

Corso Web MVC

Introduzione

Emanuele Galli

www.linkedin.com/in/egalli/

Informatica

- Informatique: information automatique
 - Trattamento automatico dell'informazione
- Computer Science
 - Studio dei computer e come usarli per risolvere problemi in maniera corretta ed efficiente

Computer

- Processa informazioni
- Accetta input
- Genera output
- Programmabile
- Non è limitato a uno specifico tipo di problemi

La virtual machine è un software che fa finta di essere hardware. si installa sul pc e può avviare un sistema operativo diverso da quello presente sul pc.

Hardware, Software, Firmware

- Hardware

- Componenti elettroniche usate nel computer
- Disco fisso, mouse, ...

- Software

codice utilizzabile dall'hardware, è processabile per giungere ad uno scopo. Quando lo chiamiamo processo vuol dire che è un programma in esecuzione.

- Programma

- Algoritmo scritto usando un linguaggio di programmazione
- Codice utilizzabile dall'hardware

- Processo

- Programma in esecuzione

- Word processor, editor, browser, ...

RAM: è l'area di memoria su cui la CPU agisce, in stretto contatto con questa, è lì che ci sono le variabili i programmi. è tipicamente nella scheda madre. è modificabile e volatile, se non salviamo i dati prima di spegnere il pc li perdiamo salvo il programma effettui salvataggi automatici.
ROM: read only memory. Dentro ci sono dei dati e programmi non più modificabili. Qui dentro c'è il Firmware. Anche questa sta nella scheda madre. Cioè nell'hardware.
EEPROM: si usa più della ROM. segue lo stesso principio di questa ma è riprogrammabile mediante processi complicati.
MEMORIA DI MASSA: disco o supporto esterno. è hardware.
BUT: è il processo di avviamento della macchina, del sistema operativo. è lo startup
BIOS: è quello che permette il but. fa parte del firmware

- Firmware

collante tra hardware e software. DNA della macchina.


- Programma integrato in componenti elettroniche del computer (ROM, EEPROM)

- UEFI / BIOS: avvio del computer
- Avvio e interfaccia tra componenti e computer

Sistema Operativo




L'istruzione è la parte minima di un algoritmo. è un comando, un ordine. deve essere ordinata e completa, in modo tale che la CPU la comprenda e agisca. L'inizializzazione, la definizione sono dei tipi di istruzione. Il loop ad esempio è un'istruzione che comprende non solo il comando ma anche il contenuto di questo comando.

- Insieme di programmi di base
 - Rende disponibile le risorse del computer
 - All'utente finale mediante interfacce
 - CLI (Command Line Interface) / GUI (Graphic User Interface)
Interfaccia a carattere. I comandi cambiano in base al sistema operativo. è un modo per accedere ai file
 - Agli applicativi
 - Facilità d'uso vs efficienza
- Gestione delle risorse: 
 - Sono presentate per mezzo di astrazioni
 - File System In un File System ci sono Folder e File. Il Folder è qualcosa che mi può contenere sia file che altri folder. è una cartelletta. Il file è unità minima di memoria di massa, è la quantità minima di memoria che possiamo utilizzare. Ha una particolarità, possiede un nome. Alcuni file sono modificabili altri possono essere modificati solo dal sistema operativo. Il File System è basato su c:. C: è il nome della periferica della memoria di massa è usato per periferiche dvd, E per chiavette usb ecc... Il File System è un albero la cui Root è C:/ e da cui poi discendono sistema operativo (Windows) ecc...
 - Ne controlla e coordina l'uso da parte dei programmi
- Semplifica la gestione del computer, lo sviluppo e l'uso dei programmi

Problem solving

la definizione è la parte preparatoria del problema.

- Definire chiaramente le **specifiche** del problema
 - Es: calcolo della radice quadrata. Input? Output?
 - Vanno eliminate le possibili ambiguità
- Trovare un **algoritmo** che lo risolva 
- Implementare correttamente la soluzione con un linguaggio di programmazione
- Eseguire il programma con l'input corretto, in modo da ottenere l'output corretto

GIGO: garbage input garbage output
non basta avere un bell'algoritmo che funziona bene serve anche saperlo gestire bene.

Algoritmo

- Sequenza di istruzioni che garantisce di dare il risultato di un certo problema
 - Ordinata, esecuzione sequenziale (con ripetizioni)
 - Operazioni ben definite ed effettivamente eseguibili
 - Completabile in tempo finito
- Definito in linguaggio umano ma artificiale
 - Non può contenere ambiguità
 - Deve essere traducibile in un linguaggio comprensibile dalla macchina

Le basi dell'informatica

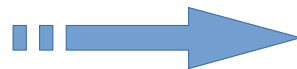
- Matematica

- L'algebra di George Boole ~1850

- Notazione binaria

- La macchina di Alan Turing ~1930

- Risposta all'Entscheidungsproblem (problema della decisione) posto da David Hilbert
 - Linguaggi di programmazione Turing-completi



- Ingegneria

- La macchina di John von Neumann ~1940

- Descrizione dell'architettura tuttora usata nei computer: Input, Output, Memoria, CPU

CPU su cui ci sarà il BUS di sistema che transiterà per trasformare gli INPUT in OUTPUT. La CPU estrapola i dati dalla memoria ed è il cervello che trafora il codice in flusso di esecuzione. I dati sono le scariche elettriche che immetto nel mio circuito che verranno fuori tramite output.

Il linguaggio turing completo è quello di programmazione. HTML non lo è ad esempio. Le caratt. sono:
1. riceve input > output emula quindi la macchina di turing
2. prende decisioni. Ha bisogno di un'istruzione condizionale if - else e ci deve essere il modo di realizzarla
3. for/while
4. Block cioè ripetere stesse operazioni
5. Presenza di istruzioni e variabili


Algebra Booleana

- Due valori
 - false (0)
 - true (1)
- Tre operazioni fondamentali
 - AND (congiunzione)
 - OR (disgiunzione inclusiva)
 - NOT (negazione)

A	B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

A	NOT
0	1
1	0

Linguaggi di programmazione



- Linguaggio macchina
 - È il linguaggio naturale di un dato computer
 - Ogni hardware può averne uno suo specifico
 - Istruzioni e dati sono espressi con sequenze di 0 e 1
 - Estremamente difficili per l'uso umano
- Linguaggi Assembly  Digita qui il testo
 - Si usano abbreviazioni in inglese per le istruzioni
 - Più comprensibile agli umani, incomprensibile alle macchine
 - Appositi programmi (assembler) li convertono in linguaggio macchina

La variabile è il concetto logico con cui io esprimo al programma la preferenza dei dati su cui voglio lavorare. Per poterla referenziare bisogna darle un nome.

La costante è invece un blocchetto in memoria che non voglio però cambiare.

ALIAS = VAR A: 3 e VAR C: A in ultima analisi le due variabili si riferiscono alla stessa cella di memoria.

Variabile

- Locazione di memoria associata a un nome, contiene un valore
- Costante: non può essere modificata dopo la sua inizializzazione
- Una singola locazione di memoria può essere associata a diverse variabili (alias)
- Supporto a tipi di variabili da linguaggi di:
 - “basso livello” → legati all’architettura della macchina 
 - “alto livello” → tipi complessi 
 - script → runtime

Array è una particolare variabile che può contenere diversi valori dello stesso tipo. Su di esso si può agire solo sintatticamente. La stringa è sua parente ma ospita solo caratteri e si possono esprimere delle operazioni sematiche.

Ha l'aspetto di un blocco in cui i valori sono divisi in celle. L'array può essere definito anche un blocco indicizzato. Le operazioni per agire su di esso devono infatti tener presente la posizione (indice) in cui si trovano le variabili interne.

Di solito non lavoriamo su una sola variabile ma su più strettamente collegate. L'array indicata con `[]` è una collezione di valori all'interno di un'unica variabile. Tipicamente con l'array si prende un bloccone di memoria indicizzata: se voglio accedere ad un certo elemento devo indicare l'indice che rimanda alla posizione. Ci sono dei linguaggi di programmazione in cui il primo elemento ha indice 0. Java è un linguaggio che parte da 0. Se voglio accedere alla cella 1 mi basta sapere l'indirizzo della cella (per esempio `temperature`) alla 2 (`temperature +1`).

Array

- Struttura dati comune a molti linguaggi di programmazione
- Basata sul concetto matematico di vettore, nel senso di matrice monodimensionale
- Collezione di elementi (dello stesso tipo) identificati da un indice
 - Il primo elemento ha indice 0 in alcuni linguaggi, 1 in altri (e anche n in altri ancora)
- Gli elementi sono allocati in un blocco contiguo di memoria, il che permette accesso immediato via indice ai suoi elementi

Linguaggi di alto livello

- Molto più comprensibili degli assembly
- Termini inglesi e notazioni matematiche
- Possono essere espressi in forma
 - **imperativa**: si indica cosa deve fare la macchina Ho il controllo ferreo di quello che deve fare la CPU.
 - **dichiarativa**: si indica quale risultato si vuole ottenere

HTML non è un linguaggio di programmazione perché non è Turing completo, è del tipo dichiarativo. In questo caso è il compilatore che è lasciato libero di agire per dare i risultati.

- A seconda di come avviene l'esecuzione si parla di linguaggi

Il linguaggio C è un compilato.

sorgente
compilato
eseguibile

- **compilati**: conversione del codice in linguaggio macchina, ottenendo un programma eseguibile

Il linguaggio compilato è affidato al compilatore o al linker che trasforma il nostro file in linguaggio c (x.c) ad esempio in linguaggio macchina per un determinato tipo di macchina. È un linguaggio più lento rispetto al secondo, per essere letto altrove deve essere decompilato e non sempre quest'attività riesce bene.

Il codice scritto in alto livello è tipicamente chiamato source code. Anche l'assembly in realtà lo è. Resta fuori quello macchina.

sorgente
interprete

- **interpretati**: il codice viene eseguito da appositi programmi

Javascript fa parte di questa famiglia. Questo linguaggio può invece correre sui browser che permette di leggerlo su diversi dispositivi. In questo caso è il browser ad essere il compilatore.

La differenza tra i due casi è che nel secondo l'utente vede il linguaggio sorgente.

Istruzioni

1 • Operazioni sequenziali

I valori booleani di vero e falso servono per decidere le operazioni sequenziali e iterative. Restano fuori quelle sequenziali, sequenze di operazioni che la mia CPU deve eseguire. Operazioni sequenziali sono standard, sono le op. "normali"

- Chiedono al computer di eseguire un compito ben definito, poi si passa all'operazione successiva

2 • Operazioni condizionali

If (C>0) then c=c*2 else c=0 questa è la logica delle operazioni condizionali. cioè eseguire un codice piuttosto che un altro in base alla condizione di verità o falsità della premessa.

[vedere appunti giorno 3](#). Le condizionali determinano un branching nel codice.

- Si valuta una condizione, il risultato determina quale operazione seguente verrà eseguita

3 • Operazioni iterative

eseguire una stessa operazione su tutti gli elementi dell'array o su un certo intervallo. per esempio for (i=1...10) print (i)

Abbiamo un blocco e finché una condizione è vera continuiamo ad agire sul blocco alla stessa maniera.

Operazione
condizionale

- Richiede di ripetere un blocco di operazioni finché non si verifica una certa condizione – se ciò non accade: loop infinito

```
var as[] = input
var sum = 0 Inizializziamo la somma. Si parte da zero perchè è l'elemento neutro per la somma.
for ( i --> n ) per i che va da 0 a n. Sto loopando tutti gli elementi. Vogliamo avere una variabile di loop che mi permette di lavorare su tutti gli elementi.
  sum = sum + as [i] (componente iesima di as) Il somma riprende nel primo caso lo 0. poi memorizza il risultato che per la prima volta è 0+il primo numero dell'array.
  print (sum)
if (sum divisibile di 2)
  print ("pari")
else
  print ("dispari")
```

Flow chart vs Pseudo codice

- Diagrammi a blocchi – flow chart
 - L'algoritmo viene rappresentato con un grafo orientato dove i nodi sono le istruzioni
 - Inizio e fine con ellissi
 - Rettangoli per le operazioni sequenziali (o blocchi)
 - Esagoni o rombi per condizioni
- Pseudo codice Rappresenta l'architettura approssimativa.
 - L'algoritmo viene descritto usando l'approssimazione un linguaggio ad alto livello, si trascurano i dettagli, ci si focalizza sulla logica da implementare

Complessità degli algoritmi

- “O grande”, limite superiore della funzione asintotica Dalla complessità minore a quella maggiore. I computer quantistici dovrebbero eseguire operazioni $O(n^2)$ in $O(n \log n)$.
 - Costante $O(1)$ Il rapporto tempo/n (numero di input array) è lineare. L'input può crescere di dimensione ma il tempo di operazione su di esso rimane costante.
 - Logaritmica $O(\log n)$ In questo caso il tempo aumenta come in una curva logaritmica ma non in maniera sensibile. Il costo logaritmico è ricavabile dagli alberi BST. Dimezzo il tempo di ricerca perchè l'albero già di per sé mi divide i valori in minori e maggiori rispetto alla radice. Si usa soprattutto del sorting.
 - Lineare $O(n)$ Data un array con n elementi devo trovare un det valore. più è lungo l'array più mi posso aspettare che il costo di ricerca aumenti. in questo caso l'algoritmo cerca elemento per elemento. [Digita qui il testo](#)
 - Linearitmico $O(n \log n)$ il costo è un pò peggio della lineare. complessità di questo tipo hanno a che fare con il sorting, il mettere in ordine. Quindi devo ordinare i valori come se dovessero essere messi su un albero.
 - Quadratica $O(n^2)$ – Polinomiale $O(n^c)$ Da questo punto la complessità aumenta di livello per cui o rinunciamo all'impresa, o attendiamo tempi molto lunghi o usiamo pochi dati.
 - Esponenziale $O(c^n)$
 - Fattoriale $O(n!)$
- Tempo e spazio Quanto tempo ci mette il mio algoritmo per essere eseguito e quanto spazio mi occupa. Queste sono le due variabili che più mi interessano. Tipicamente al crescere di una variabile corrisponde il diminuire dell'altro.
- Caso migliore, peggiore, medio Tipicamente ci importa il caso medio o il caso peggiore. In questo caso perchè è il caso da evitare.

Algoritmi di ordinamento: SORTING

Ordinare significa:


- Applicazione di una relazione d'ordine a una lista di dati
 - Naturale → crescente (alfabetico, numerico)
- Utile per migliorare
 - l'efficienza di altri algoritmi
 - La leggibilità (per gli umani) dei dati
- Complessità temporale
 - $O(n^2)$: algoritmi naive
 - $O(n \log n)$: dimostrato ottimale per algoritmi basati su confronto
 - $O(n)$: casi o uso di tecniche particolari

$n \log n$ viene fuori dal fatto che operiamo sugli alberi per cui la complessità lineare del primo ramo si moltiplica per $\log n$. c'è un teorema complicato che dimostra ciò.

Questo è il costo per lavorare su dati già ordinati. Il mio algoritmo diventa sempre più veloce.

Ingegneria del software

L'idea è quella di dividere ben le varie fasi di sviluppo del software

- **Approccio sistematico alla creazione del software**
 - Struttura, documentazione, milestones, comunicazione e interazione tra partecipanti
- **Analisi dei requisiti** è per il progetto. Prima di scrivere una riga di codice bisogna discutere per rigorizzare il progetto.
 - Formalizzazione dell'idea di partenza, analisi costi e usabilità del prodotto atteso
- **Progettazione** definire la struttura del codice, la MVC (pattern strutturale che divide in pezzi la progettazione del codice: il modello, la view e il controller dove vanno i codici), definire i requisiti dell'hardware e il software. I progettisti sanno come gestire le cose perchè hanno esperienza nella programmazione, scrivono le specifiche e poi passano il lavoro ai programmatori.
 - Struttura complessiva del codice, definizione architetturale
 - Progetto di dettaglio, più vicino alla codifica ma usando pseudo codice o flow chart
- **Sviluppo** Ricevute le specifiche dal progettista bisogna trasformarle in codice. Questo deve essere testato per capire se risponde davvero alle specifiche che sono state date.
 - Scrittura effettiva del codice, e verifica del suo funzionamento via **unit test** 
- **Manutenzione** L'aggiornamento è una manutenzione del codice ad esempio.
 - Modifica dei requisiti esistenti, bug fixing

Unit Test

- Verificano la correttezza di una singola “unità” di codice
 - Mostrano che i requisiti sono rispettati
- Verifica
 - Casi base (positivi e negativi)
 - Casi limite Tipicamente gli errori si fanno nei casi limite. Con la pratica si capisce quali sono i casi più importanti da testare.
- Ci si aspetta che siano
 - Ripetibili: non ci devono essere variazioni nei risultati
 - Semplici: facile comprensione ed esecuzione
 - E che offrano una elevata copertura del codice



Analisi dei requisiti

Progettazione

Manutenzione

L'Acceptance è fatto dal cliente.
Può però essere che
che tutto funzioni bene all'inizio e
vada poi storto dopo. In questo caso
si entra nella manutenzione. Se il risultato
è negativo si richiede alla softwarehouse
un altro programma messo poi a paragone
con il primo.

Sviluppo

Unit test

Di solito viene fatto da una persona addetta, il tester, che di mestiere fa solo quello. Se l'integration test funziona si fa il rilascio testato mediante l'Acceptance.

Integration test

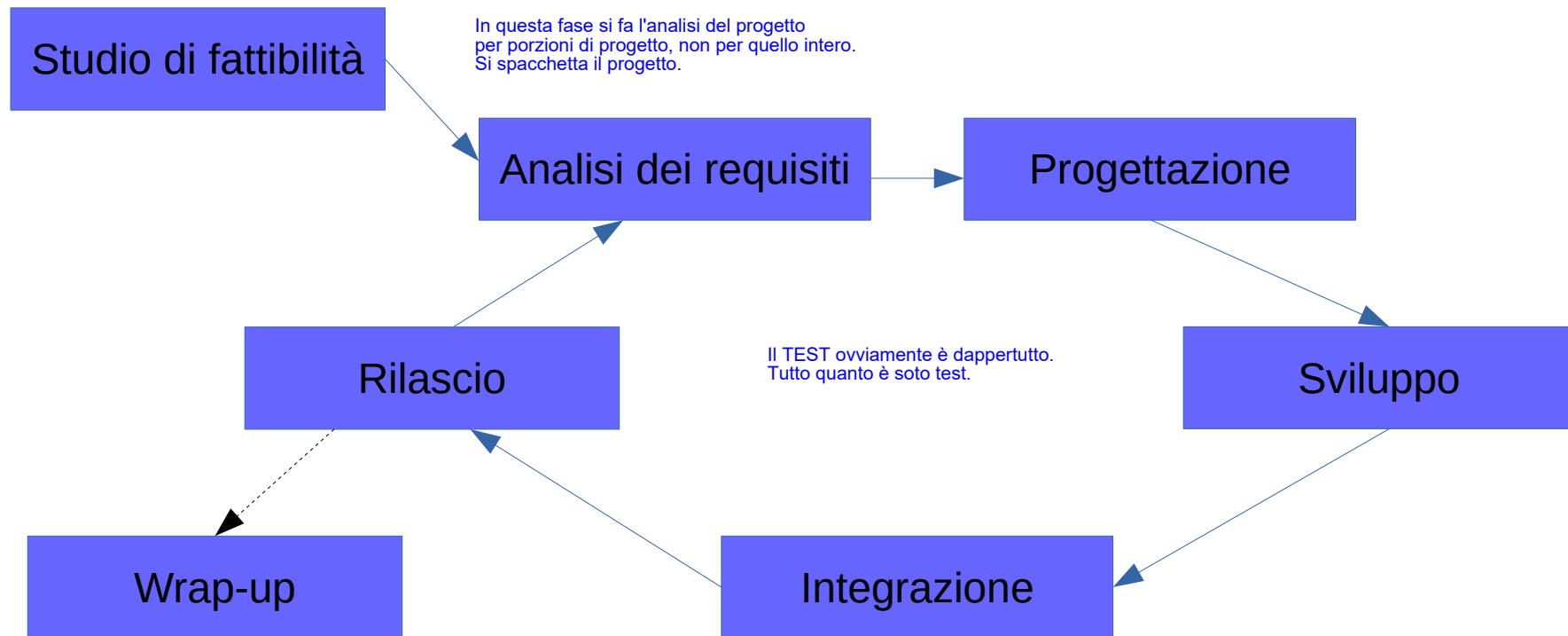
Acceptance test

Integrazione

Rilascio

Modello opposto a quello precedente. Questo è meno strutturato. È un modello startup perché è in continua evoluzione, non ci sono scadenze fisse, è difficile che si facciano previsioni e tutti comunicano con tutti. però è raro che si sbagli direzione.

Modello agile



Sarebbe il rilascio finale che di solito però non c'è.

Software Developer

- Front End Developer è la view. è quello che va a finire sulla macchina del cliente o sul cellulare.
 - Pagine web, interazione con l'utente
 - HTML, CSS, JavaScript
 - User Experience (UX)
- Back End Developer
 - Logica applicativa
 - controller Java, C/C++, Python, JavaScript, model SQL, ...
 - JavaEE, Spring, Node, DBMS, ...
- Full Stack Developer Questa è la figura più ricercata dal mercato.
 - Sintesi delle due figure precedenti