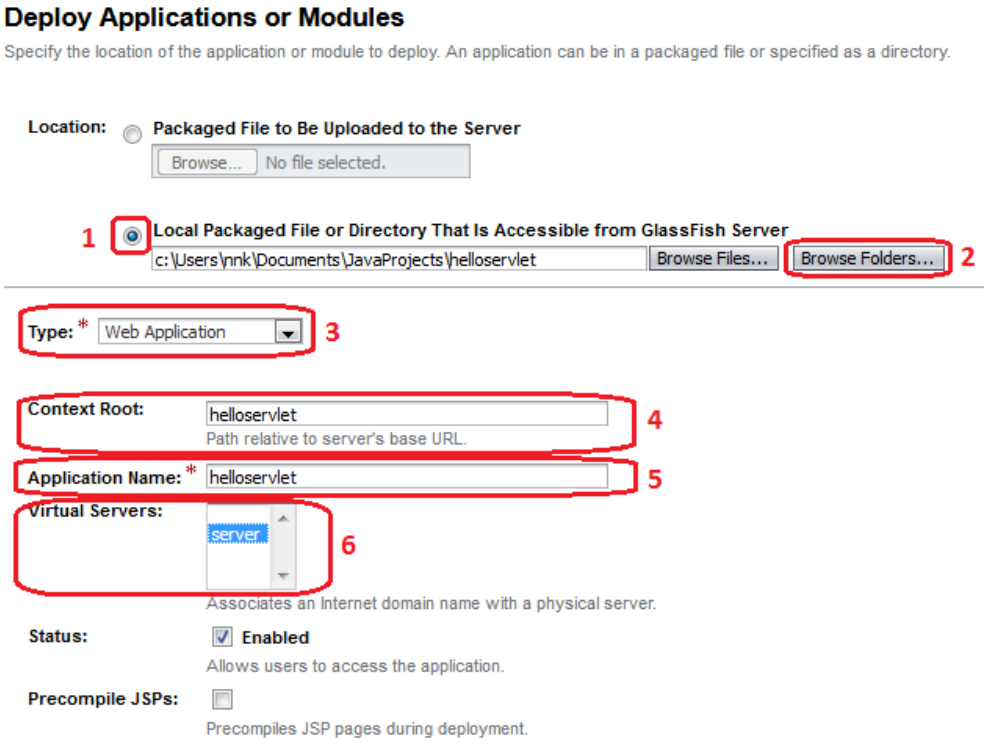
Στην πρόσφατη έκδοση (με το Jakarta Servlet API 6.0) μπορούμε να προσθέσουμε στο **web.xml** το επόμενο προοίμιο:

```
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
         metadata-complete="false"
         version="6.0">
```

7. Κάνουμε Deploy (θέτουμε σε λειτουργία)

i. Στην αρχική έκδοση

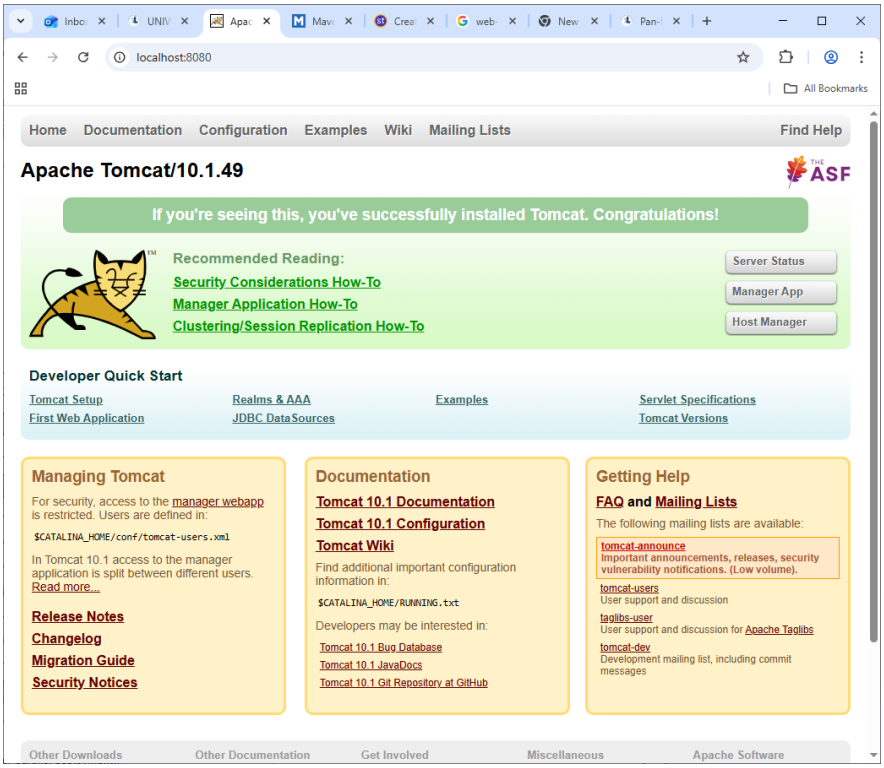
Τρέχουμε το εργαλείο διαχείρισης του Glassfish (έχουμε προηγουμένως εγκαταστήσει το Glassfish 4 και έχουμε ξεκινήσει το default domain του Glassfish, βλέπε σχετικό παράρτημα) δίνοντας τη διεύθυνση http://localhost:4848 στον browser μας. Από εκεί επιλέγουμε (κάνουμε κλικ) διαδοχικά τα «Common Tasks» και «Deploy an Application». Βλέπουμε την επόμενη εικόνα όπου εκτελούμε τις 6 υποδεικνυόμενες ενέργειες.



Εικόνα 12.4 Ρύθμιση & ανάπτυξη εφαρμογής στο GlassFish με αριθμημένα βήματα

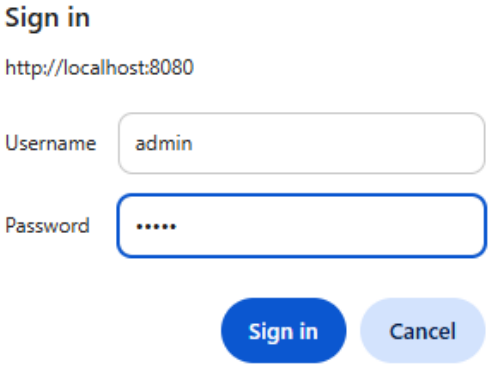
ii. Στην πρόσφατη έκδοση

Τρέχουμε το εργαλείο διαχείρισης του Tomcat (έχουμε προηγουμένως εγκαταστήσει τον Tomcat 10.1.49 και τον έχουμε ξεκινήσει, βλέπε σχετικό παράρτημα) δίνοντας τη διεύθυνση http://localhost:8080 στον browser μας. Βλεπουμε την επομενη εικονα:



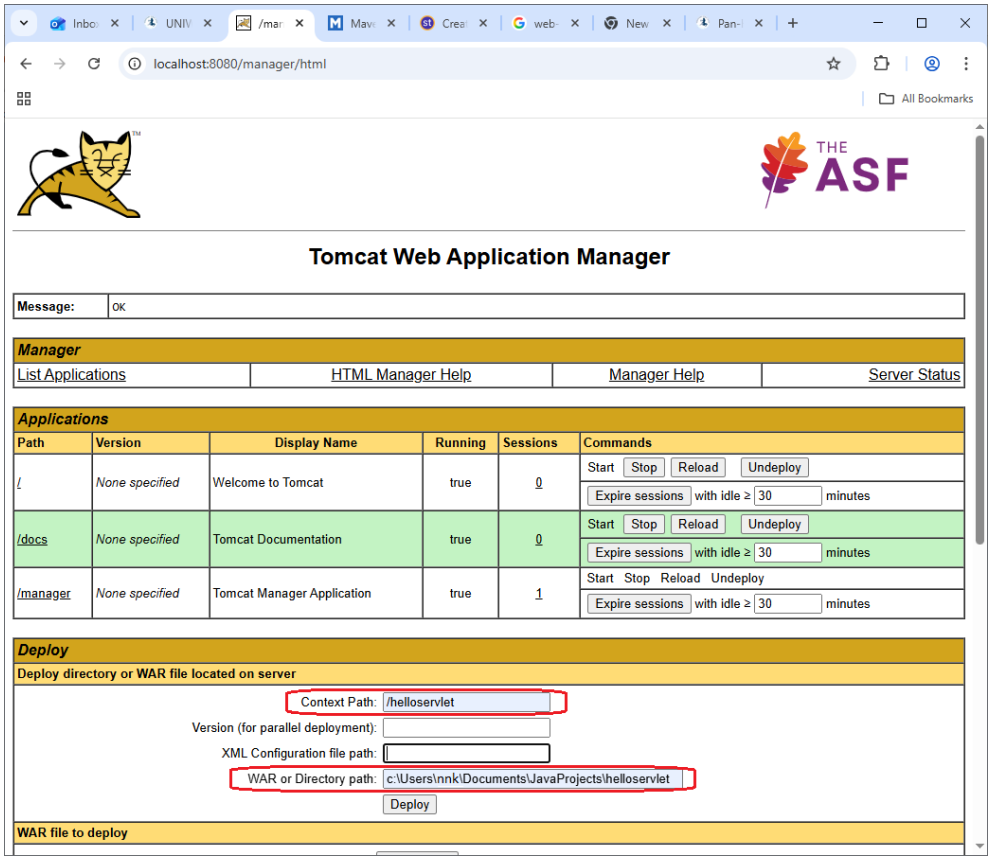
Εικόνα 12.5 Αρχική σελίδα του Apache Tomcat

Από εκεί επιλέγουμε (κάνουμε κλικ) στο κουμπι “Manager App”. Στο σημειο αυτό μας ζηταει τα στοιχεια ταυτοποιησης τα οποια εχουμε ορισει από την εγκατασταση του Tomcat. Βλεπουμε:



Εικόνα 12.5 Εισαγωγή στοιχειων ταυτοποίησης για την σύνδεση

Αφου εισαγαγουμε (πληκτρολογησουμε) τα στοιχεια ταυτοποιησης εμφανιζεται η επομενη εικονα. Σε αυτή εισαγαγουμε το Context Path και το Directory Path του project helloservlet και παταμε το κουμπι “Deploy”.



Εικόνα 12.6 Εισαγωγή ContextPath & Directory Path

8. Τρέχουμε την εφαρμογή

Στον browser δίνουμε τη διεύθυνση http://127.0.0.1:8080/helloservlet/hello_servlet

Και όπως αναμένουμε θα δούμε την εικόνα του «**Καλησπέρα Κόσμε**» που είδαμε παραπάνω (πριν 6 σελιδες).

Δομή web.xml

Έχουμε δει ένα παράδειγμα του **web.xml**. Αυτό έχει μια ετικέτα **<web-app>** (και τη συμμετρική της **</web-app>**) μέσα στην οποία φωλιάζουν μία ή περισσότερες ετικέτες **<servlet>** και ακολουθούν ίσες σε πλήθος ετικέτες **<servlet-mapping>**. Κάθε ετικέτα **<servlet>** συνδυάζεται με μία ετικέτα **<servlet-mapping>** και ο συνδυασμός βασίζεται στο κοινό περιεχόμενο της ετικέτας **<servlet-name>** που εμπεριέχουν αμφότερες οι ετικέτες **<servlet>** και **<servlet-mapping>**. Στο παράδειγμα μας το κοινό περιεχόμενο των δύο ετικετών **<servlet-name>** είναι η αυθαίρετα επιλεγμένη αλλά μοναδική (εντός του **web.xml**) αλφαριθμητική ακολουθία HelloServletByNNK. Η ουσία βρίσκεται στην ετικέτα **<servlet-class>** (που φωλιάζει στην **<servlet>**) και **<url-pattern>** (που φωλιάζει στην **<servlet-mapping>**). Το περιεχόμενο της **<servlet-class>** είναι το όνομα του αρχείου της εκτελέσιμης κλάσης και το περιεχόμενο της **<url-pattern>** είναι το διαθέσιμο ψευδώνυμο που θα διατίθεται στους χρήστες. Στο παράδειγμά μας η εκτελέσιμη κλάση είναι η HelloServlet.class (για αυτό και η ετικέτα servlet-class>HelloServlet **</servlet-class>**) και το ψευδώνυμο για τους χρήστες είναι hello_servlet (για αυτό και η ετικέτα <url-pattern>/hello_servlet**</url-pattern>**). Αν είχαμε δύο εφαρμογές (δύο servlets) το αρχείο **web.xml** θα ήταν (ότι προσθέσαμε είναι με ετικέτες σε άλλο χρώμα) όπως παρακάτω:

```
<web-app>

   <servlet>
      <servlet-name>HelloServletByNNK</servlet-name>
      <servlet-class>HelloServlet</servlet-class>
   </servlet>

   <servlet>
      <servlet-name>ArbitraryName2</servlet-name>
      <servlet-class>ClassFileName2</servlet-class>
   </servlet>

   <servlet-mapping>
      <servlet-name>HelloServletByNNK</servlet-name>
      <url-pattern>/hello_servlet</url-pattern>
   </servlet-mapping>

<servlet-mapping>
      <servlet-name>ArbitraryName2</servlet-name>
```

```
      <url-pattern>/PseudoName2</url-pattern>
   </servlet-mapping>

</web-app>
```

Θα επανέλθουμε παρακάτω για να δούμε λεπτομερέστερα τη δομή του web.xml.