I file WAR

- WAR è una contrazione di Web ARchive.
- I file WAR consentono di archiviare un'intera applicazione web in un unico file.
- Tomcat fornisce (tramite il Tomcat Manager) un'interfaccia grafica per effettuare il deployment di un file WAR direttamente dal browser.

Creazione di un file WAR

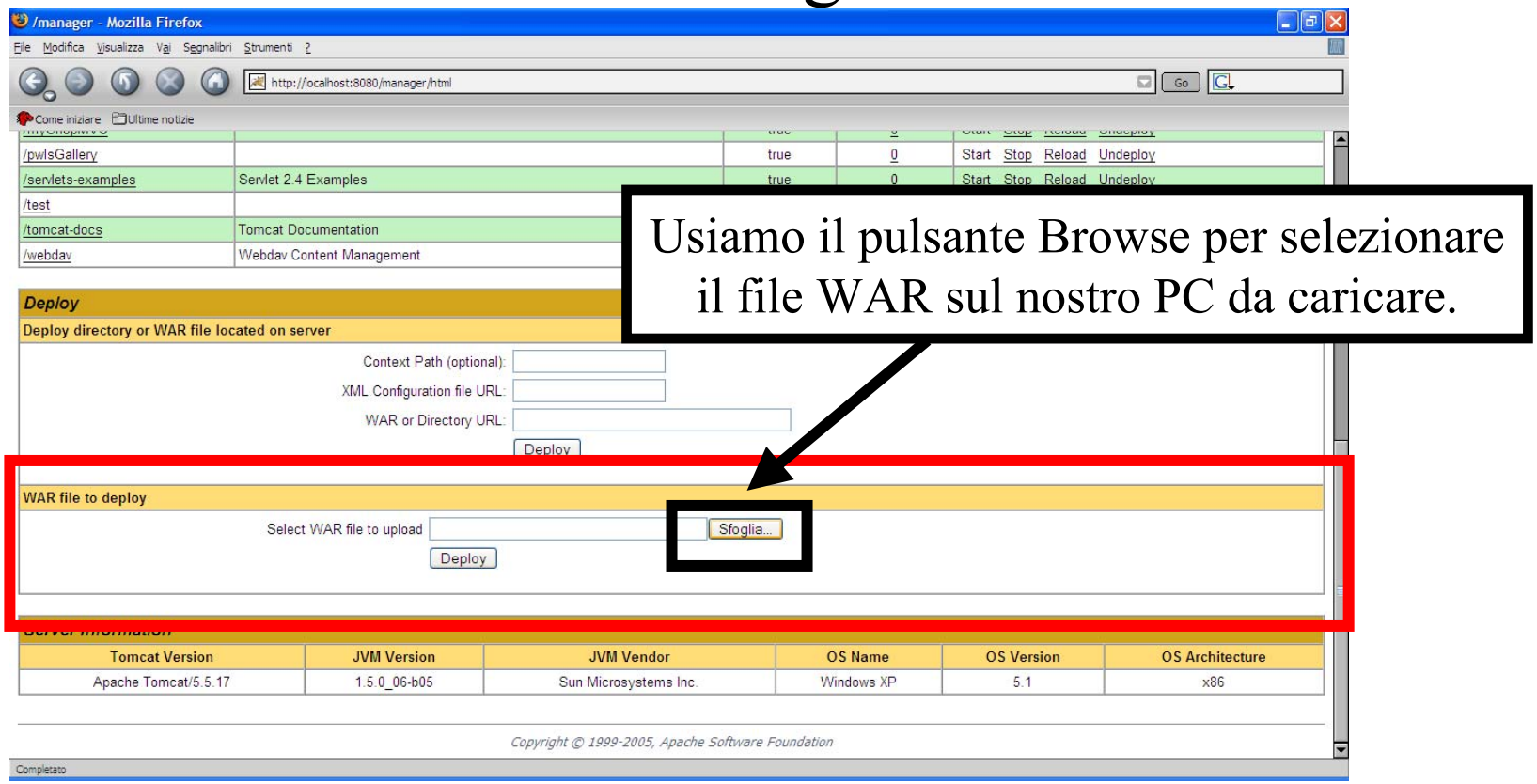
- Dalla directory radice dell'applicazione web che si vuole archiviare, digitare il comando seguente:

  `jar –cvf <nome-file>.war *`

- In seguito al comando precedente tutti i file dell'applicazione saranno "impacchettati" nel file **`<nome-file>.war`**.

Dispiegamento di un file WAR

- Usiamo il Tomcat Manager:



Usiamo il pulsante Browse per selezionare il file WAR sul nostro PC da caricare.

Dispiegamento di un file WAR

- In seguito al dispiegamento di un file WAR (dopo aver premuto Deploy) tramite il Tomcat Manager avviene quanto segue:
  - **`<nome-file>.war`** viene copiato in **`$TOMCAT_HOME/webapps`**;
  - l'archivio viene scompattato in una directory con nome **`<nome-file>`**;
  - viene avviata l'applicazione **`<nome-file>`**.
- Affinché il dispiegamento vada a buon fine non deve esserci un'altra applicazione con nome **`<nome-file>`**.

Servlet con classi ausiliarie

- Raramente un'applicazione web è composta soltanto da servlet.
- Spesso è utile/necessario far ricorso a classi esterne (eventualmente in altri package).
- Le classi ausiliarie possono essere archiviate in un file **`jar`** (da mettere nella sottodirectory **`lib`** della directory **`WEB-INF`** dell'applicazione che ne fa uso).

Un esempio riassuntivo: due servlet per gestire il login

- Come esempio riassuntivo, realizzeremo due servlet per gestire il login, logout di un'area riservata.
- Per semplicità gli account degli utenti saranno memorizzati in un file di testo (in applicazioni reali si dovrebbe usare un database).
- Utilizzeremo sessioni, I/O su file, registrazione degli errori nei file di log di Tomcat e metteremo una classe ausiliaria in un file separato.

Formato del file `utenti.txt`

- Le informazioni sugli account utente sono memorizzate utilizzando il seguente formato:

  `pippo&pluto&Mario Rossi`

  `paperino&paolino&Gianni Verdi`
- Per ogni linea compaiono nome utente, password e nome-cognome separati dal carattere **`'&'`**.

La classe ausiliaria

Questa classe viene memorizzata nel file **`Utente.java`** nella stessa directory degli altri sorgenti che la utilizzano:

```
public class Utente {
        String nomeUtente;
        String password;
        String nomeCognome;
        // costruttore:
        public Utente(String s1, String s2, String s3) {
                nomeUtente = s1;
                password = s2;
                nomeCognome = s3;
        }
        // metodo per effettuare il logout:
        public void Logout() {
                nomeUtente = null;
                password = null;
                nomeCognome = null;
        }
}
```

La servlet Login (I)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Login extends HttpServlet {
   private String path;
   private boolean stampaMessaggio; // flag: se vale true
            // provoca la stampa di un messaggio d'errore.
   public void init(ServletConfig conf) throws
   ServletException {
        super.init(conf);
        path=getServletContext().getRealPath("")+
             getInitParameter("nomeFile");
        stampaMessaggio = false;
   }
```

path memorizza il percorso al file degli utenti.

Alla prima richiesta non stamperemo messaggi d'errore

La servlet Login (II)

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Area riservata - Login</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<STRONG>Inserisci i dati richiesti:</STRONG><BR>");
    out.println("<FORM METHOD=\"post\" ACTION=\"Login\">");
    out.println("Nome utente: <INPUT TYPE=\"text\" NAME=\"Utente\""+
                " SIZE=\"10\"><BR>");
    out.println("Password: <INPUT TYPE=\"password\" NAME=\"Password\""+
                " SIZE=\"10\"><BR>");
    out.println("<INPUT TYPE=\"submit\" NAME=\"Login\""+
                " VALUE=\"Invia &gt;&gt;\">");
    out.println("<INPUT TYPE=\"reset\" VALUE=\"Annulla\">");
    out.println("</FORM>");
    ...
```

Form per l'inserimento di nome utente e password

La servlet Login (III)

```
    ...
    if(stampaMessaggio) {
       out.println("<BR><STRONG><FONT COLOR=\"red\">"+
                   "Nome utente e/o password non sono"+
                   " corretti!</FONT></STRONG>");
       stampaMessaggio = false;
    }

    out.println("</BODY></HTML>");
}

// Fine del metodo doGet().
```

Nel caso in cui il login non sia effettuato correttamente, la variabile **`stampaMessaggio`** viene impostata a **`true`** per visualizzare un messaggio d'errore.

La servlet Login (IV)

Recupero dei parametri del form

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
  throws ServletException, IOException {
  String utente = req.getParameter("Utente");
  String password = req.getParameter("Password");
  boolean autenticato = false;
  stampaMessaggio = true;
  try {
    BufferedReader input =
                    new BufferedReader(new FileReader(path));
    String linea = null;
    while((linea = input.readLine()) != null) {
      String[] dati = linea.split("&");

      if(utente.equals(dati[0]) && password.equals(dati[1])) {
        autenticato = true;
        stampaMessaggio = false;
        ...
```

Controllo della correttezza del login

La servlet Login (V)

```
                HttpSession s = req.getSession(true);
                Utente u = (Utente)s.getAttribute("Login.utente");

                if(u == null) { // creazione nuovo oggetto
                        u = new Utente(dati[0], dati[1], dati[2]);
                }
                else  { // aggiornamento dei dati
                        u.nomeUtente = dati[0];
                        u.password = dati[1];
                        u.nomeCognome = dati[2];
                }
                // memorizzazione dell'oggetto nella sessione
                s.setAttribute("Login.utente", u);
                break;
            }
        }
        input.close(); // chiusura del flusso di input
}
```

La servlet Login (VI)

```
    catch(IOException e) {
        log("Login Servlet: errore nella lettura del file "
            +path+" (dettagli: "+e.toString()+")");
    }

    if(autenticato) // login corretto -> area riservata
        res.sendRedirect("AreaRiservata");
    else // login non corretto: ripresento il form
        doGet(req, res);
  }

}
```

AreaRiservata (I)

La servlet AreaRiservata controlla se l'utente ha effettuato correttamente il login (cercando un oggetto di tipo Utente nella sessione attiva e controllando che sia inizializzato correttamente).

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AreaRiservata extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
              throws ServletException, IOException {
    HttpSession s = req.getSession(false);
```

AreaRiservata (II)

```
if(s != null) { // controllo se esiste una sessione attiva
  Utente u = (Utente)s.getAttribute("Login.utente");
  if(u != null) { // esiste un oggetto di tipo Utente nella sessione?
    if(u.nomeUtente != null) { // l'utente ha effettuato il login?
      res.setHeader("Cache-Control","no-cache"); // browser HTTP 1.1
      res.setHeader("Pragma","no-cache"); // browser HTTP 1.0
      res.setDateHeader("Expires",0);     // impedisce caching al proxy
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<HTML>");
      out.println("<HEAD><TITLE>Area riservata - Login</TITLE></HEAD>");
      out.println("<BODY>");
      out.println("<STRONG>Benvenuto "+u.nomeCognome+"!</STRONG>");
      out.println("<FORM METHOD=\"post\" ACTION=\"AreaRiservata\">");
      out.println("<INPUT TYPE=\"submit\" NAME=\"Logout\""+
                  " VALUE=\"Logout &gt;&gt;\">");
      out.println("</FORM>");
      out.println("</BODY></HTML>");
```

AreaRiservata (III)

```
        } else res.sendRedirect("Login"); // l'utente non ha effettuato il login
      } else res.sendRedirect("Login");   // non esiste un oggetto di tipo Utente
                                          // nella sessione
    } else res.sendRedirect("Login");     // non esiste una sessione attiva
  }
  public void doPost(HttpServletRequest req, HttpServletResponse res)
              throws ServletException, IOException {
    // recupero della sessione attiva:
    HttpSession s = req.getSession(false);
    // recupero dell'oggetto di tipo Utente:
    Utente u = (Utente)s.getAttribute("Login.utente");
    // logout:
    u.Logout();
    // memorizzazione dell'oggetto nella sessione:
    s.setAttribute("Login.utente", u);
    // redirezione al form di login:
    res.sendRedirect("Login");
  }
}
```

Modifiche da apportare al file `web.xml` (I)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
 ...
 <servlet>
   <servlet-name>Login</servlet-name>
   <servlet-class>Login</servlet-class>
   <init-param>
     <param-name>nomeFile</param-name>
     <param-value>/WEB-INF/data/utenti.txt</param-value>
   </init-param>
 </servlet>
 <servlet-mapping>
   <servlet-name>Login</servlet-name>
   <url-pattern>/servlet/Login</url-pattern>
 </servlet-mapping>
  ...
```

Modifiche da apportare al file **web.xml** (II)

```
...
<servlet>
    <servlet-name>AreaRiservata</servlet-name>
    <servlet-class>AreaRiservata</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AreaRiservata</servlet-name>
    <url-pattern>/servlet/AreaRiservata</url-pattern>
</servlet-mapping>

</web-app>
```

Modifiche da apportare al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>
<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/Login">Interfaccia di Login
           ad un'area riservata</A>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Impedire il caching di una pagina

- A volte è opportuno impedire il caching di una pagina generata dinamicamente, per evitare problemi come:
  - la visualizzazione di una risorsa protetta da un meccanismo di login basato su form, quando è stato effettuato il logout (premendo il pulsante aggiorna del browser ci si accorge che la pagina era rimasta in cache).
- Gli header corretti da aggiungere alla risposta HTTP sono i seguenti:
  - **`Cache-Control, no-cache`** (per browser compatibili con HTTP 1.1)
  - **`Pragma, no-cache`** (per browser compatibili con HTTP 1.0)
  - **`Expires, 0`** (per impedire il caching ai proxy server)

Esercizio

- Aggiungere alla servlet Login una funzionalità che permetta di registrare su file i tentativi errati di login.
- In particolare registrare la data e l'ora del tentativo e il nome utente e la password usati.