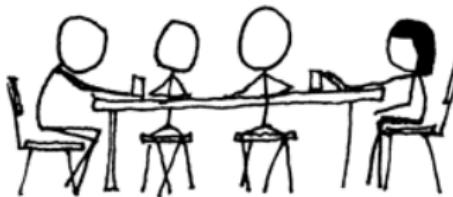


YOUR PARTY ENTERS THE TAVERN.

I GATHER EVERYONE AROUND
A TABLE. I HAVE THE ELVES
START WHITTLING DICE AND
GET OUT SOME PARCHMENT
FOR CHARACTER SHEETS.

HEY, NO RECURSING.



Applicazioni Web 2010/11

Lezione 6: Esempio: TODO-List online multiutente

Matteo Vaccari

<http://matteo.vaccari.name/>
matteo.vaccari@uninsubria.it



(cc) Matteo Vaccari. Published in Italy.
Attribution – Non commercial – Share alike 2.5

tadalist.com

Things to pack for my trip

- Sunscreen
- Sandals (maybe the green ones)
- Toothpaste
- Socks, socks, and more socks
- Thomas Jefferson biography
- 8 t-shirts
- 4 pairs of shorts
- Blue suit and orange tie
- Black and brown belt

Si parte da una lista di liste

My Lists [Create a new list](#)

MY LISTS

- [Cose da imparare per la lezione6](#) — 3 left

Si crea una nuova lista

Name your new list (Ex: "Things I need to do today")

Cose da imparare per la lezione6

[Create this list](#) or [Cancel](#)

Si crea un elemento della lista

Cose da imparare per la lezione 6

Architettura Model–View–Controller

+ Add this item

[Close](#)

Eccetera...

Cose da imparare per la lezione 6

- Architettura Model-View-Controller
- Come progettare un'applicazione
- Come organizzare il codice in un'applicazione

+ Add this item

[Close](#)

Si possono marcare come "fatti"

Cose da imparare per la lezione 6

- Architettura Model-View-Controller
- Come progettare un'applicazione

+ Add this item

[Close](#)

- Come organizzare il codice in un'applicazione

Cose che abbiamo tralasciato

- Registrazione utenti
- Autenticazione
- Autorizzazione
- Condivisione liste
- Riordinare gli elementi
- ...

Come progettare un'applicazione web

- Lista dei requisiti funzionali
- Lista dei requisiti *non* funzionali
- Progettare il flusso delle pagine
- Progettare le url
- Progettare la base di dati

+ Add this item

[Close](#)

Requisiti funzionali

- Creare liste
- Aggiungere cose a una lista
- Marcare una cosa come “fatta”
- Tenere separate le liste per utente

Requisiti non funzionali I

Tag: Robustezza.NoDowntime

Scale: percentuale di tempo in un mese in cui il sistema è disponibile

Measure: poll automatico ogni 5 minuti

Goal: > 99.8% (max downtime 1h 26' in un mese)

Tag: Robustezza.NoErrors

Scale: percentuale transazioni HTTP con esito 5xx

Measure: esaminare i log del frontend

Status: [Settimana 5-8/11/2010, Url=Postpayment]: 0,7%

Status: [Settimana 5-9/11/2010, Url=Tutte]: 0,12%

Goal: [Url=Tutte]: < 0.01%

Requisiti non funzionali II

Tag: Latenza

Scale: tempo di elaborazione

Measure: tempo di risposta riportato sui log del frontend per le operazioni che non coinvolgono accesso a server esterni a Setefi

Status: <?>

Goal: [Tempo medio, Url=tutte]: < 1500ms, stddev < 1000ms

Tag: Sicurezza.Accessi

Description: Il sistema deve resistere a tentativi di accesso non autorizzato, compreso privilege escalation.

Test: Analisi del design da parte di un esperto di sicurezza

Test: Tentativo di penetrazione da parte di un esperto di sicurezza

Requisiti non funzionali III

Tag: Sicurezza.DatiSensibili

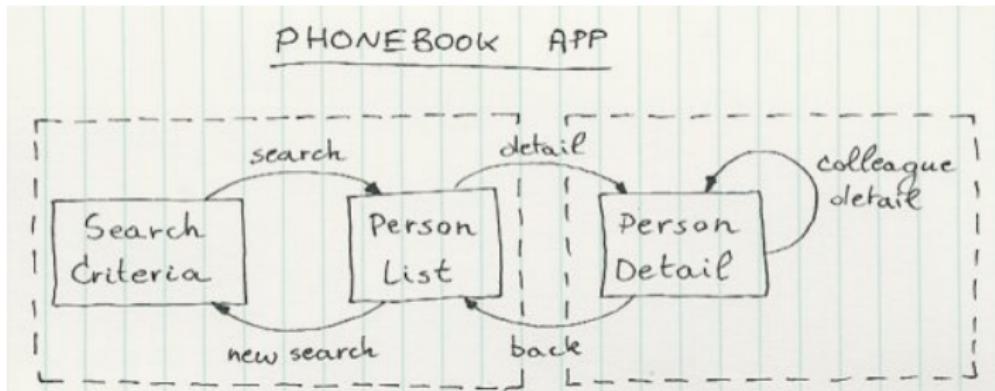
Description: I dati sensibili devono essere conservati in maniera crittografata; non devono essere esposti sull'interfaccia utente.

Test: Analisi del design da parte di un esperto di sicurezza

Tag: Trasparenza

Description: Lo stato del sistema deve essere visibile al personale tecnico. Devono essere in ogni momento noti: stato (up/down), tempi di risposta, numero di transazioni accettate e rifiutate

Il flusso delle pagine: esempio informale



<http://www.ervacon.com/products/swf/intro/index.html>

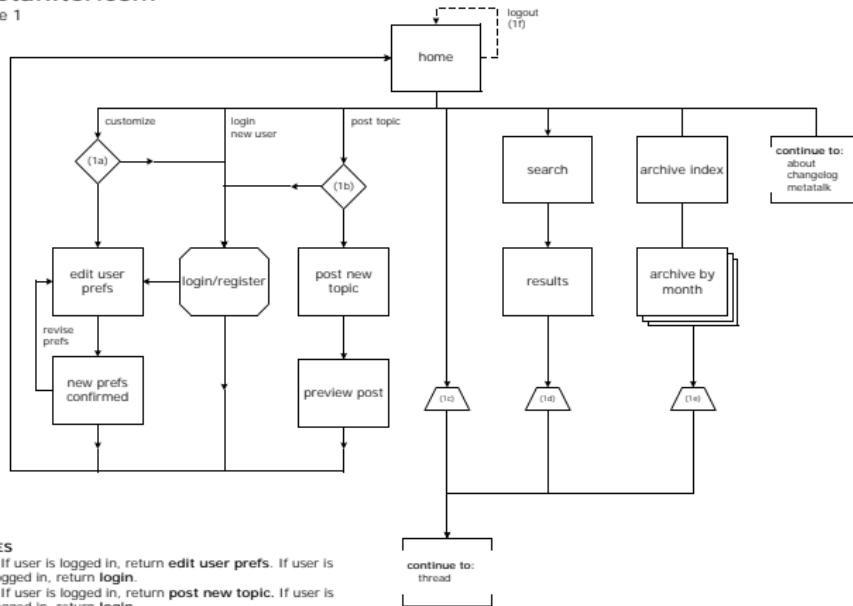
Il flusso delle pagine: esempio informale



Il flusso delle pagine: esempio formale (Garrett)

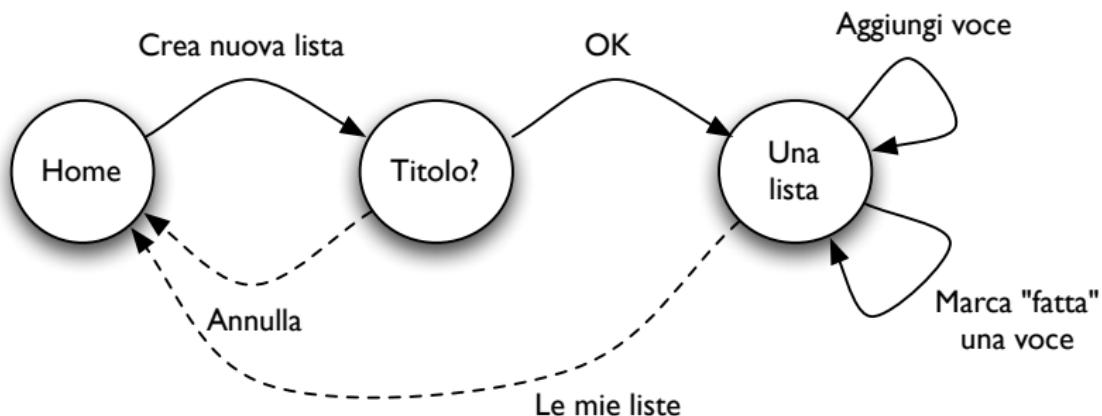
metafilter.com

page 1

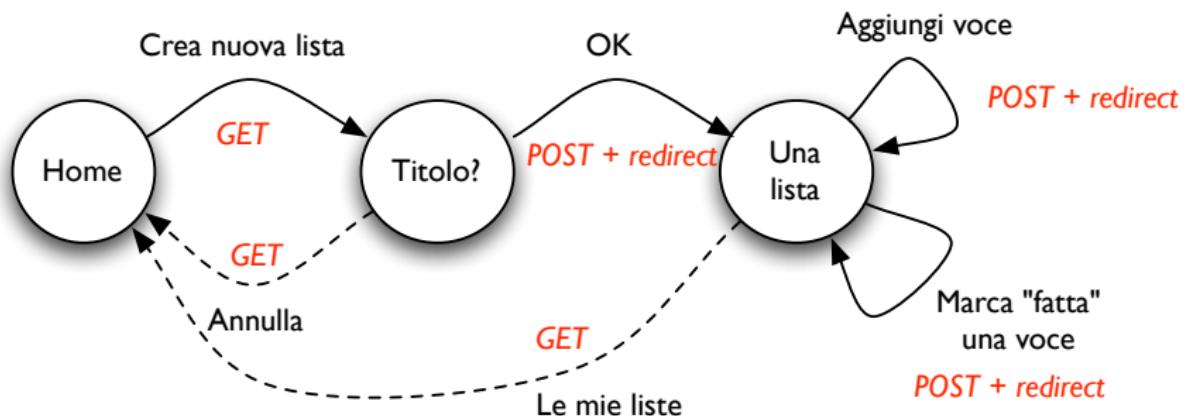


Jesse James Garrett
<http://www.jjj.net/ia/visvocab/>

Il flusso delle pagine (i)



Il flusso delle pagine (ii)



La struttura delle url (i)

`http://example.org/matteo`

`http://example.org/matteo/lists`

`http://example.org/matteo/lists/new`

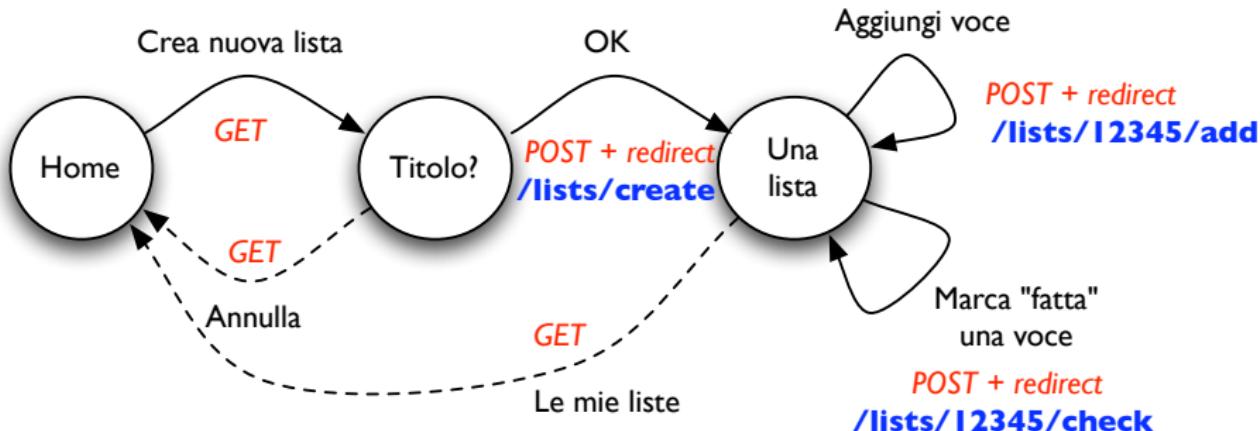
`http://example.org/matteo/lists/7276532`

La struttura delle url (ii)

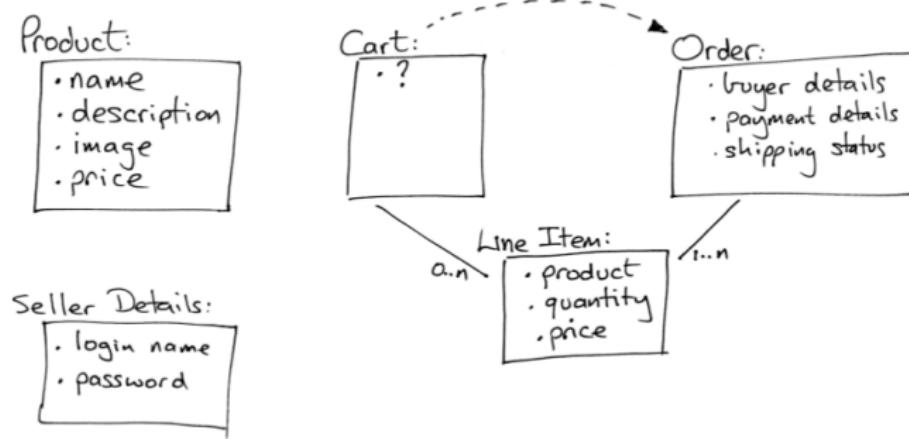
/lists

/lists/new

/lists/12345



La struttura dati: esempio informale



David Heinemeier Hansson, *Agile Web Development With Rails 3rd ed.*

La struttura dati

```
data
└── lists
    ├── list-14141.txt
    ├── list-20193.txt
    ├── list-21313.txt
    └── list-2133.txt
```

Un documento che rappresenta una lista

Title: Come progettare un'applicazione web

Author: matteo

Created: 2011-12-10 23:12:33

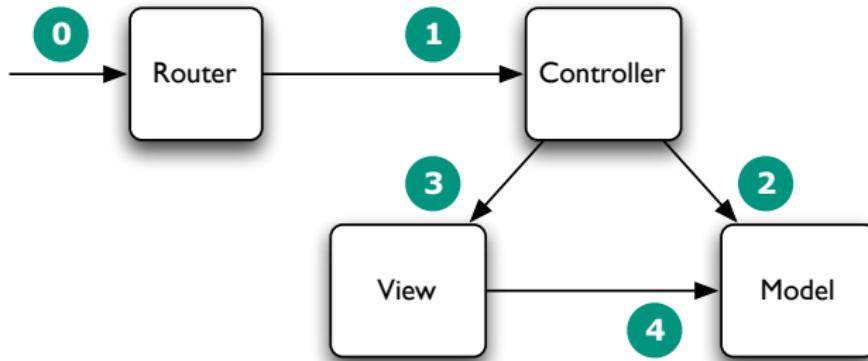
Updated: 2011-12-11 11:23:22

Progettare il flusso delle pagine

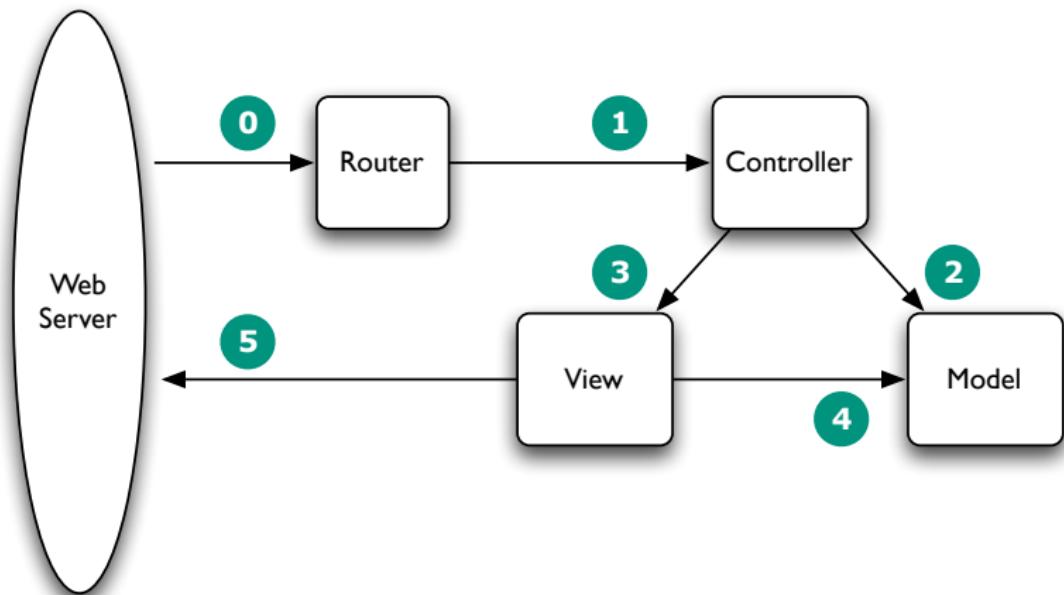
Progettare le url

[X] Progettare la base di dati

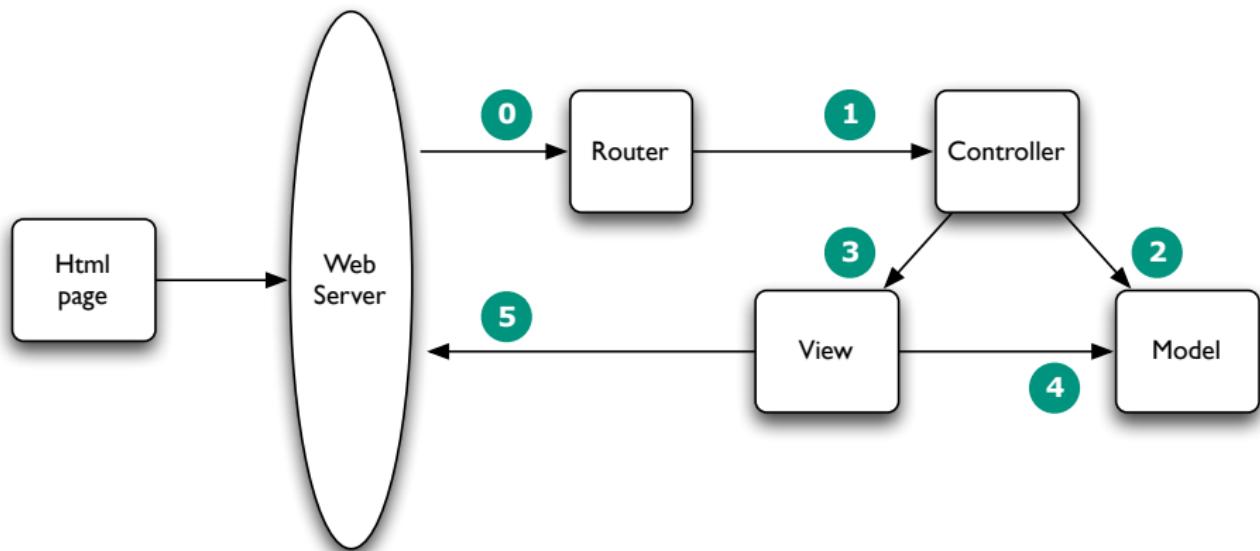
Model, View, Controller



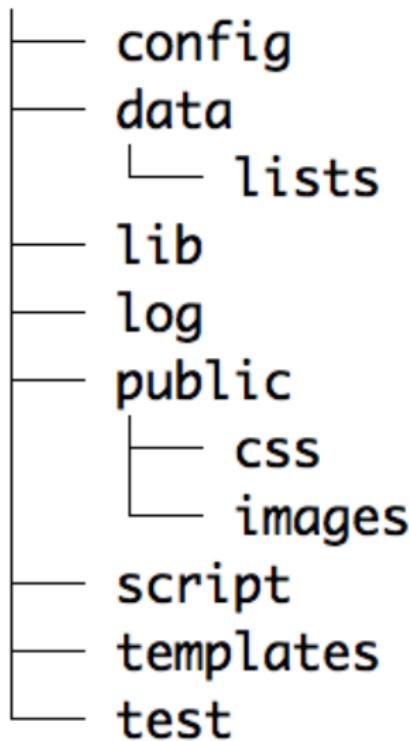
Model, View, Controller



Model, View, Controller



Organizzazione delle directory



Virtual hosts

File /etc/apache2/sites-available/todo-list

```
<VirtualHost *:80>
    ServerName todo-list
    ErrorLog /home/matteo/todo-list/log/error_log
    CustomLog /home/matteo/todo-list/log/access_log common
    DocumentRoot /home/matteo/todo-list/public
    <Directory /home/matteo/todo-list/public>
        AllowOverride All
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

Configurare /etc/hosts

127.0.0.1 todo-list

Configurare Apache localmente

File todo-list/public/.htaccess

```
AddHandler cgi-script .cgi
Options +FollowSymLinks +ExecCGI +Indexes -Multiviews

# If a request is *not* for an existing file, redirect it to the
# dispatch.cgi script
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ route.cgi [QSA,L]
```