ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Architetture del World Wide Web

Fabio Vitali

Corsi di laurea in Informatica e
Informatica per il Management
Alma Mater – Università di Bologna

Introduzione

Oggi esaminiamo:

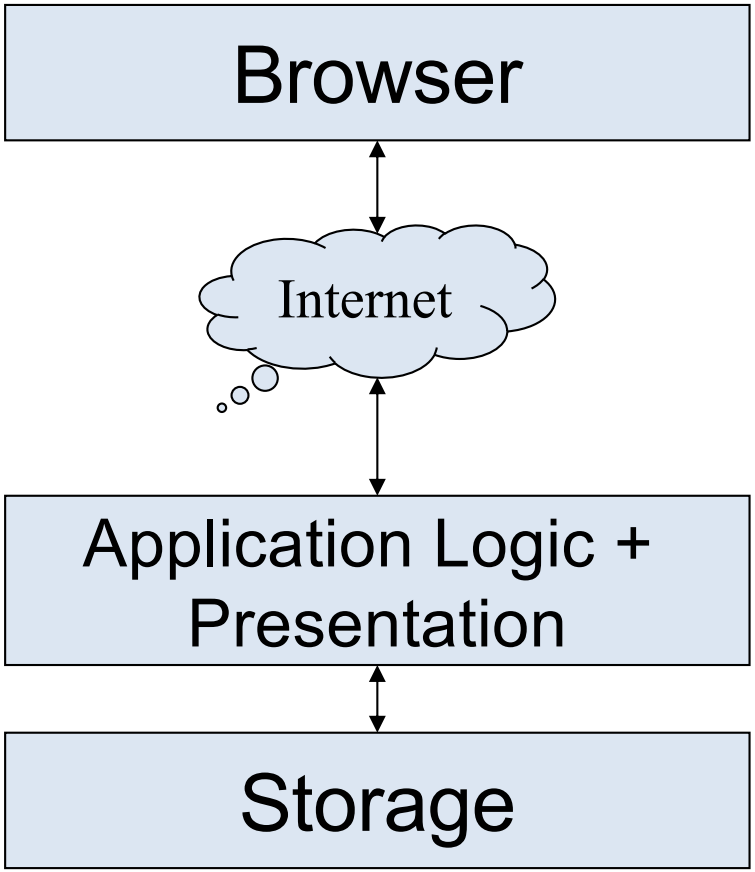
- Sito web statico
- Sito web dinamico: modello a tre livelli
  - Embedded code
  - Full application
- Sito web dinamico: modello a quattro livelli
- Rich client: applicazioni AJAX

Sito web statico

- Il server contiene una quantità di file fisici memorizzati in directory e di formati immediatamente riconoscibili dal browser (HTML, GIF, JPEG ecc.)
- C'è esattamente un file per ogni schermata possibile, ciascuno con un indirizzo (un URL) diverso.
- Il client richiede questi file ad uno ad uno e li riceve per la visualizzazione.
- Nessun contenuto visualizzato cambia rispetto al documento memorizzato su disco

- ***Pregi***: facile da realizzare, non richiede nessuna competenza tecnica
- ***Difetti***: totale mancanza di automazione e integrazione. Ogni file è indipendente dagli altri.

Siti dinamici: modello a tre livelli



- Parte dei contenuti è statica e memorizzata su file, ma la parte importante è generata in output da un'applicazione server-side.
- Questa spesso raccoglie dati da query su un DBMS, le elabora e le trasforma, e le spedisce come risposta al browser.
- L'application logic (a volte *business logic*) si modularizza rispetto allo storage per poter usufruire delle velocità e funzionalità dei DB indipendentemente dalla logica dell'applicazione.
  - Embedded code
  - Full application
- Tipicamente associata a modelli applicativi di tipo LAMP

Tre livelli: embedded code (1)

- Il sito è in realtà un'applicazione. Esso contiene alcuni file fisici memorizzati in directory e di formati immediatamente riconoscibili dal browser. C'è un file per ogni tipo di funzionalità (servizio) offerto.
- I file HTML contengono commenti speciali con codice inserito in qualche linguaggio di programmazione (PHP, ASP, ecc.). La parte puramente HTML è ripetuta su ogni risposta spedita dal server, mentre le istruzioni embedded generano codice HTML di volta in volta diverso.

```
<html>
   <head><title>PHP Test</title></head>
   <body>
      <p>Stai usando il browser
         <?php echo $_SERVER['HTTP_USER_AGENT']; ?>.
      </p>
   </body>
</html>
```

Tre livelli: embedded code (2)

- La singola pagina HTML può contenere (e di solito contiene) molti blocchi codice embedded. La parte HTML è il template, mentre il codice fornisce la parte dinamica dell'applicazione.
- Tuttavia si deve fare attenzione a mantenere sincronizzati tutti i template per quel che riguarda la presentazione. Un cambio di look al sito richiede di modificare ogni singolo file. Questa commistione di codice e template può essere considerata fragile e poco pulita.

- ***Pregi***: assolutamente potente. Si può realizzare qualunque applicazione server-side con poco sforzo e in maniera fortemente integrata ai template HTML
- ***Difetti***: architettura fragile: cambiamenti al codice possono esporre problemi di visualizzazione del template e viceversa modifiche al look possono cambiare il comportamento dell'applicazione.

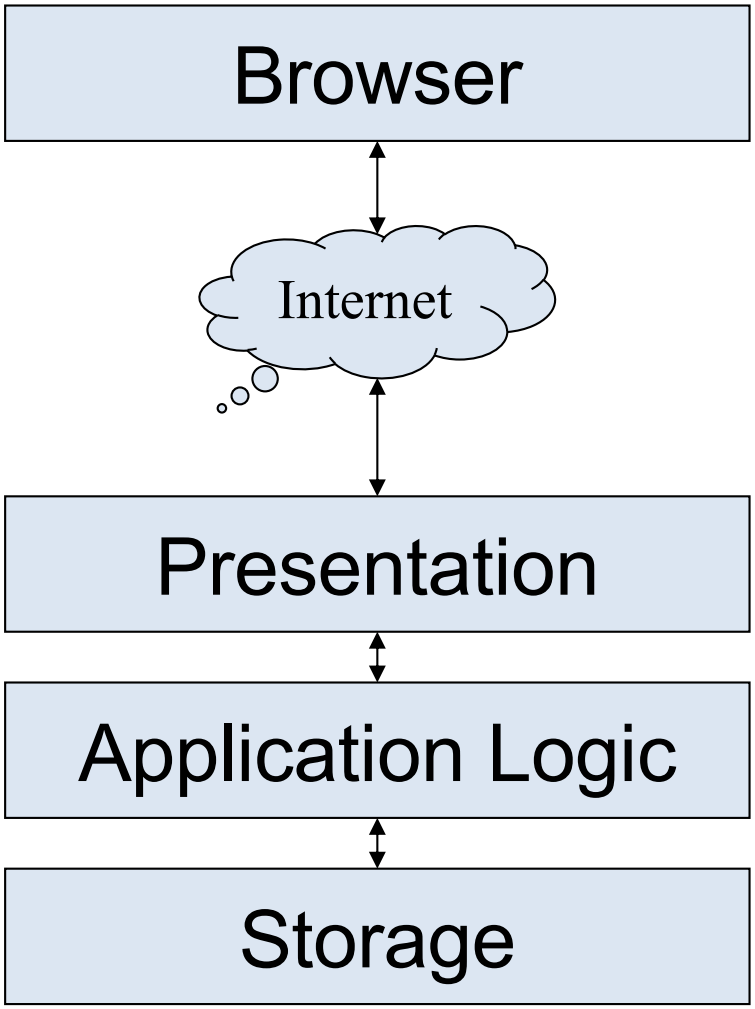
Tre livelli: full application (1)

- Si attua una separazione forte tra logica di applicazione e presentazione. Risiedono fisicamente su file diversi e vengono modificati in momenti diversi del processo di produzione, garantendo l'interdipendenza.
- In tutti i casi, il server esegue il file di programma e alla fine genera un singolo output dell'intera stringa del documento HTML.
- Ci sono vari modi per ottenerlo, ma questi sono i più comuni:
  - Il template viene generato da istruzioni nel linguaggio usato, ma contenute in un file separato, che viene richiamato all'interno del programma che gestisce la logica principale dell'applicazione
  - Il template viene contenuto in un file HTML statico che viene letto in una variabile del programma e sapientemente mescolato con ilcontenuto HTML dinamico, generato dal programma lavorando sulle stringhe.
  - Si usa un motore di template per PHP, ASP, ecc. Ogni produzione del programma diventa una variabile che viene usata all'interno del template, che è un file HTML. Il motore di template esegue il programma, poi carica il template e ne fa l'unione.

Tre livelli: full application (2)

- Pregi: si crea un'importante separazione tra application logic e presentazione. Questo permette di separare i processi di generazione e test dell'applicazione da quelli di progettazione della parte visibile del sito.
- Difetti: Questa separazione non è ancora completa. Ma si applica solo al template (layout complessivo, decorazioni, schemi di colori, parti fisse della pagina, ecc.). Il codice HTML di ogni singolo pezzo di output è ancora deciso dall'applicazione.
  - Ad esempio, che l'output di una query venga restituito come lista o come tabella è ancora una decisione dell'application logic.
  - Ad esempio, generare output personalizzati per device molto diversi è ancora difficile.

Siti dinamici: modello a quattro livelli (1)



- Nel modello a quattro livelli, il livello di application logic genera un output privo di aspetti presentazionali (e.g., un file XML) e lo passa a un altro livello, che chiamiamo appunto presentazione (anche presentation logic)
- Presentation modifica completamente il documento posizionando le varie parti dell'output dentro al template in maniera completa e assoluta, prendendo anche micro-decisioni riguardo a quale HTML usare di volta in volta.
- Template diversi possono applicarsi allo stesso output per personalizzarsi su specifici tipi di device mantenendo assolutamente intatta l'application logic.

Siti dinamici: modello a quattro livelli (2)

- Esistono motori di template che funzionano in questa maniera, ma la soluzione più semplice è usare XSLT per generare il documento HTML finale.
- Un processo della catena di produzione dell'output o semplicemente l'ultima istruzione della application logic del programma richiama il motore XSLT sul documento XML prodotto, e restituisce in output il documento HTML finale.

- ***Pregi***: completa e definitiva separazione tra application logic e presentation logic; architettura semplice, modulare, immediatamente ripetibile.
- ***Difetti***: i difetti di tutte le applicazioni server-side. Si veda la prossima slide

Rich client: Ajax e il web 2.0

- Tradizionalmente la programmazione client-server ha sempre criticato la presenza di application logic sul client.
  - Poco controllo sull'ambiente operativo
  - Difficoltà di distribuzione di versioni e patch
  - Possibilità di intromissioni da codice maligno che si spaccia per il client autorizzato.
- Tuttavia il WWW ha alcune obiezioni a queste critiche
  - ogni passo dell'applicazione richiede di consultare il server, eseguire una funzione dell'application logic, generare l'HTML finale, riceverlo e visualizzarlo. Questo può portare via molto tempo.
  - Ogni passo dell'applicazione ha un proprio URL, che richiede specifiche politiche di naming e di caching piuttosto complesse da gestire
- Inoltre ha risposte specifiche alle critiche di cui sopra:
  - I browser sono ambienti indipendenti dal sistema operativo, e (più o meno) sono standardizzati
  - Il codice comunque può risiede sul server ed essere distribuito al client ogni volta, in modo da garantire che giri sempre l'ultima versione.
  - Per lo stesso motivo, codice maligno non può essere iniettato da terze parti malevoli.

Rich client: Ajax e il web 2.0

- All'inizio dell'esecuzione del servizio, il browser carica una normalissima pagina HTML, che contiene codice Javascript e alcune librerie Ajax.
- Scopo delle librerie Ajax è duplice:
  - Garantire di fornire una libreria di funzioni comuni e indipendenti dallo specifico sistema operativo e browser utilizzato
  - Fornire librerie utili per la realizzazione di applicazioni client side in maniera facile (ad esempio, l'interazione con il server o la generazione di frammenti di output).
- Il documento HTML contiene solo le parti fisse (layout, ecc.) e le funzioni javascript. Il programma parte e attraverso un'apposita chiamata di libreria (XMLHttpRequest) chiede al server dati presentation-independent da convertire in HTML e visualizzare sulla pagina. Qui possiamo avere:
  - Esiste comunque un'applicazione server-side che genera l'XML e lo passa al client: cioè si sposta sul client solo la presentation logic
  - Sul server esiste solo il minimo indispensabile per fare interrogazioni ai server e ottenere le risposte alle query (si spostano sul client sia la presentation logic sia la application logic).

I framework

- I framework sono librerie che rendono più ricco, sofisticato e semplice l'uso di una tecnologia, come un linguaggio server-side, un linguaggio client-side o le specifiche grafiche di una pagina web.
- Server-side esistono dalla fine degli anni novanta, e hanno reso la programmazione a tre livelli drasticamente più facile.
- Client-side si sono sviluppate a partire dal 2002, su CSS e Javascript, con scopi molto difformi.

Framework server-side

- Struts, Django, Ruby on Rails, Symfony, Yii, Spring MVC, Stripes, Play, CodeIgniter, etc.
- Architettura a tre livelli, con una struttura concettuale Model-View-Controller, forniscono una varietà di servizi come:
  - Gestione di autenticazione e sicurezza
  - Accesso a database
  - Mappatura di URL
  - Scaffolding (o impalcatura): fornire strutture semplificate e generiche di applicazione che vanno poi riempite caso per caso dei dettagli.

Framework client-side - CSS

- Foundation,YAML, Twitter Bootstrap, etc.
- Un mix di classi CSS e strutture predefinite HTML per:
  - Gestione del layout di pagina (colonne, aree, stili integrati, ecc.)
  - Gestione dei layout responsive (che si adattano alla dimensione dello schermo)
  - Semplificazione e omogeneizzazione di feature frequenti (blocchi evidenziati, tab, barre di navigazione,menu dropdown, ecc.)

Framework client-side - Javascript

- Prototype, Dojo, GWT, jQuery, ExtJs, etc.
- Librerie Javascript per:
  - Omogeneizzare le versioni Javascript dei vari browser
  - Aggiungere funzionalità utili al linguaggio (Ajax, query su DOM, funzionalità Object Oriented, ecc.)
  - Fornire librerie di effetti grafici sofisticati (animazioni e transizioni)
  - Integrare piattaforme tradizionali (PC) e mobili (tablet e smartphone)
  - Fornire widget grafici di uso frequente (grid, tree, menu, form, date picker, ecc.)
  - Gestire template per frammenti HTML riusati frequentemente

Single-page web applications

- Sito web complesso e sofisticato composto in realtà da un unico documento HTML piuttosto ricco e complesso. E' possibile grazie a:
  - L'aumento della velocità di rete,
  - La disponibilità di Ajax e
  - La sofisticazione dei framework Javascript e CSS.
- Sia siti informativi sia siti di applicazione.
  - Gestione semplificata
  - Connessioni HTTP semplificate e unificate
  - Estrema flessibilità di reazione dell'applicazione
  - Offline editing

Progressive web applications

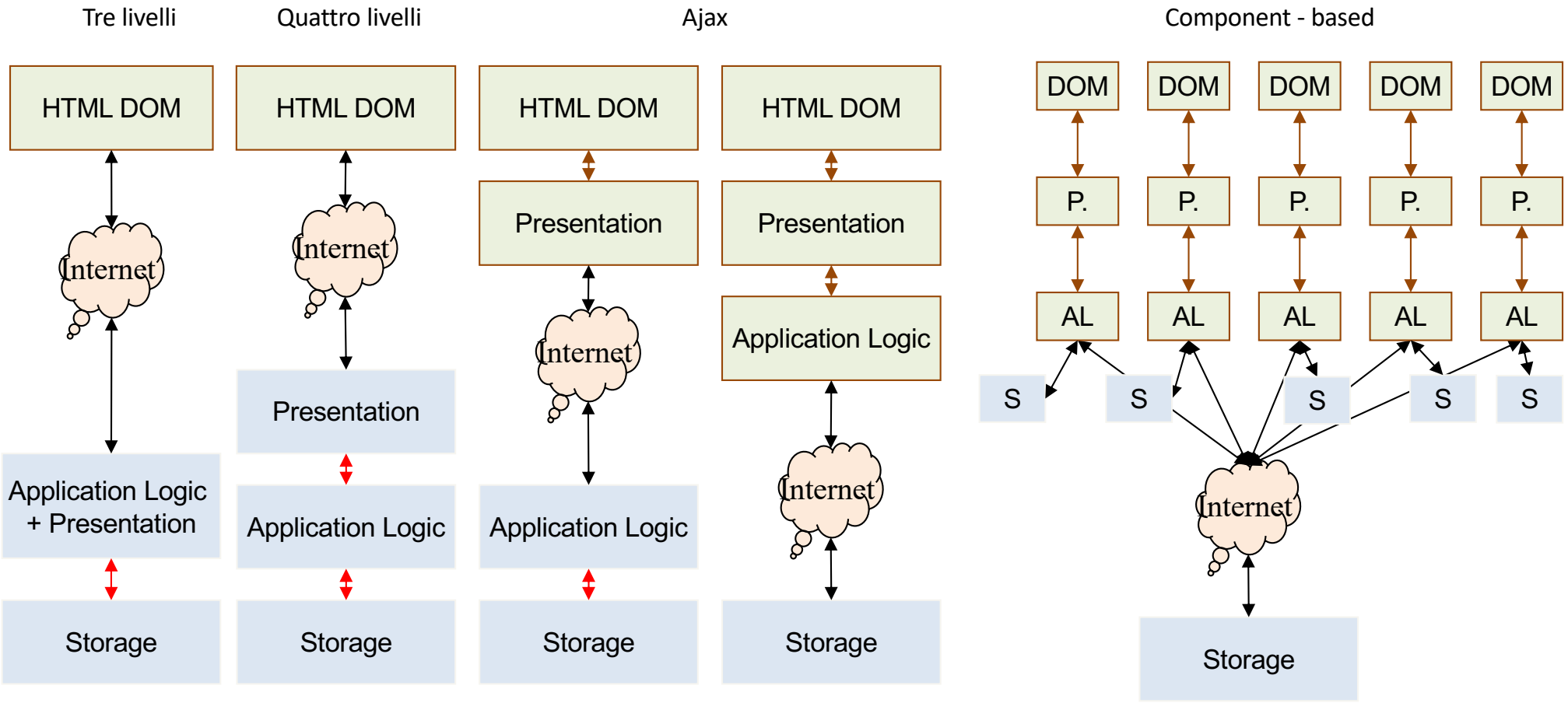
- Applicazione basata su tecnologie web che sembri nativa sul device su cui è installato. Frequente soprattutto nei device mobili
- L'ambiente di esecuzione è comunque un browser, ma l'app usa l'intero schermo (o finestra).
- L'app è installabile (tutto il codice viene trasferito sul device) e può funzionare anche in assenza di rete.
- Può mantenere localmente codice, stato dell'esecuzione, dati in maniera permanente. Può accedere per intero alle periferiche del device e può avere parti pre-compilate per ottenere velocità paragonabili ai binari nativi

Framework client-side - componenti

- Web components, AngularJS, React, Vue, Angular

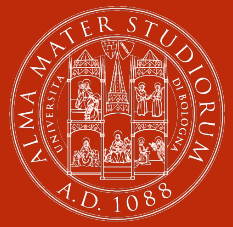
- Librerie Javascript per:
  - Creare strutture isolate, modulari ed incapsulate.
  - Ciascuna componente contiene e gestisce il proprio storage, la propria application logic e la propria presentation logic:
    - Storage: autonomia nella interazione coi server – uso di storage locale
    - Application logic: Incapsulamento di script autonomi ed isolati. Interfacce esplicite e controllate con altri componenti
    - Presentation logic: template incapsulati nel componente, con il proprio markup ed il proprio styling
  - Estensione del linguaggio HTML con elementi personalizzati (*custom elements*), frammenti di documento non pronti per la visualizzazione nascosti e inaccessibili (*shadow DOM*), strutture di HTML disattivi e pronti all'uso sulla base delle esigenze della application logic (*template*).

Un confronto



Conclusioni

- Qui abbiamo parlato dei
  - Principali tipi di applicazioni web



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Fabio Vitali

Corso di tecnologie web, A.A. 2017-18

Fabio.vitali@unibo.it

www.unibo.it