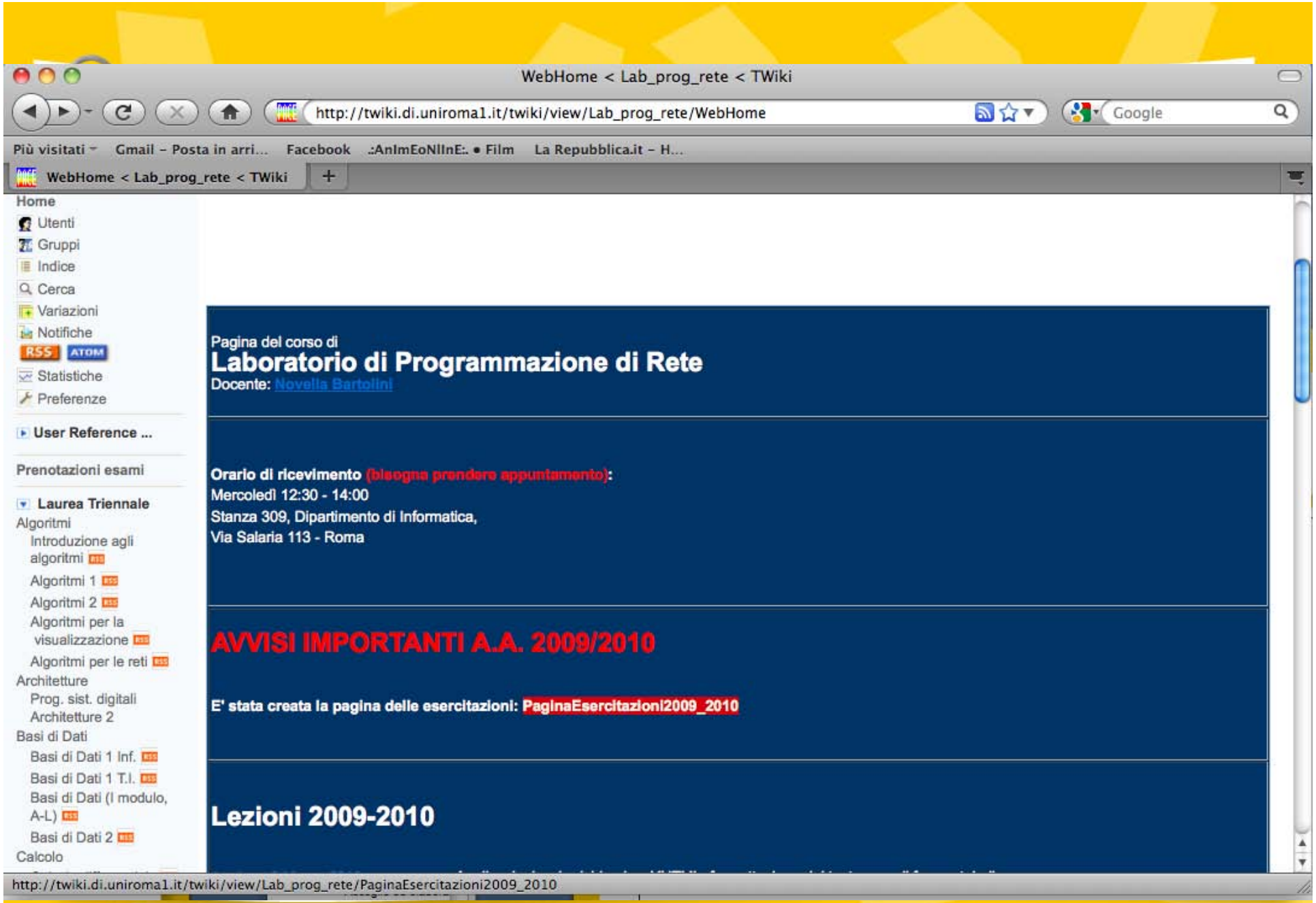
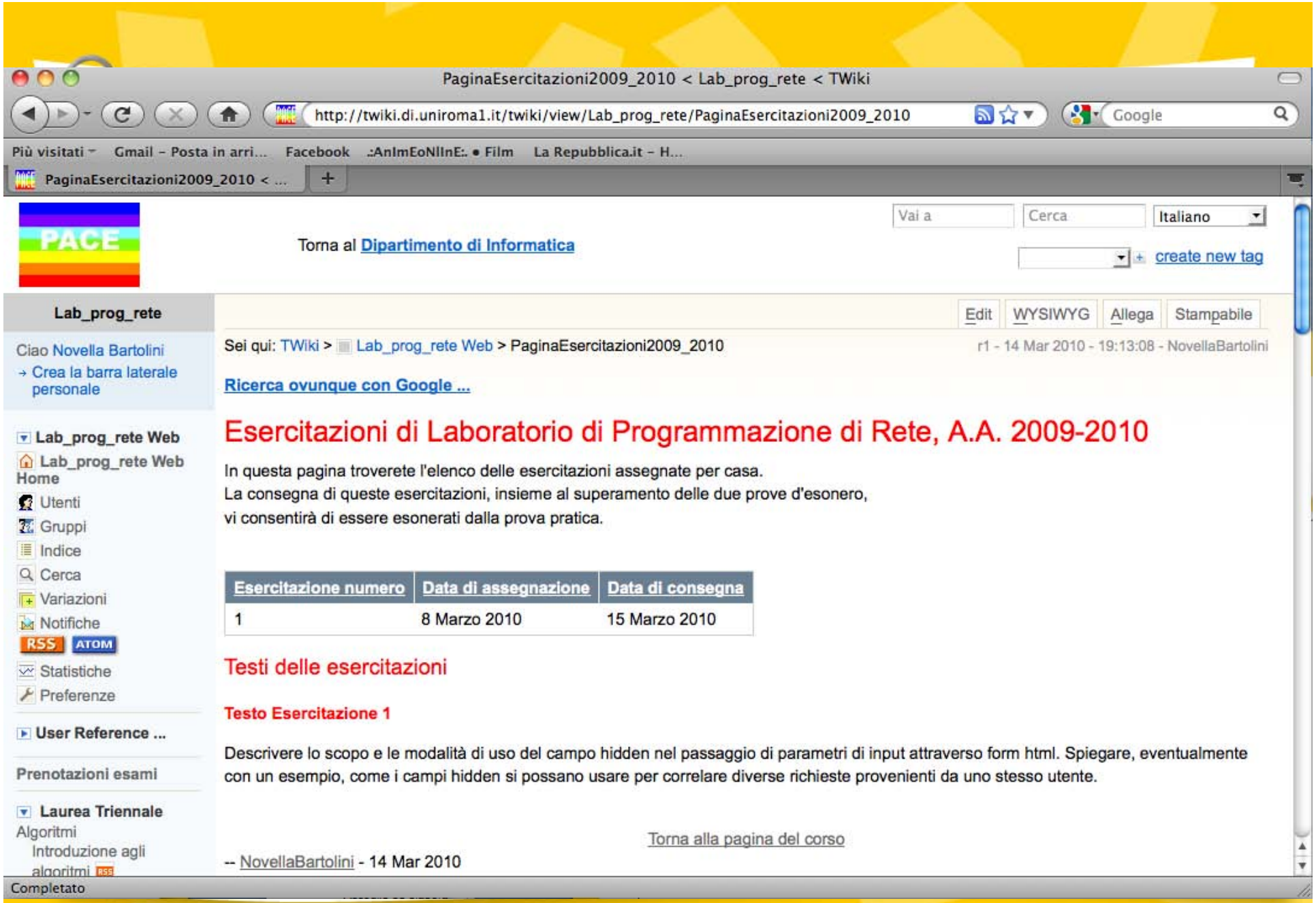
Laboratorio di Programmazione di Rete

Lezione del 15 Marzo 2010

Docente: Novella Bartolini
Ricevimento: Mercoledì ore 12:30-14:00
Via Salaria 113, terzo piano, stanza 309
Email: bartolini@di.uniroma1.it


WebHome < Lab_prog_rete < TWiki
http://twiki.di.uniroma1.it/twiki/view/Lab_prog_rete/WebHome
Google
Più visitati
Gmail - Posta in arri...
Facebook
:AnImEoNlInE:. • Film
La Repubblica.it - H...
WebHome < Lab_prog_rete < TWiki
Home
Utenti
Gruppi
Indice
Cerca
Variazioni
Notifiche
RSS
ATOM
Statistiche
Preferenze
User Reference ...
Prenotazioni esami
Laurea Triennale
Algoritmi
Introduzione agli algoritmi
Algoritmi 1
Algoritmi 2
Algoritmi per la visualizzazione
Algoritmi per le reti
Architetture
Prog. sist. digitali
Architetture 2
Basi di Dati
Basi di Dati 1 Inf.
Basi di Dati 1 T.I.
Basi di Dati (I modulo, A-L)
Basi di Dati 2
Calcolo
Pagina del corso di
Laboratorio di Programmazione di Rete
Docente: Novella Bartolini
Orario di ricevimento (bisogna prendere appuntamento):
Mercoledì 12:30 - 14:00
Stanza 309, Dipartimento di Informatica,
Via Salaria 113 - Roma
AVVISI IMPORTANTI A.A. 2009/2010
E' stata creata la pagina delle esercitazioni: PaginaEsercitazioni2009_2010
Lezioni 2009-2010
http://twiki.di.uniroma1.it/twiki/view/Lab_prog_rete/PaginaEsercitazioni2009_2010


PaginaEsercitazioni2009_2010 < Lab_prog_rete < TWiki
http://twiki.di.uniroma1.it/twiki/view/Lab_prog_rete/PaginaEsercitazioni2009_2010
Google
Più visitati
Gmail - Posta in arri...
Facebook
:AnImEoNlInE: • Film
La Repubblica.it - H...
PaginaEsercitazioni2009_2010 < ...
PACE
Torna al Dipartimento di Informatica
Vai a
Cerca
Italiano
create new tag
Lab_prog_rete
Edit
WYSIWYG
Allega
Stampabile
Ciao Novella Bartolini
→ Crea la barra laterale personale
Sei qui: TWiki > Lab_prog_rete Web > PaginaEsercitazioni2009_2010
r1 - 14 Mar 2010 - 19:13:08 - NovellaBartolini
Lab_prog_rete Web
Lab_prog_rete Web Home
Utenti
Gruppi
Indice
Cerca
Variazioni
Notifiche
RSS
ATOM
Statistiche
Preferenze
User Reference ...
Prenotazioni esami
Laurea Triennale
Algoritmi
Introduzione agli algoritmi
Ricerca ovunque con Google ...
Esercitazioni di Laboratorio di Programmazione di Rete, A.A. 2009-2010
In questa pagina troverete l'elenco delle esercitazioni assegnate per casa.
La consegna di queste esercitazioni, insieme al superamento delle due prove d'esonero,
vi consentirà di essere esonerati dalla prova pratica.
Esercitazione numero
Data di assegnazione
Data di consegna
1
8 Marzo 2010
15 Marzo 2010
Testi delle esercitazioni
Testo Esercitazione 1
Descrivere lo scopo e le modalità di uso del campo hidden nel passaggio di parametri di input attraverso form html. Spiegare, eventualmente con un esempio, come i campi hidden si possano usare per correlare diverse richieste provenienti da uno stesso utente.
Torna alla pagina del corso
-- NovellaBartolini - 14 Mar 2010
Completato

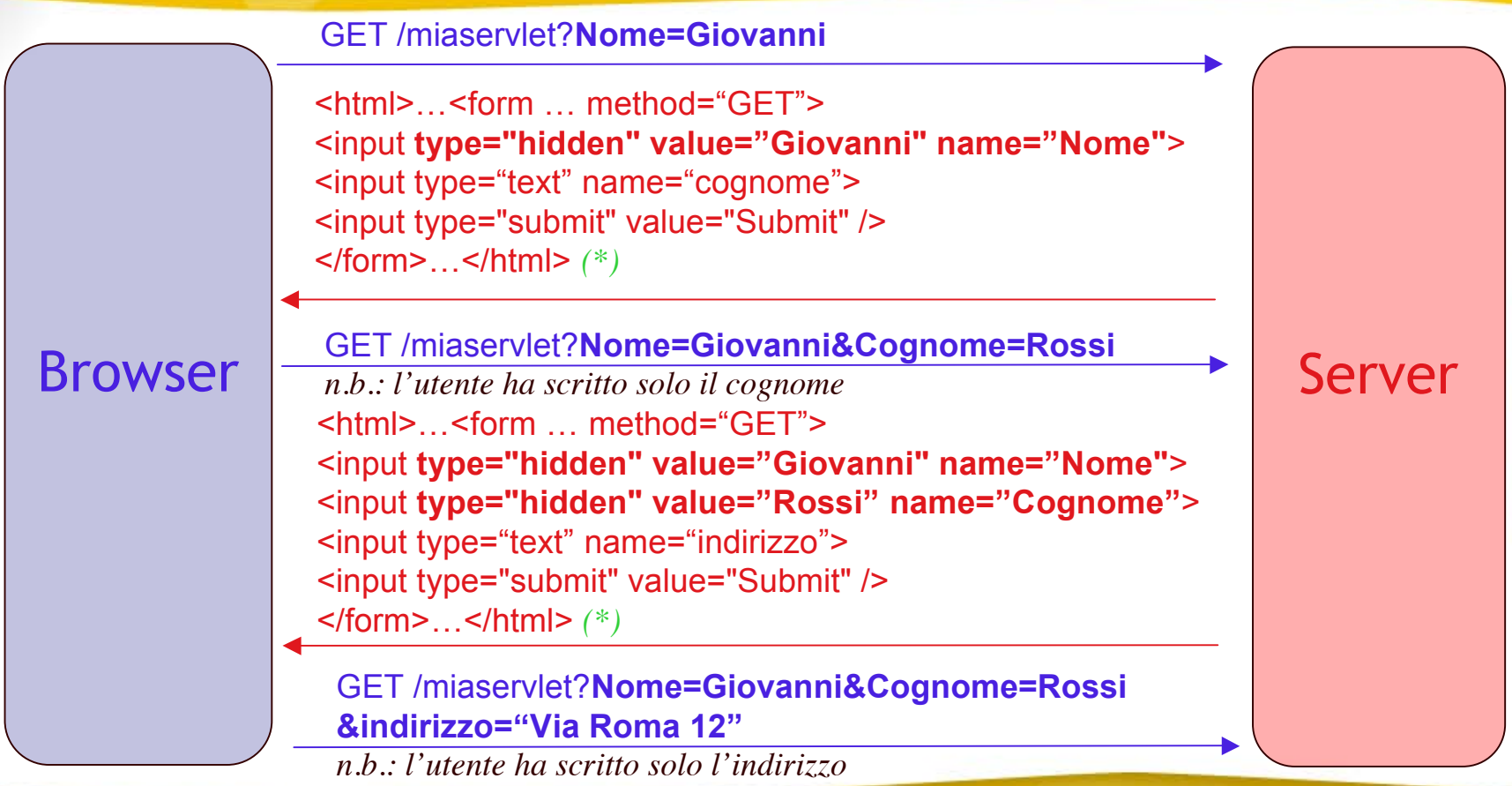
Caratteristiche del protocollo HTTP

E' il protocollo per il trasferimento di iper-testi (HyperText Transfer Protocol)

- Corrisponde al livello applicativo (stack iso/osi)
  - Presuppone un protocollo (TCP) orientato alla connessione
- Meccanismo di Richiesta/Risposta (Client/Server)
- E' possibile un trasferimento bi-direzionale di informazioni
  - il client può inviare informazioni a un server tramite un form, il server invia (di solito) pagine web al client
- Basato sul meccanismo di naming degli URI
- **Senza stato**
  - **Ogni richiesta HTTP è autonoma, il server non tiene una cronologia delle richieste**

Gestione della sessione tramite parametri hidden

Gestione della sessione tramite parametri hidden



(*) *Il testo della pagina html viene generato dinamicamente dalla servlet in funzione dei parametri ricevuti nella richiesta, sia che fossero hidden sia che fossero visibili nel form*

Interazione Client - Web Server

- Sempre basata sul protocollo HTTP indipendentemente dalla soluzione adottata dal lato del server
- L'URL oggetto della richiesta è usato per selezionare la risorsa lato server che si vuole usare

Protocollo HTTP

- Protocollo stateless
  - Ad ogni richiesta del client segue una risposta del server. Nessuna correlazione tra richieste successive.
- Quando un client invia una richiesta al server, specifica un comando
- La prima linea della richiesta contiene il nome del comando, un URL e la versione del protocollo in uso:

```
GET /main.html HTTP/1.0
```

Protocollo HTTP (segue)

- Alla richiesta vengono appese informazioni opzionali
  - versione del browser,
  - i tipi di file che possono essere elaborati…

```
User-Agent: Mozilla/4.0 (compatible; MIE 4.0; Windows 95)
Accept: image/gif, image/jpeg, text/*, */*
```

Protocollo HTTP (segue)

- Il server elabora la richiesta e spedisce una risposta, specificando la versione del protocollo e uno status code (header della risposta):

```
HTTP/1.0 200 OK
```

- Altre informazioni inviate nell'header della risposta
  - Server software
  - Content-type della risposta

Protocollo HTTP: GET & POST

- GET e POST sono i comandi HTTP utilizzati più frequentemente
- Progettati inizialmente per scopi diversi
  - GETting information (read)
  - POSTing information (write)

- GET: informazioni utili per formulare la richiesta appese all'URL
- POST: informazioni incluse nel corpo della richiesta

Protocollo HTTP: GET

- I parametri della richiesta sono visibili
- La lunghezza della query string è limitata dal browser
  - Molti browser non consentono l'uso di query string di più di 240 caratteri
- La richiesta può essere inserita nei bookmark e ripetuta quante volte si vuole
- Si può ripetere la richiesta effettuando il "reload" dal browser

Protocollo HTTP: POST

- I parametri della richiesta non sono visibili all'utente
- La quantità di dati che si possono inviare è illimitata (anche megabytes)
- Le richieste di POST non possono essere inserite nei bookmark, né spedite via email o ricaricate.

Protocollo HTTP: GET e POST

- La distinzione funzionale con cui i metodi erano stati progettati si è persa MA persistono differenze sostanziali
- GET
  - parametri visibili
  - lista breve
  - inseribile nei bookmark e ripetibile
- POST:
  - parametri nascosti
  - lunghezza illimitata
  - non inseribile nei bookmark e non ripetibile
- Non usare richieste di GET per gestire ordini o aggiornamenti di un database o trasferire password

Sviluppo di applicazioni di rete

- Descrizione del paradigma client-server
- Implicazioni del paradigma client-server nel funzionamento dei sistemi web
  - Funzionamento di un web server
  - Generazione di pagine dinamiche con tecnologie lato client e lato server

Tecnologie lato client o lato server?

lato client all'interno di Applet, o attraverso metodi di scripting dal lato client (JavaScript), o in applicazioni stand-alone.

Richiedono il supporto del client

lato server associato a Servlet, agli Enterprise JavaBean, agli Agenti, alle API ed ai servizi di Transaction Management, agli Application Server ed ai protocolli di programmazione distribuita (es. CORBA)

Non richiedono alcun supporto da parte del client

Perché soluzioni lato server

- Soluzioni lato client
  - problemi di prestazioni e di portabilità
- Soluzioni lato server
  - accesso a informazioni che sono disponibili esclusivamente dal lato del server (es. database etc.)
  - minimi requisiti in termini di capacità di calcolo e storage dal lato del client (è il server a fare il grosso del lavoro)
  - il client non deve avere altro che il browser per interpretare le pagine html

Soluzioni lato server

- CGI (un processo - una richiesta a programma cgi)
  - Il web server inoltra le richieste a programmi esterni
  - L'output generato dal programma esterno viene intercettato dal web server e spedito al client nella forma di una pagina html (generata dinamicamente dal programma cgi)
  - Per ogni richiesta del client, il server deve creare un nuovo processo
    - Forte limitazione sul numero di richieste che possono essere gestite in parallelo
- Fast-CGI (un processo - un programma cgi)
  - Soffre ancora del problema della proliferazione dei processi, uno per ogni programma CGI

Altre soluzioni lato server

- ASP
  - Consente l'inserimento di codice all'interno delle pagine HTML (HTML-embedded), che viene eseguito dal server
  - E' ottimizzato per la generazione di piccole porzioni di contenuto dinamico
- PHP
  - Consente l'uso di codice "HTML-embedded"
  - Uso di un linguaggio interamente nuovo e poca disponibilità di API rispetto a JSP e Servlet

Servlet

- Programma applicativo che viene eseguito da un server
  - Accoglie ed elabora richieste provenienti dal client (attraverso comandi http: POST o GET, ma anche attraverso altri protocolli)
  - Produce una risposta HTTP contenente codice HTML generato dinamicamente
- Molto diffuse per applicazioni che fanno uso di database
  - Thin clients (client molto semplici che richiedono supporto minimo)
  - Meccanismo request/response

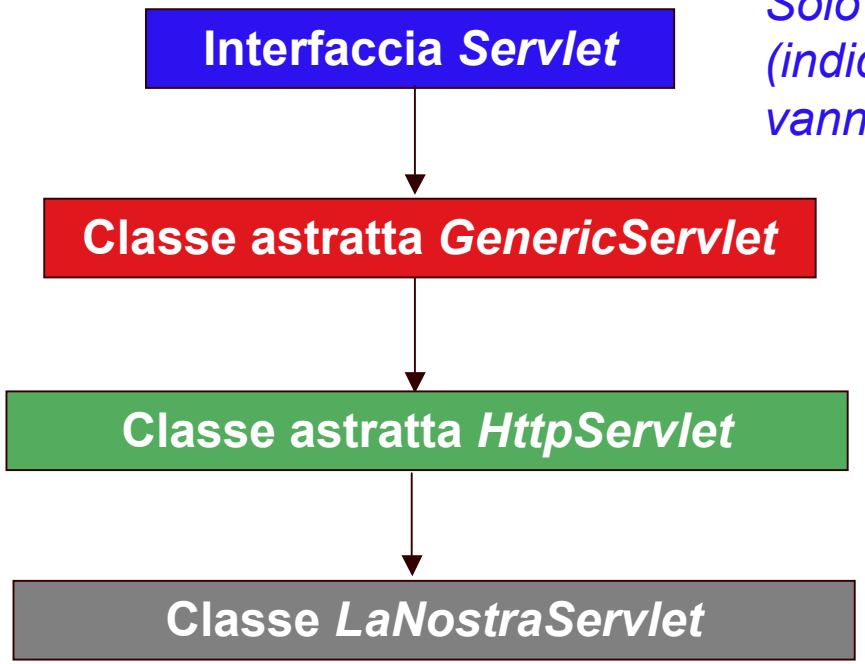
Servlet e Servlet Container

- Una servlet è una classe Java (che implementa l'interfaccia Servlet).
- Un servlet container può ospitare più servlet (con relativi alias).
- Quando una servlet viene invocata per la prima volta, il servlet engine genera un thread Java che inizializza l'oggetto Servlet.
  - Questo *persiste* per tutta la durata del processo relativo al servlet container (salvo esplicita de-allocazione).
- Ogni servlet è un thread all'interno del Servlet Container (vs CGI dove viene eseguito un processo esterno)

Possibili usi di una Servlet

- Supportare richieste multiple in modo concorrente, come per esempio conferenze on-line, servizi di comunicazione
- Aggiornare, eliminare o consultare dati contenuti in un DB remoto tramite protocollo TCP/IP
- Applicazioni basate sull'autenticazione degli utenti, gestione di sessioni di navigazione con creazione di oggetti persistenti
- Ottimizzazione delle prestazioni tramite redirezione di richieste ad altre Servlet in altri server, allo scopo di bilanciare il carico di lavoro

Servlet API



Solo interfaccia (indica quali metodi vanno implementati)

implements Servlet (indipendente dal protocollo in uso; non istanziabile)

extends GenericServlet (si usa solo con il protocollo HTTP; non istanziabile)

extends HttpServlet (istanziabile)

Servlet

- Le servlet possono essere utilizzate qualunque sia il servizio espletato dal server, ovvero qualunque sia il protocollo di interazione client/server: es HTTP, FTP
- Nella sua forma più generale, una servlet è un'estensione di una classe javax.servlet.GenericServlet che implementa l'interfaccia javax.servlet.Servlet
- Le servlet usate nel web sono estensioni della classe javax.servlet.http.HttpServlet che implementa l'interfaccia javax.servlet.Servlet

Interfaccia Servlet

- Interfaccia Servlet (package *javax.servlet*)
  - Tutte le servlet devono implementare questa interfaccia
  - Tutti i metodi dell'interfaccia Servlet vengono invocati dal servlet container secondo un prefissato *ciclo di vita*
    - (vi ricordate come funziona con le applet? Init-start-paint-stop-destroy vengono sempre chiamati in sequenza dal browser per eseguire una applet)

Ciclo di vita di una servlet

- Caricamento della servlet in memoria
- Il container delle servlet invoca il metodo init
  - Solo in questo momento la servlet è in grado di rispondere alla prima richiesta
  - Inizializzazione delle variabili globali
  - Invocato una sola volta
- Il metodo service gestisce le richieste
  - Riceve la richiesta
  - Elabora la richiesta
  - Confeziona un oggetto risposta
  - Invocato ad ogni richiesta del client
- Il metodo destroy rilascia le risorse allocate dalla servlet quando il container la termina

Classi astratte per definire le Servlet

- Esistono due classi astratte che implementano l'interfaccia *Servlet*
  - **GenericServlet (package javax.servlet)**
  - **HttpServlet (package javax.servlet.http)** (quest'ultima implementa l'interfaccia servlet indirettamente, estendendo GenericServlet)
- Queste classi forniscono l'implementazione di default di tutti i metodi dell'interfaccia *Servlet*
- Il metodo chiave è *service*:

  riceve gli oggetti ServletRequest e ServletResponse che forniscono accesso agli stream di i/o permettendo la ricezione e l'invio di informazioni al client

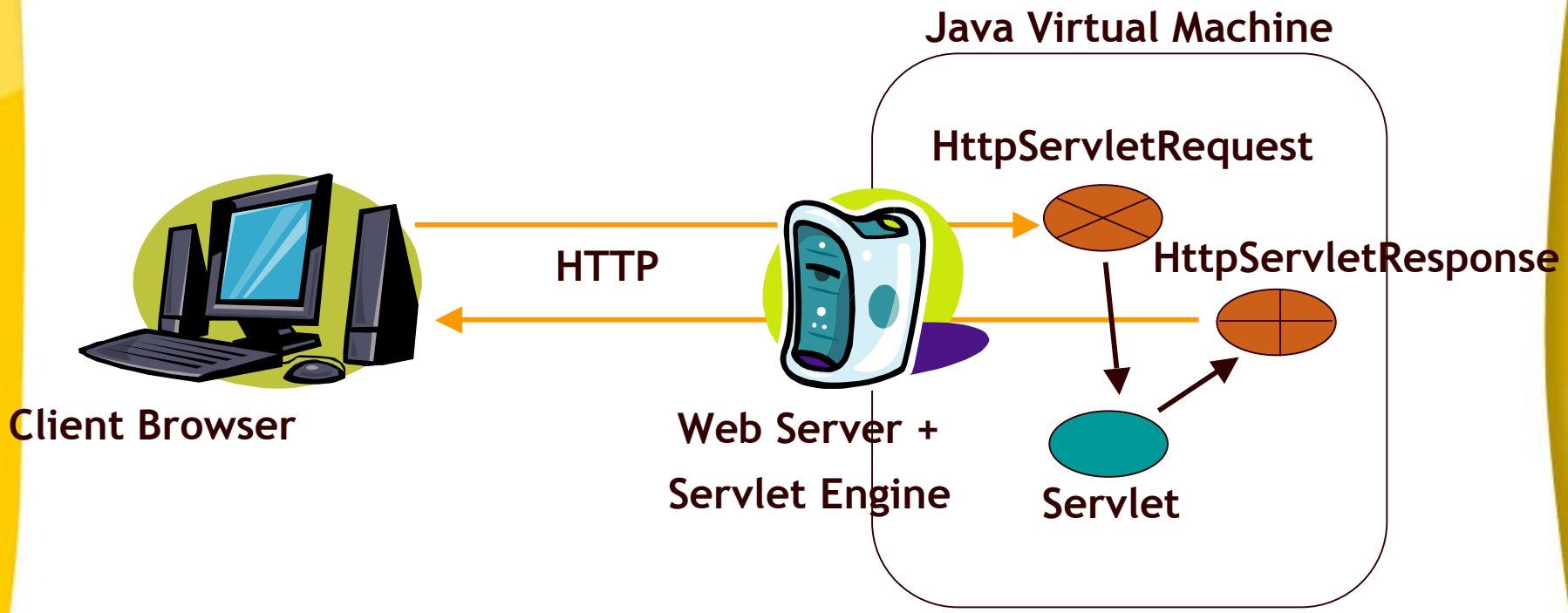
Metodi dell'interfaccia Servlet

Method	Description
void init(ServletConfig config)	
	The servlet container calls this method once during a servlet's execution cycle to initialize the servlet. The ServletConfig argument is supplied by the servlet container that executes the servlet.
ServletConfig getServletConfig()	
	This method returns a reference to an object that implements interface ServletConfig. This object provides access to the servlet's configuration information such as servlet initialization parameters and the servlet's ServletContext, which provides the servlet with access to its environment (i.e., the servlet container in which the servlet executes).
String getServletInfo()	
	This method is defined by a servlet programmer to return a string containing servlet information such as the servlet's author and version.
void service(ServletRequest request, ServletResponse response)	
	The servlet container calls this method to respond to a client request to the servlet.
void destroy()	
	This "cleanup" method is called when a servlet is terminated by its servlet container. Resources used by the servlet, such as an open file or an open database connection, should be deallocated here.

Funzionamento delle servlet HTTP

- Il server web riceve dal browser del client una richiesta HTTP GET o POST
- Il server web direziona la richiesta HTTP al motore servlet o servlet engine (pr. èngine) o servlet container
- Se la servlet non è ancora in memoria viene caricata e inizializzata dal servlet engine (esecuzione del metodo init)
- Il servlet engine incapsula la richiesta HTTP in una classe **HttpServletRequest** e la passa al metodo doPost o doGet della servlet
- La servlet risponde scrivendo il codice HTML nella **HttpServletResponse** che viene rimandata al web server e poi riconsegnata al client via HTTP

Servlet e Web Server



La classe HttpServlet

- Sovrascrive il metodo service della classe GenericServlet
- Prevede i metodi per rispondere alle richieste HTTP
  - GET
  - POST (ma anche HEAD, PUT, OPTIONS ecc.)
- Il metodo service() invoca i metodi doGet e doPost
  - Il metodo doGet() risponde alle richieste GET
  - Il metodo doPost() risponde alle richieste POST
    - Ricevono come parametri gli oggetti HttpServletRequest e HttpServletResponse
- Non implementare il metodo service se si usa questo tipo di servlet

Altri metodi della classe HttpServlet

Method	Description
doDelete	Called in response to an HTTP *delete* request. Such a request is normally used to delete a file from a server. This may not be available on some servers, because of its inherent security risks (e.g., the client could delete a fi le that is critical to the execution of the server or an application).
doHead	Called in response to an HTTP *head* request. Such a request is normally used when the client only wants the headers of a response, such as the content type and content length of the response.
doOptions	Called in response to an HTTP *options* request. This returns information to the client indicating the HTTP options supported by the server, such as the version of HTTP (1.0 or 1.1) and the request methods the server supports.
doPut	Called in response to an HTTP *put* request. Such a request is normally used to store a file on the server. This may not be available on some servers, because of its inherent security risks (e.g., the client could place an executable application on the serv er, which, if executed, could damage the server —perhaps by deleting critical files or occupying resources).
doTrace	Called in response to an HTTP *trace* request. Such a request is normally used for debugging. The implementation of this method automatically returns an HTML document to the client containing the request header information (data sent by the browser as part of the request).

Interfaccia HttpServletRequest

- Il servlet engine che esegue la servlet
  - Crea un oggetto **HttpServletRequest**
  - Lo passa al metodo **service** della servlet http
- L'oggetto **HttpServletRequest** contiene la richiesta del client
- Esistono molti metodi per estrarre informazioni dalla richiesta del client. Ne vediamo alcuni →

Interfaccia HttpServletRequest (Cont.)

Method	Description
String getParameter(String name)	
	Obtains the value of a parameter sent to the servlet as part of a get or post request. The name argument represents the parameter name.
Enumeration getParameterNames()	
	Returns the names of all the parameters sent to the servlet as part of a post request.
String[] getParameterValues(String name)	
	For a parameter with multiple values, this method returns an array of strings containing the values for a specified servlet parameter.
Cookie[] getCookies()	
	Returns an array of Cookie objects stored on the client by the server. Cookie objects can be used to uniquely identify clients to the servlet.
HttpSession getSession(boolean create)	
	Returns an HttpSession object associated with the client's current browsing session. This method can create an HttpSession object (true argument) if one does not already exist for the client. HttpSession objects are used in similar ways to Cookies for uniquely identifying clients.

Interfaccia HttpServletResponse

- Il servlet engine
  - Crea un oggetto HttpServletResponse
  - Lo passa al metodo service della servlet (attraverso i metodi doGet e doPost)
- Ogni chiamata ai metodi doGet e doPost riceve un oggetto che implementa l'interfaccia HttpServletResponse
- Esistono molti metodi che consentono la scrittura della risposta da inviare al client. Ne vediamo alcuni →

Interfaccia HttpServletResponse

Method	Description
void addCookie(Cookie cookie)	
	Used to add a Cookie to the header of the response to the client. The Cookie's maximum age and whether Cookies are enabled on the client determine if Cookies are stored on the client.
ServletOutputStream getOutputStream()	
	Obtains a byte-based output stream for sending binary data to the client.
PrintWriter getWriter()	
	Obtains a character-based output stream for sending text data to the client.
void setContentType(String type)	
	Specifies the MIME type of the response to the browser. The MIME type helps the browser determine how to display the data (or possibly what other application to execute to process the data). For example, MIME type "text/html" indicates that the response is an HTML document, so the browser displays the HTML page.

Servlet container

Jakarta project

- Progetto Jakarta (Apache Software Foundation)
- Implementazione di riferimento degli standard per la realizzazione di servlet e di Java Server Pages

- Obiettivo: *"[...] to provide commercial-quality server solutions based on the Java Platform that are developed in an open and cooperative fashion"*

Architetture per l'esecuzione di servlet

- Servlet container (o servlet engine)
  - E' il server che esegue una servlet
- Servlet e JSP sono supportate direttamente o attraverso plugin dalla maggior parte dei Web servers e application server
  - Apache TomCat
  - Sun ONE Application Server
  - Microsoft's Internet Information Server (IIS)
  - Apache HTTP Server + modulo JServ
  - BEA's WebLogic Application Server
  - IBM's WebSphere Application Server
  - World Wide Web Consortium's Jigsaw Web Server

Configurare il server Apache Tomcat (1)

1. Installare j2sdk
2. Definire la variabile PATH=c:\j2sdk---\bin (per poter trovare il compilatore)
3. Definire la variabile JAVAHOME=c:\j2sdk--- (perché tomcat trovi la jvm)
4. Installare il server TomCat

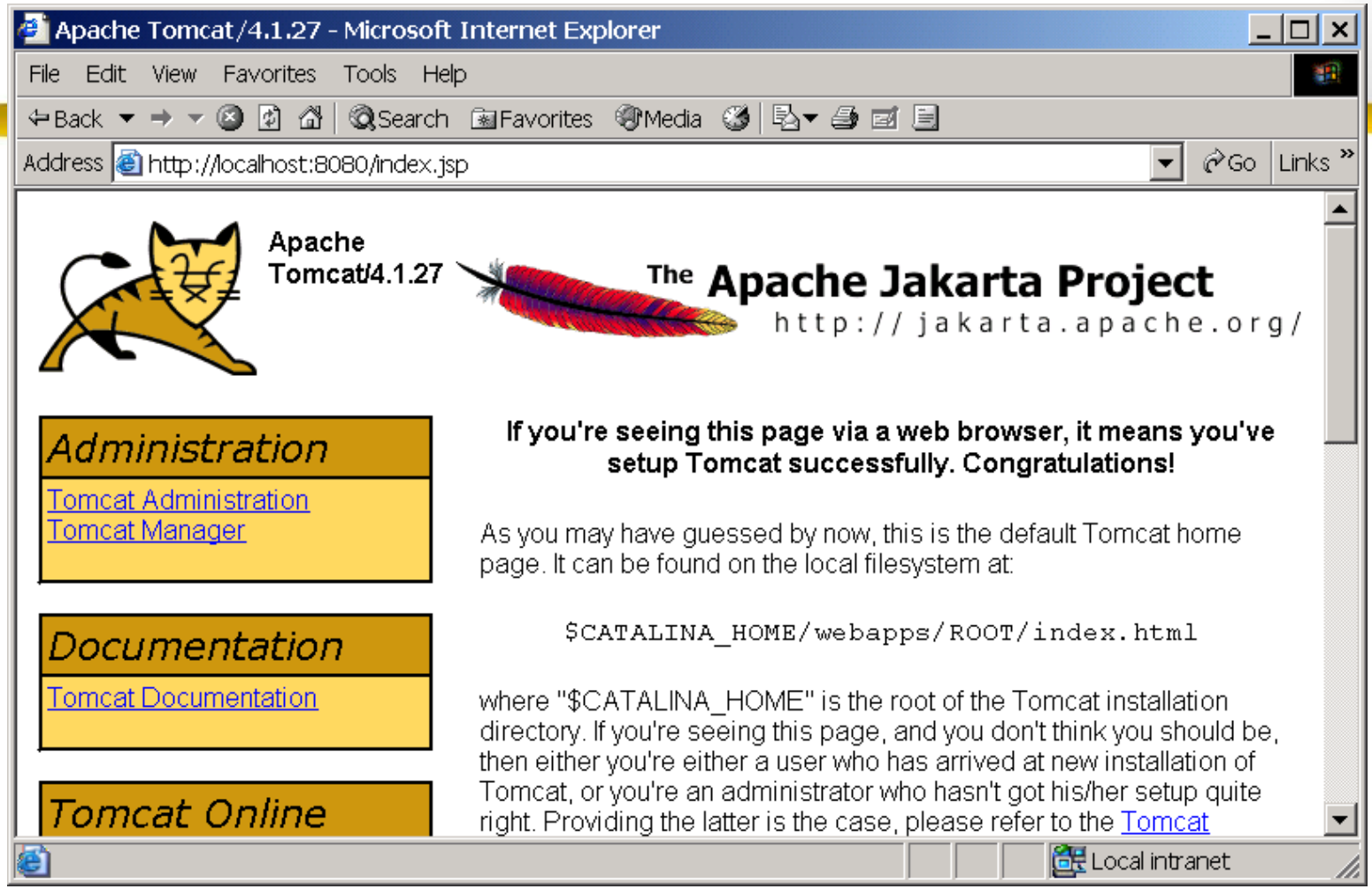
Configurare il server Apache Tomcat (2)

5. Definire la variabile CATALINA_HOME=c:\Programmi\Apache…\Tomcat---\

6. Definire la variabile CLASSPATH=c:\Programmi\Apache…\Tomcat---\common\lib\servlet-api.jar;c:\Programmi\Apache…\Tomcat---\common\lib\jsp-api.jar

7. TEST: Avviare il server Tomcat (startup), invocare il server da un browser all'indirizzo http://localhost:8080/

Configurare il server Apache Tomcat



Se Tomcat è installato correttamente dovreste vedere questa pagina all'indirizzo http://127.0.0.1:8080/