ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA
SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea in Ingegneria Informatica

PSICOPATOLOGIE DELL'INGEGNERE

Elaborata nel corso di: Sistemi Embedded

Tesi di Laurea di:
RICCARDO SORO

Relatore:
Prof. ALESSANDRO RICCI

ANNO ACCADEMICO 2003–2004
SESSIONE II

PAROLE CHIAVE

Prima

Seconda

Terza

Quarta

Quinta

Dedicato alla mia famiglia e ai miei amici, in particolare a Chiara che mi ha sempre sostenuto.

Indice

Introduzione

Il seguente elaborato si pone come obbiettivo l'analisi delle diverse tecnologie utilizzabili nello sviluppo di applicazioni che sfruttano il web come piattaforma. In particolare tratterá alcuni dei linguaggi maggiormente impiegati e le strategie adottabili per risolvere i problemi piú comuni in cui uno sviluppatore potrebbe imbattersi nella creazione di un'applicazione a pagina singola.

Nel primo capitolo saranno illustrate l'evoluzione del web e le principali tecnologie che hanno permesso il mutamento dal web statico al web dinamico, fino allo sviluppo di vere e proprie applicazioni paragonabili a quelle desktop.

Nel secondo capitolo l'enfasi si sposterá sull'aspetto architetturale delle applicazioni basate sul web, con una maggiore attenzione alle applicazioni a pagina singola. Saranno inoltre presentate alcune funzionalitá presenti nei framework piú popolari, le quali permettono di risolvere le problematiche piú comuni nello sviluppo di questo tipo di applicazioni.

Nel terzo capitolo verranno analizzati i nuovi linguaggi proposti come possibili sostituti delle tecnologie fino ad ora utilizzate, con l'obbiettivo di prevedere il miglior scenario per ogniuno di essi.

La seguente tesi quindi cercherá di fornire una panoramica sulle modalitá di sviluppo delle attuali applicazioni web, provando a prevedere quale linguaggio prevarrá.

Capitolo 1

Sviluppo applicazioni web

da fare

1.1 Web 1.0

Il Web é il principale servizio offerto da Internet e permette di consultare un elevato numero di contenuti e allo stesso tempo dei servizi messi a disposizione da altri utenti della rete.

Si basa principalmente su 3 componenti:

1. html:il linguaggio di markup per le pagine web.
2. http:un protocollo di rete che permette lo scambio di informazioni.
3. url:l'indirizzo che contraddistingue in maniera univoca una risorsa.

La prima idea di web fú teorizzata da Vanner Busgh nel 1945, ma solo nell'agosto 1991 Tim Berners-Lee pubblicó il primo sito internet della storia, con l'obbiettivo di favorire la condivisione di documenti tra i ricercatori. Divenne quasi subito popolare e numerose entitá commerciali inziarono ad investire grandi quantitá di denaro nello sviluppo e nella progettazione di siti web. In quel periodo il paradigma utilizzato per la programmazione dei siti prevedeva un'interazione unilaterale,cioé forniva all'utente solamente la possibilitá di scegliere e visualizzare i contenuti messi a disposizione da altri, ma non di modificare in alcun modo lo stato o le informazioni. I file erano in formato html e non era prevista alcuna modifica o manipolazione da parte

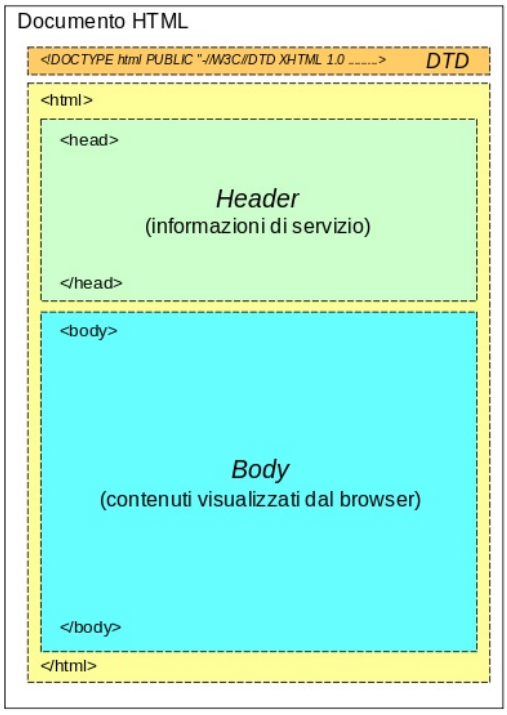

Figura 1.1: Struttura di una pagina html

del server prima dell'invio del file al browser dell'utilizzatore, da qui il nome web statico.

1.1.1 HTML

Html é un linguaggio di markup proposto da Tim Berners-Lee nel 1989 per la formattazione e l'impaginazione dei documenti ipertestuali. Questo linguaggio definisce il layout della pagina web tramite l'utilizzo di particolari tag di formattazione, i quali hanno lo scopo di specificare la funzione, il colore, la grandezza e la dimensione della porzione di testo da essi delimitata. Come si puó notare nella figura 1.1, il file html deve iniziare con una stringa che indica il tipo di sintassi e la versione in cui é stato scritto il documento per poter garantire la corretta visualizzazione da parte del browser. La

```
<!DOCTYPE html>
<html>
<head>
<title>Titolo della pagina</title><!--non visibile nella
pagina del browser-->
</head>
<body>
<!--da qui in poi gli elementi sono visibili nella pagina
del browser-->
<h1>Le pesche</h1>
<p>Le pesche sono un frutto sano.</p>

</body>
</html>
```

Le pesche

Le pesche sono un frutto sano.

Figura 1.2: Esempio di una pagina html.

struttura piú esterna é contenuta nei tag ¡html¿ e ¡/html¿ e comprende obbligatoriamente due sottosezioni:head e body. Head contiene tutte le informazioni di controllo che non sono visualizzate dal browser, mentre body contiene quelle che l'utilizzatore puó vedere a schermo. Dentro queste due sottosezioni é possibile utilizzare ulteriori tag per effettuare controlli o per specificare la formattazione dei contenuti.

Nella figura 1.2 si puó notare come i tag dentro la sezione Head non siano direttamente visibili nella pagina caricata dal browser, al contrario di quelli contenuti in body, i quali vengono visualizzati.

1.1.2 Protocollo http

Il protocollo HTTP (HyperText Transfer Protocol) é di tipo applicativo e viene utilizzato per il trasferimento di informazioni sul web nell'architettura client-server, nella quale il server resta solitamente in ascolto sulla porta 80. La prima versione (0.9) fu pubblicata negli anni '80, ma la prima versione effettivamente disponibile (1.0) fu sviluppata sempre da Tim Berners-Lee nell'anno 1991. La prima versione era molto limitata, non prevedeva infatti la possibilitá di ospitare piú di una risorsa all'interno del server, era priva di meccanismi di sicurezza e non potevano essere riutilizzate le connessioni aperte; tutto questo portó alla sua evoluzione (versione 1.1) nel 1999.

Il protocollo http si basa su un meccanismo di richiesta da parte del client (solitamente il browser) e di risposta da parte del server (solitamente la macchina su cui é ospitata la risorsa), tanto che sono previsti solo due

tipi di messaggi:uno per la richiesta e uno per la risposta. Le connessioni vengono solitamente chiuse dopo il ricevimento della risposta garantendo un numero limitato di connessioni attive contemporaneamente, ma svantaggiando lo sviluppo di servizi web con meccanismi di sessione. Questo difetto fu poi risolto implementando i coockie che, con le dovute configurazioni, permettono di conservare lo stato della sessione dell'utente anche con un protocollo stateless come l'HTTP.

1.1.3 URL

Con il termine URL (Uniform Resurce Locator) si intende la sequenza di caratteri che identifica in maniera univoca una risorsa nella rete. La struttura dell' URL si compone solitemente di 7 parti:

protocollo://[username:password@]host[:porta]/percorso[?querystring][#fragment]

1. Specifica il protocollo da utilizzare nel dialogo con il server, di dafault ḤTTP.

2. É possibile indicare le credenziali per l'accesso alla risorsa richiesta; alcuni browser impediscono questi parametri a causa del basso livello di sicurezza. L'username e la password infatti vengono trasmesse in chiaro e l'utilizzatore potrebbe diventare vittima di phishing.

3. L'host identifica il server sul quale é ospitata la risorsa e puó essere rappresentato usando l'indirizzo IP o indicando il nome di dominio; nel secondo caso il browser si occuperá di convertirlo in IP attraverso il DNS.

4. Indica la porta del servizio di rete al quale si vuole effettuare la richiesta; di default é 80 per il protocollo HTTP e 443 per il protocollo HTTPS.

5. Indica il percorso nel file system del server in cui giace la risorsa che si desidera ricevere; se non viene specificato il server restituirá un percorso impostato di default.

6. É possibile inserire una serie di stringhe separate dalla parte precedente dal simbolo "?" che consentono l'invio al server di informazioni extra da parte del client (es. ...?parametro1=valore&parametro2v̄alore2)

7. Indica una parte o una porzione della risorsa.

1.2 Web 2.0 e applicazioni multipagina

Il paradigma di sviluppo descritto nella sezione precedente risulta poco pesante per il server che non dovrá eseguire alcuna attivitá computazionale dal momento che non avviene nessuna modifica dei contenuti inviati all'utente. Nonostante le buone performance, questo paradigma soffre di un numero elevato di limiti, come l'impossibilitá di permettere all'utente di interagire con la pagina o di adeguare le informazioni da inviare in base all'utilizzatore che le richiede. Queste mancanze portarono molti siti web statici alla migrazione verso il paradigma dinamico.

Per permettere al server di generare pagine html in maniera dinamica furono inizialmente sviluppate le CGI (Common Gateway Inteface) con le quali era possibile delegare ad un programma esterno la generazione in tempo reale del codice html, anche se con limitazioni e risultando una pratica molto pesante per il calcolatore. Negli anni successivi i browser divennero sempre piú completi di funzionalitá come il supporto dei linguaggi di scripting e nei web server si inziarono ad utilizzare linguaggi ideati per la creazione di pagine dinamiche; il web era diventato una possibile piattaforma sulla quale sviluppare vere e proprie applicazioni.

1.2.1 HTML5

Il web dinamico, comunemente chiamato web 2.0, non potrebbe esistere senza HTML5 che ha introdotto numerose novitá allo scopo di mettere a disposizione degli sviluppatori un'insieme di funzionalitá per la creazione di pagine web dinamiche e non piú statiche. In particolare, questa nuova versione, prevede la possibilitá di salvare in locale una quantitá elevata di dati fino a permettere l'utilizzo di un' applicazione web anche in assenza momentanea di connessione.

Grazie a queste nuove novitá ci fu un notevole aumento di web app sviluppate unendo HTML5, css e JavaScript; questi 3 linguaggi infatti assieme riescono a fornire tutte le funzionalitá richieste per lo sviluppo di una generica applicazione:HTML5 rappresenta il formato delle pagine e il suo contenuto, css gestisce gli aspetti presentazionali e l'estetica della pagine e JavaScript gestisce le interazioni dell'utente.

1.2.2 CSS

Prima dell'arrivo di css l'unico modo che aveva lo sviluppatore per modificare la formattazione delle proprie pagine html era quello di sfruttare i tag proprietari forniti dai browser. Questi tag peró risultavano spesso pesanti, ridondanti e incompatibili con gli altri browser, soprattutto con i dispositivi mobile; questo causava la visualizzazione sul browser di una pagina mal formattata, confusa e di difficile lettura. Per tentare di risolvere tali problemi, nel 1996 W3C emanó le specifiche della prima versione di CSS, con l'obbiettivo di separare il contenuto della pagina dalla sua formattazione, riuscendo a risolvere almeno in parte i problemi sopra descritti. Le successive versioni permisero di creare fogli di stile separati per i dispositivi mobili garantendo una buona compatibilitá anche con questi device piú recenti.

1.2.3 JavaScript

JavaScript é un linguaggio di scripting interpretato, debolmente orientato agli oggetti e tipizzato, comunemente utilizzato nella programmazione web lato client per la creazione di siti e applicazioni web. É in grado di creare effetti dinamici sulla pagina tramite funzioni invocate da eventi scatenati dalle interazioni dell'utente, come il click del mouse. Fu da subito molto apprezzato per la sua semplicitá, per la sua natura di linguaggio asincrono e perché soddisfava in pieno le esigenze delle prime applicazioni web.

Recentemente si utilizza anche lato server grazie a Node.js, una piattaforma event driven sviluppata quasi interamente in javascript, che offre un alto livello di efficienza grazie al suo modello di networking. Quest'ultimo non sfrutta la programmazione concorrente, ma rimane in sleep fino al verificarsi di un evento immediatamente gestito, eventualmente in maniera asincrona.

Le caratteristiche principali di JavaScript sono le seguenti:

1. é un linguaggio interpretato,quindi il codice non deve essere compilato, ma viene eseguito dall'interprete incluso nel browser in run time incaricando quindi il client di effettuare le operazioni computazionali necessarie alleggerendo di conseguenza il server.

2. la sintassi é simile a quella di c e java

3. é un linguaggio debolmente tipizzato

4. é un linguaggio debolmente orientato agli oggetti

Per lo sviluppo delle prime applicazioni basate sul web, javascript fu molto apprezzato per le sue caratteristiche, ma con il continuo aumento di complessitá delle applicazioni basate sul web si é sentita la neccessitá dello sviluppo di framework che aiutassero il programmatore nell'organizzazione del codice.

1.2.4 AJAX

Con il termine AJAX (Asynchronous JavaScript and XML) si intende una tecnica multipiattaforma di sviluppo di applicazioni web dinamiche e interattive che permette uno scambio di dati tra client e server asincrono e in background. Questa tecnica é alla base dello sviluppo delle applicazioni che usano il web come piattaforma perché permette la modifica della pagina senza che il browser debba effettuare un'operazione di refresh.

Per poter eseguire le operazioni di scambio dati in background si sfruttano le funzionalita offerte dai linguaggi di scripting come JavaScript, mentre per il markup e lo stile ci si appoggia a HTML e CSS. A differenza di come suggerisce il nome,i dati scambiati possono essere in vari formati e non necessariamente in XML,infatti é possibile trasferire degli oggetti in formato JSON o del semplice testo in HTML.

La figura 1.3 mostra come AJAX funga da intermediario in grado di gestire le richieste del client e le risposte del server; le prime vengono scatenate da una chiamata in JavaScript, mentre le seconde vengono ricevute come dati di tipo XML(o altri formati) dai quali viene generato il codice HTML e CSS.

Per poter essere correttamente utilizzate, le applicazioni web sviluppate con AJAX devono essere eseguite su browser che supportino tutte le tecnologie sopraelencate, ma al giorno d'oggi sono interamente supportate da tutti i dispositivi, compresi i device mobili. Grazie a AJAX diventó possibile sviluppare applicazioni che stessero interamente in una sigola pagina web, delegando ai linguaggi di scripting l'onere di modificare le informazioni visualizzate sul browser tramite l'utilizzo delle comunicazioni asincrone e in background con il server.

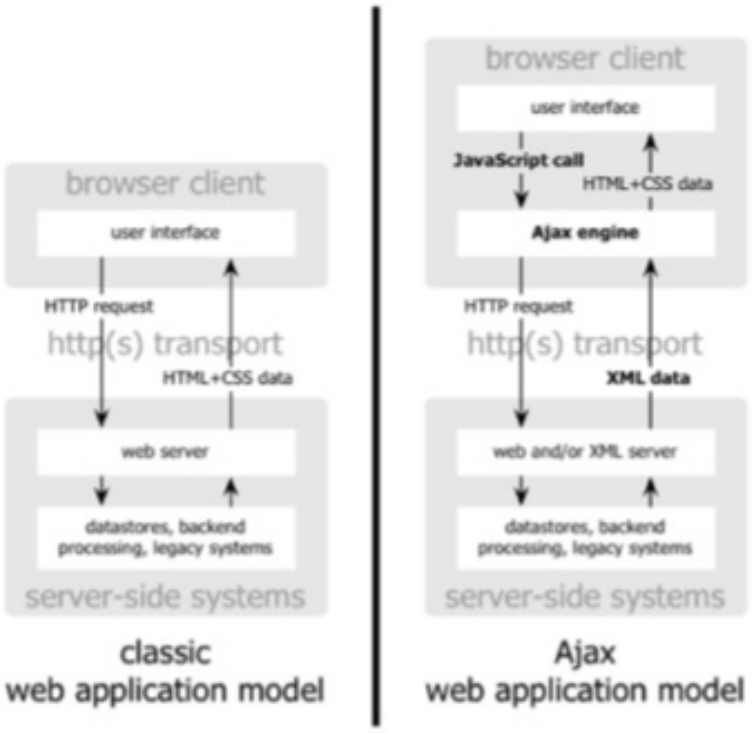

Figura 1.3: Differenze tra l'utilizzo di ajax o del metodo classico

1.3 Singole Page Application

Con il termine spa (single page application) si intende un'applicazione web contenuta interamente in una pagina al fine di migliorare la user experience fino a rendere l'esperienza paragonabile all'utilizzo di un'applicazione desktop. A differenza delle applicazioni desktop, le spa sono molto facili da aggiornare, dal momento che basta caricare sul server l'ultima versione senza costringere l'utente a reinstallare il programma. Le applicazioni multipagina richiedono il refresh quasi ad ogni interazione con la pagina, generando quindi dei tempi di attesa maggiori rispetto ad una spa. Con questo ultimo tipo di programmazione della pagina, invece, si é in grado di ottenere un'applicazione molto veloce (solo il caricamento inziale diventa leggermente piú lungo) e l'utilizzo della banda é solitamente limitato all'accesso ai dati, che spesso sono contenuti in database remoti. Per ridurre ulteriormente l'uso di banda si puó optare per il salvataggio in cache dei dati,se il tipo di applicazione lo permette, al fine di limitare gli accessi al database e di conseguenza ridurre i tempi di caricamento necessari al corretto funzionamento dell'applicazione.

Come si puó notare nelle figure 1.4 e 1.5, un'altra differenza rispetto alla programmazione multipagina riguarda l'utilizzo dei file html: nella spa non si avranno piú molte pagine html complete, ma se ne avrá una contenente i placeholders i quali saranno successivamente riempiti dinamicamente tramite le funzioni messe a disposizione da JavaScript.

Con l'avanzare del tempo le applicazioni web sono diventate sempre piú complesse e gli sviluppatori hanno avvertito la necessitá di framework che li aiutassero nell'organizzazione e nella stesura del codice per evitare risultati caotici e facilitare il riutilizzo e la manutenzione nel tempo.

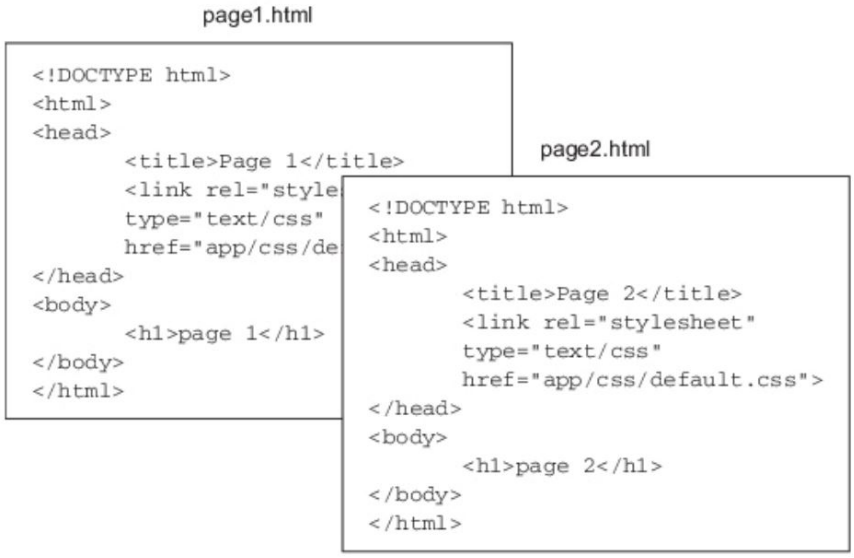

Figura 1.4: Struttura html di un'applicazione multipagina

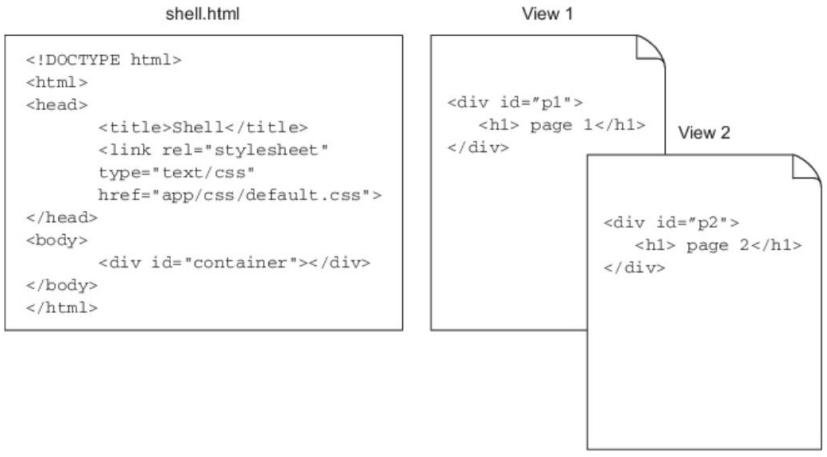

Figura 1.5: Struttura html di una single page applicatione

Capitolo 2

Progettazione di SPA e utilizzo di framework

Un framework é un'architettura logica di supporto su cui un software puó essere progettato e realizzato al fine di semplificarne lo sviluppo. Nella progettazione di una spa il suo utilizzo porta notevoli vantaggi, tra cui una migliore organizzazione del codice e l'accesso a funzionalitá di livello nettamente superiore rispetto al puro javascript. Bisogna peró tenere conto che il suo utilizzo puó essere visto come una dipendenza, pertanto si avrá un prodotto finale piú pesante; stará alla sviluppatore decidere se i vantaggi superano i costi.

Ogni framework di JavaScript solitamente incentiva un solo pattern progettuale tra MVC,MVP e MVVM.

2.1 Utilizzo pattern MV*

Con il termine mv* si comprende l'insieme dei tradizionali design pattern che utilizzano la suddivisione in model, view e un terzo elemento. Il model fornisce i metodi per l'accesso ai dati dell'applicazione, mentre il view si occupa della visualizzazione dei dati contenuti nel model e dell'interazione tra utenti e agenti.

Il terzo elemento varia a seconda del pattern utilizzato ed ognuno di essi offre modi diversi per separare i dati dalla logica e dalla rappresentazione dei risultati. Solitamente i framework si differenziano, oltre che per le eventuali

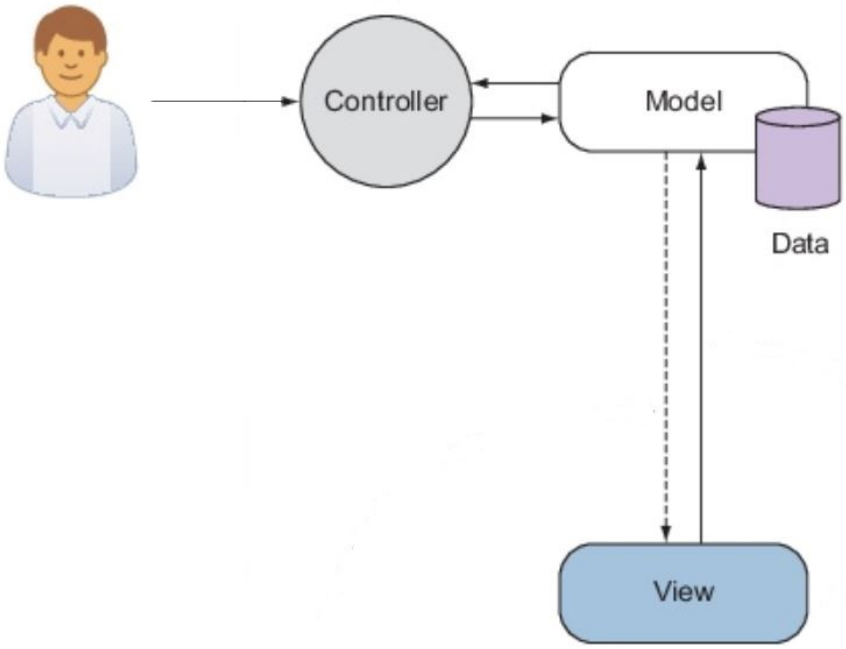

Figura 2.1: Logica del pattern MVC

funzionalitá diverse, per il tipo di patter mv* che favoriscono durante lo sviluppo dell'applicazione.

2.1.1 MVC

Nel pattern MVC, il primo ideato, il terzo elemento é chiamato controller e funge da entry point per l'applicazione. In questa strategia il controller contiene la logica per elaborare gli input dell'utente e inviare i comandi al model per aggiornare lo stato, il quale si occupa quindi di notificare alla view il cambiamento di stato e quest'ultima accede ai nuovi dati attraverso il model.

La figura 2.1 aiuta a comprendere meglio le relazioni tra i vari componenti di questo pattern, in particolare si nota come l'utente interagisca in maniera diretta solo con il Controller e come il model funga da intermediario tra il Controller, la View e i dati.

```
<html ng-app="MVCApp">
<head>
<title>TITOLO</title>
<script
SRC="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.13
/angular.js">
</script>
<script type="text/javascript">
//creazione del Controller
var app = angular.module('MVCApp', []);
    app.controller('Cliente', function($scope) {
//Creazione del Model
    $scope.cliente =  {
         'Nome'     :   'Mario Rossi',
         'Indirizzo':   'Via Poldo n.6',
         'Email'    :   'mariorossi@gmail.com'
     }
    });
</script>
<!-- Creazione della View -->
</head>
<body>
<p>Visualizzazione nella pagina dei dati<br>del model
passando per il controller</p>
<div ng-controller="Cliente">
<h3>{{ cliente.Nome }} </h3>
<h3>{{ cliente.Indirizzo }} </h3>
<h3>{{ cliente.Email }} </h3>
</div>
</body>
</html>
```

Visualizzazione nella pagina dei dati
del model passando per il controller

Mario Rossi

Via Poldo n.6

mariorossi@gmail.com

Figura 2.2: Utilizzo del pattern MVC con il framework Angular

Questo pattern risulta una buona scelta se la logica, il model e la Unit Interface hanno un livello di complessitá tra loro equivalente. L'uso del MVC risulta efficace nello sviluppo di applicazioni web composte da molte pagine dalle dimensioni contenute, come ad esempio un catalogo. Il suo utilizzo é supportato dal framework Angular.js; é presente un esempio dell'utilizzo di questo pattern con Angular.js nella figura 2.2.

In questa figura si puó notare come il model contenga le informazioni del cliente e fornisca, tramite la variabile Cliente, l'accesso al controller dei dati. La view, passando per il controller, riempie i tag h3 con le informazioni contenute nel model.

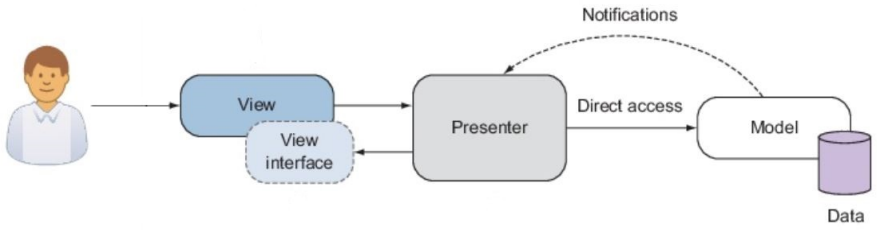

Figura 2.3: Struttura del pattern mvp

2.1.2 MVP

Nel pattern MVP, nato come alternativa al pattern MVC, il terzo elemento é chiamato presenter e si frappone tra gli altri due componenti. In questa strategia gli input dell'utente vengono ricevuti dalla view, la quale procede inoltrandoli al presentatore che ha il pieno accesso sul model. Il model invia delle notifiche al presentatore che si occupa anche dell'aggiornamento della view.

Come si nota nella figura 2.3, questo pattern sposta l'entry point sulla view, concentrandone peró la logica nel presentatore che mantiene aggiornati model e view in modo da specializzare i compiti di questi due. L'uso del pattern MVP risulta efficace nello sviluppo di applicazioni composte da pagine molto complesse contenenti un elevato numero di funzionalitá, come ad esempio un gioco.

2.1.3 MVVM

Il pattern MVVM, sviluppato con l'obbiettivo di semplificare la programmazione ad eventi, vede come terzo componente il model view. Quest'ultimo svolge una funzione di intermediario tra vista e modello e si occupa della gestione della logica della view; interagisce quindi col modello per ottenere i dati per poi rielaborarli e renderli disponibili in una forma utilizzabile dalla view.

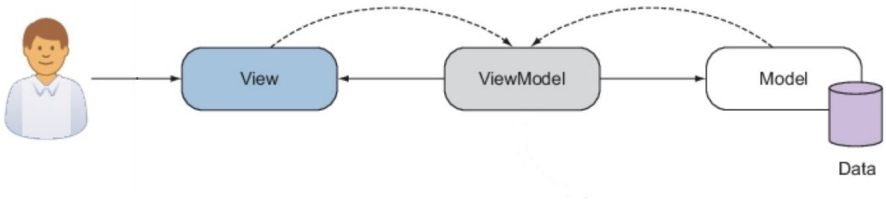

Figura 2.4: Struttura del pattern mvvm

2.2 Modularitá

Un modulo puó essere definito come una parte o un componente di qualcosa di piú grande (es: modulo di pagamento) e lo stesso concetto puó essere utilizzato in fase di progettazione di un'applicazione; in particolare nelle spa, le quali hanno spesso un livello di complessitá elevato.

JavaScript é un linguaggio molto libero e flessibile, di conseguenza si rischia di avere del codice disordinato, per evitare questo problema é stato studiato il module pattern che permette una divisione delle varie porzioni di codice in base allo scopo per cui sono state create.

I moduli sono utili non solo per mantenere ordinato il progetto,ma anche per tenere alcune parti di codice private, creare delle API pubbliche richiamabili da altri moduli, evitare conflitti di nomi molto difficili da risolvere in fase di debug e per distinguere facilmente funzioni e variabili con nomi simili.

2.2.1 Implementazione

Non é necessario appoggiarsi ai framework per l'implementazione del module pattern, poiché puó essere interamente costruito in JavaScript. Per creare un modulo in JS é necessario assegnare ad una variabile globale una funzione anonima, dentro la quale si istanzieranno le variabili e le funzioni private; nel return si inseriscono invece le funzioni pubbliche che si vogliono rendere accessibili dal resto del programma.

Come emerge dall'esempio nella figura 2.5, le funzioni pubbliche possono essere richiamate specificando il nome della variabile globale e la funzione di cui si necessita. Per applicazioni di piccola grandezza l'uso dei moduli é sufficiente a garantire ordine, in applicazioni medio-grandi invece si necessita

```
var numberModule = (function() {
    var num1 = 2;

    function addNumbersInternally(num2) {
            return num1 + num2;
    }

    return {
            addTwoNumbers : function(num2) {
                    alert(addNumbersInternally(num2));
            }
     };

})();

numberModule.addTwoNumbers(2);
```

Figura 2.5: da fare

spesso di un' ulteriore divisione dei moduli in sotto-moduli, che saranno implementati con la stessa logica; diventano quindi delle funzioni pubbliche del primo modulo composte da funzioni pubbliche e private.

Per accedere ad una funzione di un sotto-modulo si deve perció passare sia dal modulo padre che dal secondo modulo, portando ad un ulteriore livello di divisione delle funzioni in base allo scopo. Una buona pratica, se si hanno dei moduli di una discreta grandezza, é dividerli fisicamente in file separati e inclusi nel file HTML dentro ai tag SCRIPT.

2.2.2 Ottimizzazione del download delle dipendenze

Nella maggior parte dei browser il tag SCRIPT blocca la normale esecuzione dell'applicazione finché il download non termina e il codice non viene interamente eseguito.

Per evitare problematiche HTML5 ha introdotto due attributi del tag SCRIPT che migliorano drasticamente il tempo di caricamento della pagina:defer e async.

1. -Defer: specifica che il codice contenuto nel file venga eseguito solo al completo caricamento della pagina.

2. -Async: specifica che lo script viene venga eseguito in maniera asincrona con il caricamento della pagina

Suddividendo l'applicazione in moduli e salvandoli su file distinti c'é peró il rischio che una parte dell'applicazione non sia scaricata in tempo e che il corpo principale provi ad eseguire una funzione pubblica di un modulo non ancora completamente scaricato. In questa eventualitá l'utente riscontra inevitabilmente un malfunzionamento, oppure un blocco parziale o totale dell'applicazione.

Per risolvere tale problema sono state sviluppate librerie che gestiscono il download asincrono dei moduli e che permettono l'inizio dell'esecuzione di un modulo solo dopo che tutte le relative dipendenze sono scaricate e disponibili. Per poter funzionare hanno bisogno di una piccola configurazione nella quale si devono specificare per ogni modulo tutte le dipendenze necessarie, dopo di che la libreria si occuperá della gestione dei download e della corretta esecuzione dei moduli.

2.3 Router

La caratteristica che contraddistingue una spa da un'applicazione web multipagina é appunto il non dipendere da piú pagine ma di concentrare l'applicazione solo su una. Nessuna applicazione un minimo complessa puó peró essere concentrata in una sola vista, perció é necessario un elemento che ci permetta di navigare da una view all'altra passandole eventualmente degli stati. Questo puó essere risolto, anche se con alcune problematiche, cambiando dinamicamente l'intera view, perdendo peró l'utilizzo dei comandi indietro e avanti incorporati nel browser.

Per simulare completamente un'applicazione multipagina (dove questi comandi sono naturalmente supportati) sono state ideate delle tecniche di routing all'interno della spa.

Nella figura 2.6 é possibile osservare il funzionamento di un'applicazione multipagina nella quale, per cambiare la view, é necessario effettuare una richiesta al server che ospita la pagina da visualizzare.

Il router invece, come si puó osservare nella figura 2.7, intercetta queste richieste e fornisce istantaneamente le informazioni per visualizzare correttamente la pagina eliminando la necessitá di una connessione col server.

Figura 2.6: da fare

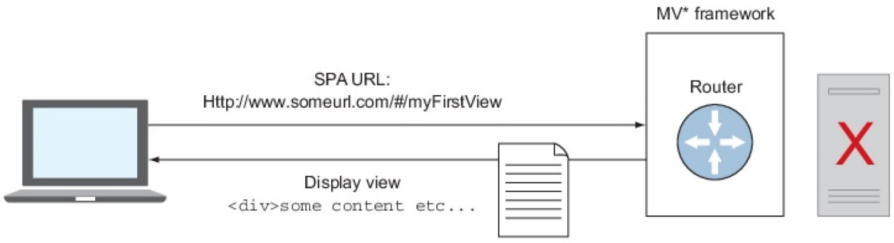

Figura 2.7: da fare

2.3.1 Caratteristiche e funzionalitá

Un ipotetico utente si aspetterá di poter utilizzare i normali comandi del browser per muoversi all'interno dell'applicazione e di poter associare ad ogni vista un url univoco. Per risolvere questo problema sono stati introdotti all'interno della spa i router che restano in attesa di un evento e, riempendo i ¡div¿ del file html, cambiano la view; a questo punto l'utente avrá la sensazione che la pagina visalizzata sia completamente nuova.

Alcuni framework possiedono internamente una sorta di router o, in alternativa, é possibile includere una libreria esterna specializzata; in entrambi i casi, peró, si necessita di una configurazione.

Solitamente, si configurano impostando 4 o 5 campi, a seconda del router:

1. name: rappresenta il nome della route.

2. verb: la funzionalitá http che si deve intercettare (es. Get).

3. path: il percorso url. Ogni volta che l'url del browser cambia, il router lo compara con questo campo per vedere se combaciano.

4. funzione: questo campo é associato a dei comandi che il router deve eseguire quando il path combacia con l'url del browser.

5. view: alcuni router prevedono la possibilitá di specificare un file html da caricare come vista.

Un cambiamento all'url causa peró un refresh della pagina perché il browser, tramite get, cercherá di scaricare la risorsa e questo va contro i principi delle spa. Per evitare questa problematica sono stati ideati due metodi:il primo utilizza il fragment dell'url, mentre il secondo le history API

2.3.2 Metodo Fragment Identifier

Il primo metodo ideato (e il piú compatibile) per evitare il refresh della pagina prevede l'utilizzo di un fragment identifier univoco per contraddistinguere le varie viste. Qesto identificatore puó essere una qualsiasi stringa e verrá unito all' URL interponendo tra i due il simbolo #. Questa seconda parte dell'url viene vista dal browser come un riferimento ad una parte del documento attuale e non ad uno diverso, evitando perció un refresh.

esempio fragment identifier: http://mysite/#id

Questo identificatore puó essere impostato settando il parametro windows.location.hash = "myId" e aggiunge un nuovo elemento nella cronologia del browser, permettendo quindi all'utente di poter utilizzare i comandi avanti e indietro all'interno della nostra spa.

2.3.3 Metodo HTML5 history API

Il metodo piú recente per evitare il refresh della pagina utilizza invece le API messe a disposizione da html5, in particolare le history API che permettono la manipolazione della cronologia del browser. Tale metodo é considerato migliore perché non obbliga lo sviluppatore a creare dei percorsi contenenti obbligatoriamente il simbolo #. I due nuovi metodi introdotti da html5 sui quali si basano i router di questo tipo sono i seguenti:

1. pushState(): permette l'inserimento di un nuovo indirizzo nella cronologia.

2. replaceState(): permette di sovrascrivere un indirizzo nella cronologia con quello attuale.

Con il loro utilizzo, il router é quindi in grado di modificare l'url visualizzato nel browser senza causare un refresh. Normalmente, i router all'interno dei framework e delle librerie specializzate supportano entrambi i metodi, sará compito dello sviluppatore scegliere quello piú opportuno per l'applicazione.

2.4 AngularJS

AngularJS é un framework scritto in JavaScript con l'obiettivo di risolvere alcuni dei problemi che vengono riscontrati dagli sviluppatori durante lo sviluppo di applicazioni a singola pagina. Questo framework permette di estendere gli attributi di HTML introducendone alcuni che fungono da direttive; questi poi verranno letti da Angular.js e interpretati. Tramite questi tag personalizzati lo sviluppatore é in grado di legare le parti di ingresso e di uscita della pagina al modello, il quale é rappresentato da variabili in JavaScript. Angular supporta il pattern MVC, ma, grazie alla sua flessibilitá, permette anche lo sviluppo di applicazioni con il pattern MVVM e, con piú difficoltá, con il pattern MVP; per questo motivo é considerato uno dei pochi framework definito MV*.

2.4.1 Attributi in AngularJS

Esistono molti attributi interpretabili da AngularJS ed é possibile, per lo sviluppatore, crearne dei nuovi personalizzandone il comportamento. L'elenco completo delle direttive con le relative descrizioni é disponibile presso il sito www.w3schools.com nella sezione dedicata ad Angular.

Nella figura 2.8 si puó notare che, per poter utilizzare AngularJS, é necessario inserirlo tra le dipendenze all'inizio della pagina. In questo esempio viene creata una semplice applicazione che calcola dinamicamente il costo in base a quantitá e prezzo, settabili dall'utilizzatore. Per ottenere l'aggiornamento dinamico del costo é stato utilizzato il tag ng-model per legare i valori degli input alle variabili quantitá e prezzo, per poi visualizzarne il prodotto. Queste variabili sono state inizializzate tramite l'attributo data-ng-init, ma, come si nota nell'esempio, é possibile cambiare il valore di queste inserendo un valore negli input; il costo si aggiorna automaticamente al variare del valore di una delle due variabili.

2.4.2 Moduli in AngularJS

La creazione di un modulo puó essere effettuata intermente in JavaScript, spesso peró i framework mettono a disposizione dei metodi che semplificano la sua creazione e la gestione delle sue dipendenze.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1
.6.4/angular.min.js"></script>
<body>

<div data-ng-app="" data-ng-
init="quantita=2;prezzo=3">

<h2>Calcola Costo</h2>
quantità:<input type="number" ng-model="quantita">
<br>
prezzo:<input type="number" ng-model="prezzo">

<p><b>costo totale:</b> {{quantita * prezzo}}</p>

</div>

</body>
</html>
```

Calcola Costo

quantità: 3
prezzo: 4

costo totale: 12

Figura 2.8: Il codice rappresenta una semplice pagina scritta utilizzando AngularJS

Angular offre il metodo module(), il quale accetta come parametri una stringa che rappresenta il nome del modulo e un vettore nel quale specificare le eventuali dipendenze. É possibile accedere ad un modulo precedentemente creato passando come singolo parametro la stringa del nome del modulo al quale vogliamo accedere.

La figura 2.2 contiene un esempio di utilizzo del metodo module() in AngularJS; nella prima riga viene specificato il modulo(MVCApp) a cui si fa riferimento nella porzione di codice delimitata dai tag html, perció l'oggetto cliente dal quale vengono lette le informazioni é contenuto nel controller chiamato Cliente definito nel modulo MVCApp.

2.4.3 Router in Angular.JS

Angular é un framework che supporta l'implementazione dei router per cambiare dinamicamente la view, al fine di facilitare la creazione di un'applicazione a pagina singola.

Nella figura 2.9 é possibile osservare il codice di una semplice spa che contiene tre pagine, nella quale l'utilizzatore puó muoversi utilizzando i link ad inizio pagina. Per utilizzare i router é necessario, per prima cosa, ag-

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs
/1.6.4/angular.min.js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs
/1.6.4/angular-route.js"></script>
<body ng-app="myApp">
<p><a href="#/!">Main</a></p>
<a href="#!londra">City 1</a>
<a href="#!parigi">City 2</a>
<div ng-view></div>
<script>
var app = angular.module("myApp", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.htm"
    })
    .when("/londra", {
        templateUrl : "london.htm"
    })
    .when("/parigi", {
        templateUrl : "paris.htm"
    });
});
</script>
</body>
</html>
```

Main

City 1 City 2

Main

Figura 2.9: Il codice rappresenta una semplice spa implementata in Angular con l'utilizzo dei router

Name	Status	Type	Initiator	Size	Time	Waterfall 400.00 ms
angular.min.js	200	script	tryit.asp?filename...	57.5 KB	316 ms	
angular-route.js	200	script	tryit.asp?filename...	11.6 KB	409 ms	
main.htm	200	xhr	VM2026 angular,...	208 B	47 ms	

Figura 2.10: La tabella rappresenta le risorse di cui l'applicazione nella figura 2.9 necessita nella fase iniziale

giungere la relativa dipendenza (angular-route.js), dopo di che é possibile utilizzare il modulo ng-Route che si occupa di visualizzare le pagine dell'applicazione senza effettuare operazioni di refresh. Nella fase di creazione dei link viene specificato il percorso a cui indirizzano il browser nel caso in cui l'utente clicchi su di essi, il router deve essere configurato per intercettare questi percorsi tramite il metodo $routerProvider.when(), nel quale é possibile specificare il percorso da intercettare e il file html da caricare.

Nella figura 2.10 si puó notare l'ordine con cui il browser scarica le risorse necessarie all'applicazione nella sua fase iniziale; non sono presenti i file html di londra e parigi perché in questa fase non sono necessari.

L'immagine 2.11 dimostra come alla pressione del link da parte dell'utente venga scaricato il file html contenente le informazioni da mostrare, senza causare un refresh di tutta la pagina; tutte queste operazioni sono gestite dal router precedentemente configurato. Con questa strategia l'utente é in grado di muoversi all'interno dell'applicazione anche tramite i tasti indietro e avanti contenuti nel browser, dal momento che vengono aggiunte nella cronologia dal router.

Nella figura 2.12 sono state scaricate tutte le risorse di cui l'applicazione necessita, a questo punto non é piú necessaria alcuna connessione ad internet e l'applicazione é in grado di funzionare anche in modalitá offline.

Main

City 1 City 2

London

London is the capital city of England.

It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

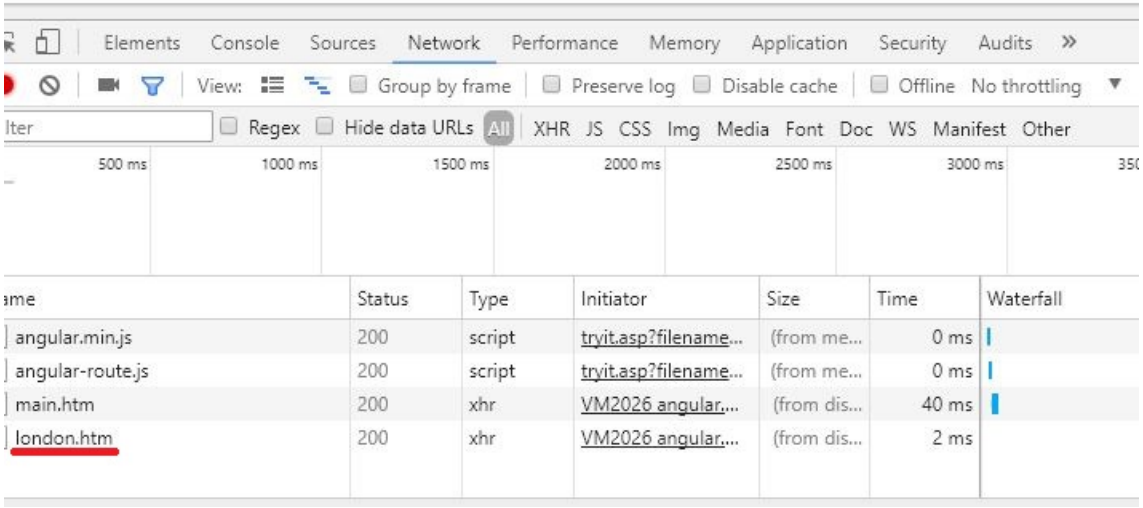

Figura 2.11: Da fare

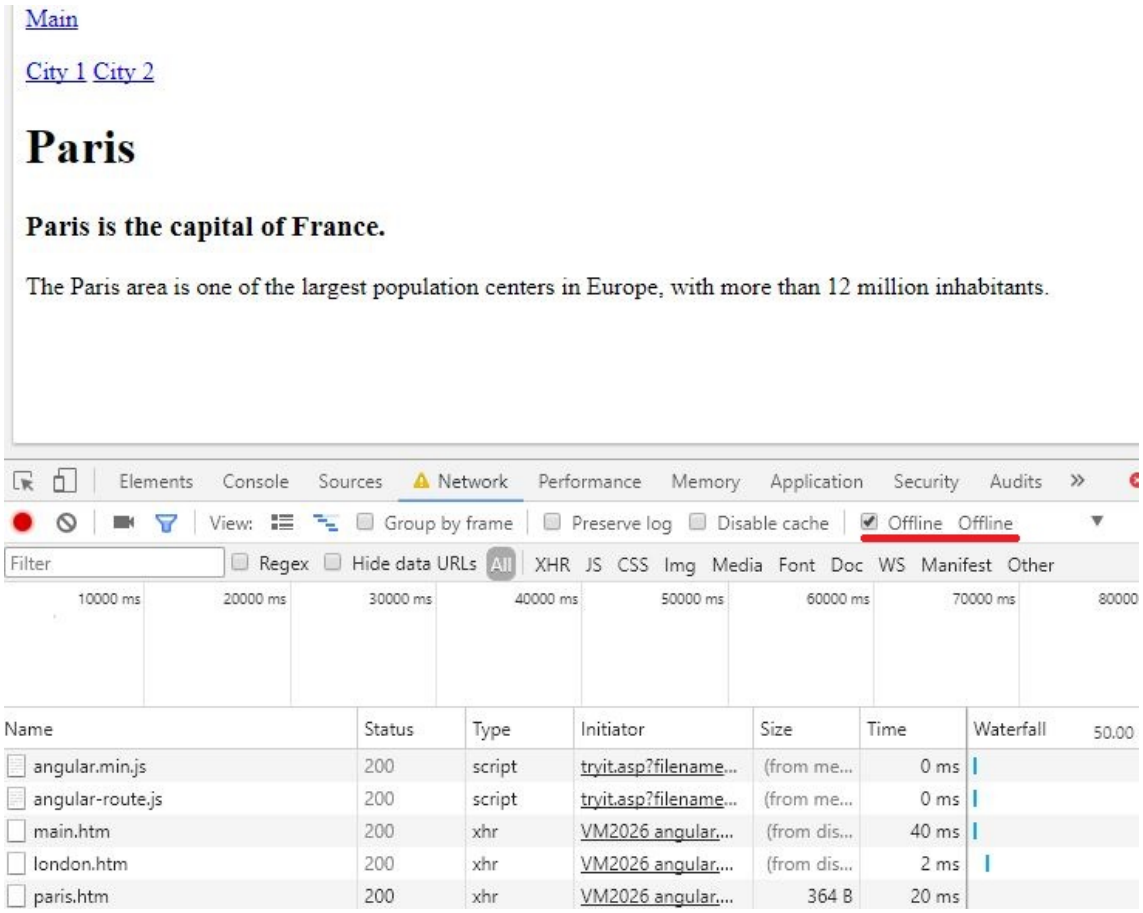

Figura 2.12: Da fare

Capitolo 3

Oltre JavaScript - Nuovi Linguaggi per lo sviluppo di Web App Complesse

Dal momento che il linguaggio JavaScript é risultato troppo limitato per lo sviluppo delle applicazioni web che negli anni sono diventate sempre piú grandi e complesse, sono stati sviluppati dei framework con l'obiettivo di aiutare lo sviluppatore nella gestione e nell'organizzazione del codice. Un'altra valida alternativa per il programmatore é servirsi di nuovi linguaggi di programmazione piú evoluti di JavaScript, i quali permettono al codice scritto di essere convertito in semplice JavaScript.Quest'ultimo viene quindi utilizzato come una sorta di codice assembly su cui sviluppare applicazioni con linguaggi di livello piú alto. I primi a proporre un linguaggio di questo tipo sono stati Google (Dart) e Microsoft (Typescript).

3.1 Dart

Dart, presentato per la prima volta ad ottobre del 2011, é un linguaggio di programmazione per il web sviluppato da Google con l'obbiettivo di sostituire javascript nello sviluppo di applicazioni basate sul web. Il linguaggio Javascript infatti ha alcune limitazioni impossibili da risolvere evolvendolo ulteriormente ed il team di Google ha deciso di optare per la creazione di un linguaggio nuovo e piú moderno, in grado di risolvere queste limitazioni e di offrire prestazioni migliori.

```
class CoppiaDiNumeri{
 num x, y;
 //esempio Named constructor
 CoppiaDiNumeri.ugualiTraloro(num p) : x = p, y = p;
 CoppiaDiNumeri.aZero() : x = 0, y = 0;
 CoppiaDiNumeri(this.x, this.y);
}

void main() {
  var coppia = new CoppiaDiNumeri.aZero();
}
```

Figura 3.1: Il codice mostra l'utilizzo del named costructor in Dart

Le caratteristiche piú importanti di Dart sono le seguenti:

1. É un linguaggio orientato agli oggetti di tipo Class-Based
2. Supporta l'ereditarietá, ma solo da una classe. Non supporta quindi l'ereditarietá multipla.
3. Permette la scrittura di codice non tipizzato.
4. Rende disponibili due modalitá di esecuzione, una per facilitare il debug e l'altra per migliorare le prestazioni.

3.1.1 Costruttori

Dart é un linguaggio che supporta le classi, di conseguenza prevede l'utilizzo di un costruttore; in particolare ammette il named constructor e il factory constructor. Questo linguaggio non supporta l'overloading, di conseguenza, al fine di poter definire piú di un costruttore all'interno di una classe, é stato definito il named constructor. Nella figura 3.2 si puó notare come sia possibile istanziare l'oggetto CoppiaDiNumeri anche attraverso il costruttore CoppiaDiNumeri.aZero() grazie al Named constructor.

Dart supporta nativamente l'utilizzo del Factory Pattern tramite il costruttore di tipo factory, nella figura **??** é riportata una classe che contiene

```
class Lettera {
  final String nome;
  static Map<String, Lettera> _cache = new Map<String, Lettera>();
  factory Lettera(String nome) {
    if (_cache.containsKey(nome)) {
      return _cache[nome];
    } else {
      final lettera = new Lettera._internal(nome);
      _cache[nome] = lettera;
      return lettera;
    }
  }
  Lettera._internal(this.nome);
}
main() {
  var letteraX = new Lettera('X');
  var altraLetteraX = new Lettera('X');
  print(identical(letteraX, altraLetteraX));  //stampa true
}
```

Figura 3.2: Il codice mostra l'utilizzo del factory costructor in Dart

un costruttore di questo tipo. In questa porzione di codice la variabile altraLetteraX dovrebbe istanziare un oggetto differente dalla variabile letteraX ma, dal momento che esiste giá un oggetto di tipo Lettera con il nome 'X', la seconda variabile conterrá l'oggetto creato dalla prima.

3.1.2 Funzioni

Esistono tre diverse notazioni per le funzioni nel linguaggio Dart: named functions, anonymous functions e arrow functions. Nella figura 3.3 é possibile osservare la sintassi delle tre funzioni: il primo metodo é molto simile agli altri linguaggi come Java, il secondo metodo prevede una funzione nella quale non é specificato il nome e il tipo di ritorno ed il terzo valuta l'espressione e restituisce il risultato di essa. Normalmente le funzioni anonime

```
\\funzione con nome

void saluti(String nome){
 final frase = 'ciao $nome';
 print(frase);
}

\\funzione anonima

window.on.click.add((event) {
print('Hai cliccato!');
})

\\funzione arrow

class Quadrato{
 num lato;
 bool isContains (num x, num y) => (x < lato) && (y < lato);
}
```

Figura 3.3: Il codice mostra i 3 tipi di funzioni ammessi da Dart

sono utilizzate per la gestione di eventi e callback e le arrow ogni volta che é possibile utilizzarle, nei restanti casi si utilizzano le funzioni con nome.

3.1.3 Integrazione con html

Tutti i linguaggi orientati al web hanno la necessitá di poter interagiore con le pagine html, nel caso di Dart é possibile inserire il codice dentro al tag script, settando il valore dell'attributo type a "'application/dart"'. L'importazione di script esterni é permessa grazie al comando #surce, mentre le librerie possono essere incluse con #import. Nella figura **??** si puó osservare un esempio di integrazione di Dart in un file html e di come sia possibile importare uno script esterno.

```
<html>
 <body>
  <script type='application/dart'>
   #source(CiaoMondo.dart)
   void main() {
    hello('Dart integrato in HTML');
   }
  </script>
  <div id="message"></div>
 </body>
</html>
```

Figura 3.4: Il codice mostra l'integrazione di Dart in un file html

3.1.4 Compilazione

Nel linguaggio Dart, il codice puó essere compilato in due diverse modalitá: controllata o di produzione. Nel primo caso, vengono eseguiti maggiori controlli ed é consigliato usarla in fase di sviluppo e test. Con la modalitá produzione, invece, Dart compila il codice con la migliore efficienza possibile, ignorando alcuni comandi utili solo in fase di progettazione (es specificare il tipo di variabile) per ottenere un prodotto piú performante; questa modalitá é consigliata per la creazione del prodotto finale. Per poter essere eseguito sfruttando al massimo le sue performance, Dart necessita di una particolare virtual machine all'interno del browser, questo lo ha reso un linguaggio meno versatile rispetto alla concorrenza, dal momento che questa VM é poco diffusa. In alternativa é possibile convertire il codice in puro JavaScript, al costo di una diminuizione delle prestazioni.

3.2 TypeScript

TypeScript é un linguaggio di scripting proposto da Microsoft nel 2012, con l'obbiettivo di risolvere le limitazioni del linguaggio JavaScript. A differenza di Dart, si pone come superset di JavaScript, é quindi pienamente retrocompatibile con il linguaggio di scripting, garantendo agli sviluppatori

```
class Saluti {
    nome: string;
    constructor(nome: string) {
        this.nome = nome;
    }
    saluta() {
        return "Ciao, " + this.nome;
    }
}

class SalutiFormali extends Saluti{
    constructor(nome: string) {
        super ("Sig. "+nome);
    }
}

let saluti = new SalutiFormali("Franco");
alert(saluti.saluta());
```

Figura 3.5: Esempio di utilizzo di classi e ereditarietá in TypeScript.

bassi tempi d'apprendimento e fornendo la possibilitá di riutilizzare codice e librerie scritte in JavaScript.

Tra le caratteristiche piú interessanti di questo linguaggio emergono le seguenti:

1. Ereditarietá e polimorfismo, sono possibili grazie alla natura del linguaggio che ammette le classi.

2. Compatibilitá con tutte le librerie JavaScript.

3. Possibilitá di utilizzare la tipizzazione statica opzionale.

4. Supporto nativo alla logica dei moduli.

Anche se piú recente rispetto a Dart, ha avuto subito un enorme successo grazie alla sua sintassi molto simile a JavaScript.

```
module Modulo {
    export class Saluti {
    nome: string;
    constructor(nome: string) {
        this.nome = nome;
    }
    saluta() {
        return "Ciao, " + this.nome;
    }
}
}
var saluto = new Modulo.Saluti("Franco");
alert(saluto.saluta());
```

Figura 3.6: Il codice mostra l'utilizzo dei moduli in Typescript.

3.2.1 Classi

Ammettendo le classi, TypeScript supporta di conseguenza anche il concetto di ereditá; nella figura 3.5 si puó vedere come la classe SalutiFormali estenda Saluti, di conseguenza ne eredita il metodo saluta(). Dal codice riportato nell'esempio si puó notare come TypeScript, ammettendo le classi, semplifichi notevolmente la scrittura del codice il quale, se scritto in puro JavaScript tramite i prototipi, sarebbe molto piú complesso.

3.2.2 Moduli

Una funzionalitá molto apprezzata di TypeScript é il supporto nativo ai moduli, molto utili nella strutturazione del codice. Nell'esempio della figura 3.6 si puó osservare un esempio di modulo; per poter essere contenuta all'interno di esso, la classe Saluti deve essere preceduta da export, dopo di che é possibile accedervi dall'esterno utilizzando Modulo.Saluti.

3.3 React.js

React é una libreria Javascript realizzata da Facebook e Istagram con l'obbiettivo di facilitare la creazione di un'applicazione a singola pagina, fornendo una struttura che la divide in componenti dinamici e riutilizzabili.

```
class Lista extends React.Component {
 aggiungiElemento(elemento){
  return <li>{elemento}</li>
 }
 render() {
    return (
      <div className="lista">
        <h1>lista</h1>
        <ul>
         {this.aggiungiElemento("pesche")}
         {this.aggiungiElemento("susine")}
         {this.aggiungiElemento("albicocche")}
        </ul>
      </div>
    );
  }
}
```

Figura 3.7: La figura contiene una classe in React.js.

Ogni componente React é una classe Javascript che estende la classe Component, la quale rappresenta un blocco atomico di codice HTML e le le sue eventuali componenti dinamiche. I componenti sono isolato tra di loro e questo aiuta notevolmente in fase di sviluppo ad avere un codice meglio organizzato e meno caotico, in oltre facilita il riutilizzo di porzioni di codice. Ogni componente é composto unicamente da metodi JavaScript tra cui il metodo render() che ritorna la parte di codice HTML da mostrare. A differenza di molti altri linguaggi e framework, React.js non usa pattern MV* e non permette la modifica diretta del DOM; ne fornisce uno virtuale sul quale possono essere cambiati elementi dinamicamente e React provvederà a modificare quello reale in maniera efficiente e veloce. Nonostante non si basi su pattern MV*, React é un linguaggio completo e performante; non implementa nativamente i router, ma sono numerose le librerie di terze parti che possono fornirli.

3.3.1 Classi

Le classi in React devono estendere React.Component e contenere il metodo render(). Come si puó osservare dall'immagine **??**, il metodo render ritorna le modifiche che dovranno essere effettuate nel file html; nell'esempio verrá

creata una lista contenente tre elementi. La loro creazione viene effettuata chiamando il metodo aggiungiElemento(), il quale restituisce il codice html corrispondente all'elemento da aggiungere nella lista.

3.4 Angular

Angular, comunemente chiamato Angular2+ per non confonderlo con AngularJS é un linguaggio basato su TypeScript per lo sviluppo di applicazioni basate sul web. La prima versione stabile fu rilasciata nel 2016 dal team che sviluppó AngularJS. La sintassi del nuovo linguaggio é completamente diversa dal suo predecessore e molti concetti chiave in AngularJS non sono piú supportati. Nel marzo 2017 Angular é stato aggiornato alla versione 4, per supportare le ultime versioni di TypeScript e per incrementare sensibilmente le prestazioni dell'applicazione.

3.4.1 Differenze tra AngularJS e Angular

Angular2+ adotta un approccio di User Interface di tipo component-based, perció il concetto di direttive di AngularJS é stato sostituito dai Component. Adottando TypeScript, questa nuova versione inserisce il concetto di classi e di conseguenza di ereditarietá, permette inoltre di includere nel progetto tutte le librerie di TypeScript.

Nella figura 3.8 sono presenti alcune parti di codice di due semplici applicazioni identiche scritte in Angular2+ e AngularJS; si puó notare come a sinistra non siano piú presenti il tag ng-controller, il controller e la variabile $scope, che vengono incapsulati nei componenti, definiti come una classe con il decoratore @Component. Un'altra differenza sono i router; nella prima versione si utilizzavano attraverso $routeProvider, mentre in Angular2+ si definiscono in un modulo esterno definendo il nome, il path ed il componente associato al percorso.

Figura 3.8: La figura contiene porzioni di codice in AngularJS e Angular2+.

Capitolo 4

Conclusioni

da fare

Ringraziamenti

Ringrazio la mia famiglia e gli amici per avermi sostenuto.

Bibliografia