UnCommon Web

ovvero: come imparai a non preoccuparmi e ad amare il web.

- Marco Baringer <mb@bese.it>
- http://common-lisp.net/project/ucw

Il Problema

- Lo strumento non ci permette di esprimere, direttamente, quello che pensiamo.

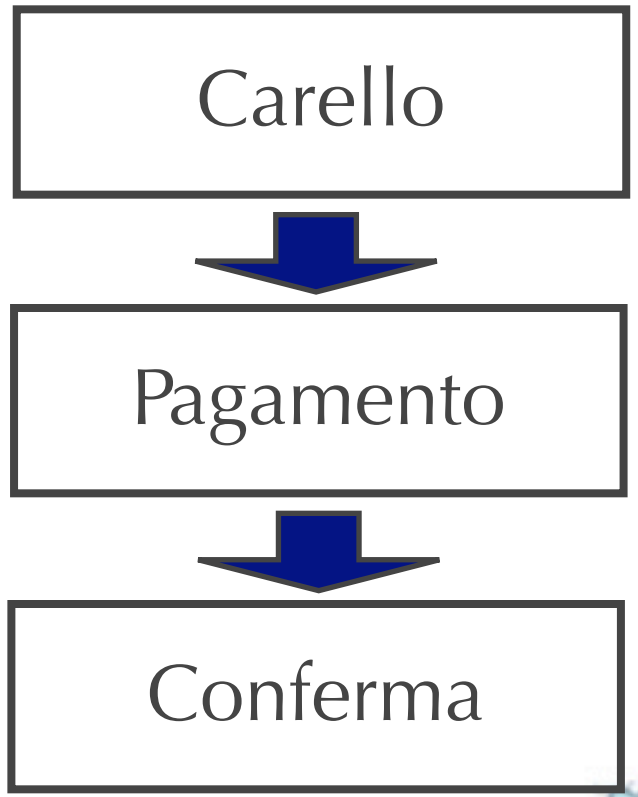
É colpa del HTTP

- HTTP, essendo un protocollo asincrono e stateless, vede ogni richiesta e risposta come un cosa a se.
- Noi vediamo ogni richiesta e risposta come un parte di una sequenza.

E Cosa Vogliamo Esprimere?

Un'applicazione non é un insieme di pagine distinte bensì un insieme di sequenze di pagine.

Per esempio: Prima c'é la pagina di descrizione del carrello, dopo, se l'utente vuole, si procede ai dati di pagamento, infine si conferma l'ordine.



E Con Questo?

- Quando noi ragioniamo sull'applicazione parliamo di pagine e di quello che deve succedere dopo.
- Ogni pagina dell'applicazione rappresenta un punto all'interno di una sequenza ed ha un futuro ed un passato.

Le Continuazioni

- Le continuazioni sono uno strumento per manipolare il “dopo.”
- Ci permettono di esprimere, direttamente, cosa succederà dopo che l’utente ha visto una pagina.

Le Continuazioni - Parte II

- Sono delle funzioni.
- Sono creati con operatori “magici”
- Possono essere chiamati più volte.
- Contengono, oltre al codice da eseguire, anche lo stato del mondo al momento della creazione.

Le Continuazioni - Parte III

- Non vi serve sapere tutto questo per usarle.

Esempio

```
(defaction aquista (carello)
  (show “carello”)
  (show “pagamento”)
  (show “conferma”))
```

UCW

the UnCommon Web application framework

Le Continuazioni in UCW

- Simulate con una trasformazione del codice sorgente originale.
- Non é una soluzione perfetta (non gestisce tutto il linguaggio sorgente) ma é più che sufficente per le nostre esigenze.

Components

- Lo stato, il comportamento e la grafica di ogni elemento della GUI (finistra, menu, form, barra di navigazione, …) e' rappresentato da un component.
- Ogni component, come qualunque funzione, viene chiamato e restituisce un valore.

Actions

- Ogni azione dell'utente (link, form, ...) chiama un action.
- Un action può passare il controllo ad un'altro component (call), o può restituire il controllo (answer).
- Quando un component passa il controllo ad un'altro viene sostituito dall'altro.

Backtracking

Gestire l'utente che torna indietro e percorre una strada diversa.

- Al momento della generazione di ogni risposta viene salvato lo stato dell'applicazione.
- Prima di gestire una richiesta lo stato dell'applicazione viene ripristinato a quello che era.

Rendering

- Ogni component deve, se é visibile sullo schermo, sapersi “trasformare” in HTML.
- UCW dispone di due strumenti per la generazione di HTML: yaclml e tal.

YACLML

Insieme di macro lisp che permettono di scrivere HTML all'interno del codice:

```
(<:table :width "100%"
  (dolist (v cose)
    (<:tr
      (<:td ...)))
```

TAL

Codice all'interno di un file HTML:

```
<ul>
  <li tal:dolist="$variabili">
    <b tal:content="valore">...</b>
  </li>
</ul>
```


Esempio

Il RERL Protocol

- La gestione di ogni richiesta é specificata in termini di funzioni generiche e classi.
- UCW consiste in una implementazione di questo protocollo.

component

- continuation - cosa fare quando questo component ha finito.
- calling-component - chi ha generato questo component.
- render-on - metodo per generare il HTML per questo component.

request-context

tutto quello che ha a che fare con una coppia richiesta/ risposta http

- application
- session
- request
- response
- current-frame
- window-component

service

il metodo che fa tutto

- Gestisce un oggetto (che può essere un application, un session o un session-frame) all'interno di un certo request-context.

session-frame

una singola interazione col server

- metodo call-callback - chiamare l'handler associato ad un http request parameter
- window-component - il component "padre." questo component dovrà creare l'html per tutta la finestra del browser (ma non lo farà da solo).

session

un insieme di interazione da parte dello stesso utente con lo stesso application

- metodo get-value - restituisce il valore associato ad una certa chiave

application

un insieme di component ed action

- url-prefix - a che url deve rispondere.
- metodo make-request-context - crea un nuovo request-context.
- metodo find-session - dato un request-context restituisce l'oggetto session associato.

server

Un'istanza di UCW

- applications - le applicazioni gestite.
- backend - chi gestisce l'HTTP.
- metodo handle-request - gestire una richiesta HTTP e mandare la risposta HTTP al client.