# Web

Tuesday, 28 March 2023 10:46

- Applicazioni web
  - Basato su internet (Infrastruttura di rete con base TCP-IP)
  - Web utilizza l'internet con il protocollo http
    - Creazione applicazioni aperte => Backend con un server
    - E' un formato a caratteri, quindi è molto lento siccome, es, 256 => Stringa
      però ci permette un cross Platform
    - Applicazioni scaricate e possono essere usate offline
      - Il web server fa le richieste ad un application server che detiene le a
        Che ha il compito di eseguire dei programmi a seconda del cliente.
        Che possono essere o programmi (compilati) o script (interpretati)
        Lati in comuni:
        - URL = naming globali
        - HTTP = Invocare programmi sul server come se fossero pagine

        Problemi compilati (c++):
        - Seguenza -> Dobbiamo avere un controllo di ciò che succede a
        - La communicazione tra web server ed application server avvie
          Attraverso la CGI
          Che identifica maniera univoca per creare processo, parametr

        Problemi script (python):
        - Meno velocità
        - Più veloce da scrivere
        - Non gestiscono GCI, gli interpreti lo fanno per noi
    - I/O viene eseguito con:
      - Form per invio al server
      - Le pagine HTML può essere usato per trasferire informazioni al clien
      - Scambio informazioni client<->server con JSON javascript
        E questo è possibile siccome HTML utilizzo un payload **MIME**
        Che generalizza html, xml, json, zip, jpeg
      - Il client è sempre quello che richiede al server
    - Possiamo avere delle conversazioni con I/O che però sono prive di stato (r
      - Per potere mantenere memoria si utilizzano
        Sessioni = Cookie + Campi nascosti = HTTPSession = ID
        -> Gestito da container => Map
        -> Servlet possono accedervi pe

        Perché usare HTTPSession e non un DataBase?

Perché usare HTTPSession e non un DataBase?
E' un oggetto, quindi è volatile
E' più veloce di un DataBase
- Senza non si possono mantenere informazioni tra un messaggio e l'a
identificare i clienti
- Cookie è diciamo un piccolo blocco di dati che ci permette di identif
-> Gestione di lavoro

- Usa interfaccia grafica (Browser)
Se e' un web senza interfaccia grafica, si chiama API = Application programming

- HTML
Noi definiamo solamente quali sono le semantiche del testo
Noi abbiamo 2 richieste principali: GET-POST
Queste 2 richieste hanno diversi modi per potere trasferire le informazioni:
  - GET = Query parameter = Ciò che c'è dopo il ? In un url
  - POST = Payload = un documento

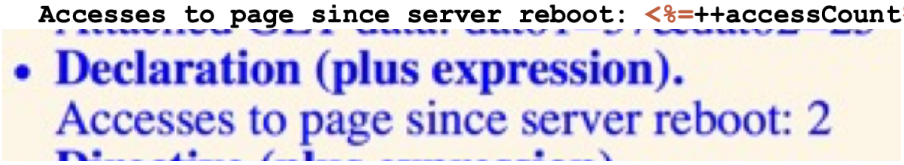
- Servlet
  - E' un componente gestito automaticamente da un container server
  Che è in grado di attivarle, disattivarle in base alle richieste del client
  - Semplice e standardizzato
  - Rigidità del modello
  - Mantengono uno stato in memoria e consentono interazione con altre ser
  - Ciclo di vita:
    - Si crea una servlet alla prima invocazione dal client e poi si tiene in r
    Quindi, ad ogni invocazione viene, preso un thread, assegnato quell
    E viene eseguito il metodo invocato
    - Ogni cliente generare una richiesta, assegnata ad un thread, che cor
    medesima servlet, perché?
      - Meno memoria
      - Meno gestione
      - Più persistenza
      - Possiamo condividere informazioni tra client
    - Terminazione
      - Quando non ci sono più thread
      - Quando un timeout specifico è scaduto
      Può scadere anche quando i thread sono attivi, ed in questo c
      Il metodo destroy() che cancella threads deve avvisare ai vari
  - Quando le richieste > num thread, i thread vengono aggiunti nella coda di
  E quando troppe richieste, viene rifiutata la richiesta → DDOS (Denial of se

- Spring / JAX-RS

  Le servlet sono pesanti, e richiedono tanto codice, quindi sono state create delle
  Una particolarità è la JSP:
  - Interazione tra container/serve avviene tramite pagine (che possono esse
  - Noi qui facciamo l'opposto delle servlet, all'interno dell'html abbiamo del
  - Ci permette di rendere html dinamico
  - Mettiamo il codice java dentro a <% %>
    - <%= → Espressioni e ritorniamo un valore
    - <%-- → Commenti
    - <%! → Dichiarazione di variabili
    - <% → Codice generico in cui possiamo aggiungerci altri codici
      - <% for (int i = 0; i < v.length; i++) { %>
        <tr><td> <%= v[i] %> </td></tr>
        <% } %>
      - <% out.println("Ciao!") %>

        PORCA - LE SLIDE SONO SBAGLIATE MA CHI CAZZO LE FA STE S

        ```
        <li><b>Declaration (plus expression).</b><br> <%!private int
          Accesses to page since server reboot: <%=++accessCount%> </
        ```

        Questo è sbagliato, SBAGLIATO, non può funzionare
        Perché? accessCount è inizializzato in page, quindi ogni volta c
        pagina viene sempre prima inizializzato a 0, e poi incrementat
        E quindi l'output sarà sempre 1, non può essere 2 siccome lo s

    - <%@ → Direttive, non influenzano la richiesta ma il compilatore
      - Page
        - Possiamon includere le librerie java
        - Buffer
        - Creazione di sessioni/no
      - Include
        - Possiamo includere altre pagine html/jsp
      - Taglib
        - Definiamo tag personalizzati
          Es.
          <%@ taglib %>
          <table:loop> … </table:loop>

    - <jsp:XXX attributi> </jsp:XXX> → Azioni
      - Forward

- Forward
  - Crea una servlet che invoca un'altra servlet
  - <jsp:forward page="login.jsp">
    <jsp:param name="user" value="Ale"/>
    </jsp:forward>
- Include
  - Richiede una richiesta ad una jsp e prende risultato
  - <jsp: include page="shotting.jsp"/>
- useBean
  - Localizzare una istanza (quindi variabili condivise (?)
  - <jsp:useBean id="cart" scope="session" class="Shoppin
    Id = nome
    Class = la classe, chiamata javaBean
    - Tutti i campi devono essere privati
    - Il costruttore non può avere parametri
    - Tutti gli accessi possono essere con
      - setXXX
      - getXXX
      - isXXX

    Possiamo anche usare Type anziché class
    Scope = Per quanto memorizzare
    - Application = Variabile globale
    - Session = Stesso client
    - Request = Solo alla singola richiesta
    - Page = Alla singola pagina
  - Es:
    <jsp:useBean id="user" class="com.Person" scope:"sess
    Qui tutti i valori sono default

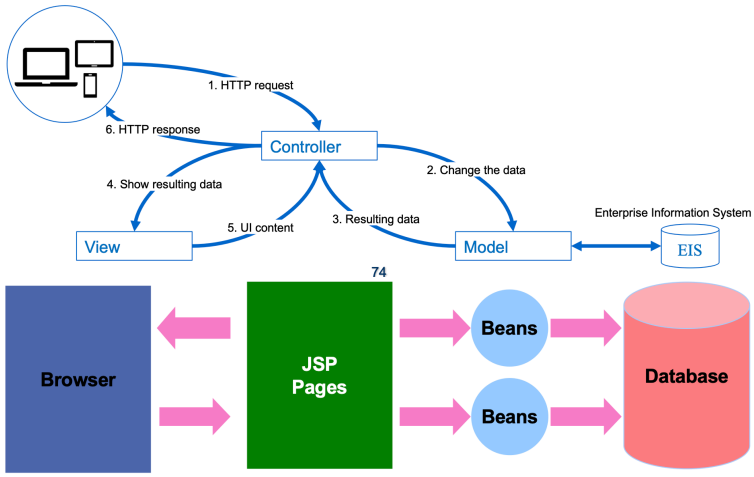
    Se vogliamo dei valori:
    <jsp:useBean id="user" class="com.Person" scope:"sess
    <% user.setDate() %>
    </>

    Si può fare anche con:
    <jsp:getProperty name="user" property="name" />
    (Ritorna un valore alla pagine, come se fosse <%=
    <jsp:setProperty name="user" property="name" value=

- Esistono variabili implicite già dichiarate:
  - Request, Response
  - Out, Page, exception
    Possiamo usare syncronized (page) per avere accesso mutua esclusi
    determinate sezioni del codice
  - pageContext, Application, Config
  - Session
    Questo funziona attraverso i cookie, quindi a seconda del cookie noi
    riferimento ad un differente session object
    E noi possiamo fare:
    - putValue → Salvare valori
    - getValue → Prendere valori

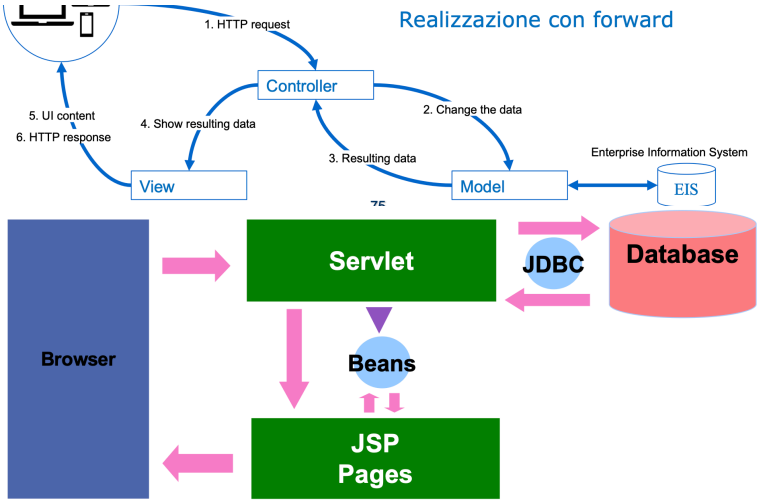
    Essa viene creata automaticamente ameno che non lo diciamo espr
    <%@ session="false" %>
- Noi vogliamo cercare di avere il numero minor possibile di funzioni java ne
  E da questo è nato MCV, cioè la separazione tra:
  - View
    Presentazione, view
  - Controler
    Azioni che controllano i model
  - Model
    I dati dell'applicazione e le operazioni

  Spiegazione:
  L'utente agisce con il controller, che cambia i model, ed i model magari è c
  database
  E una volta ritornato al controller, modifica il view
  E nel caso il view ritorna al controller per poi ritornare al client, si chiama




  Il problema è che, facendo così si decentralizza il lavoro
  Ed in più potrei essere costretto a creare tanti tag, quindi +complessi

  Invece nel caso la view và subito dal client, si chiama MCV2




Noi qui facciamo una fusione tra servlet e jsp

E quindi useremo tutte e due affinchè possiamo aveer una efficenza

Vantaggi:

- Chiara sperazione logica e rappresentazione
  - Cambiare view senza sminchiare le altre 2 parti
    => Posso fare evolvere i due separatamente, quindi teams
- Standard

Svantaggi:

- Sistema più complesso
- Concorrenza