UnCommon Web

or: How I Learned to Stop Worrying and Love the Web

Linux Day 2004

UCW

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

The Problem

- The tools don't allow us to, directly, say what we think.

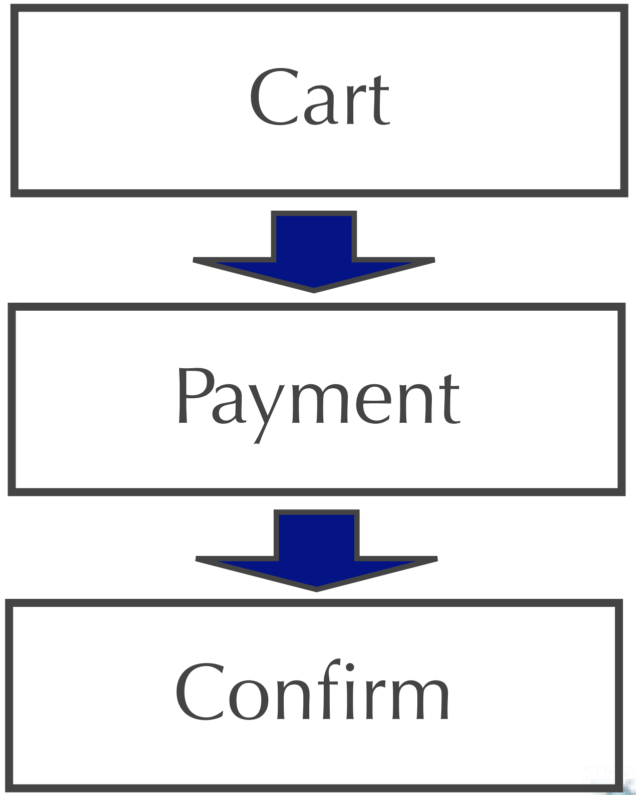
It's HTTP's fault

- HTTP, which is an asynchronous and stateless protocol, considers every request/response as an independant object.
- Developers consider every request and response as a part of a sequence of interactions.

What do we want to say?

An application is not a set of distinct pages as much as a set of sequences of pages.

Example: First we show the shopping cart page, then, if the user wants, they can continue to the payment info page, finally they can view the confirm order page.



So What?

- When developers think about an application they think in terms of what happens before and after a particular page.
- Every page in the application represents a point within a well defined sequence and has a past and a future.

Continuations

- Continuations are a tool to work with the "future."
- They allow us to express, directly, what will happpen after a user has seen a page.

Continuations - Part II

- They're functions
- They're created with "magic" operators.
- They can be called more than once.
- They contain, other than the code, the state of the world at the time they were created.

Continuations - Part III

- You don’t have to understand them to use them.

Example

```
(defaction purchase (cart)
  (show “order-contents”)
  (show “payment”)
  (show “confirm”))
```

UCW

the UnCommon Web application framework

Continuations in UCW

- Simulated by transforming the original source code.
- Not a perfect solution (doesn't handle every construct in the original language) but it's more than enough for our needs.

Components

- The state, the behaviour and the graphics of every GUI element (window, menu, form, navigation bar, …) are represented by a component.
- Every component, just like a normal function, is called and returns a value.

Actions

- Every user action (following a link, submitting a form) calls an action.
- An action can pass control to another component (call), or return control (answer).
- When a component passes control to another component it is replaced by the other camponent.

Backtracking

Dealing with users who go back and take a different road.

- When a page is generated the state of the application is saved.
- Before handling an action the state of the application is restored to what is was.

Rendering

- Every component must, if it's visible on the screen, be able to transform itself into HTML.
- UCW provides two tools for generating HTML: yaclml e tal.

YACLML

Library of lisp macros which allow html to be embedded into the code:

```
(<:table :width "100%"
  (dolist (element list-of-things)
    (<:tr
      (<:td ...)))
```

TAL

Templating library which puts code inside the HTML:

```
<ul>
  <li tal:dolist=″$list”>
    <b tal:content=″$thing”>…</b>
  </li>
</ul>
```

Example

examples/counter.lisp

The RERL Protocol

- The handling of every request/response pair is specified in terms of classes and generic functions.
- UCW is simply one possible implementation of this protocol.

class component

- continuation - what to do when this component finishes.
- calling-component - who created this component.
- render-on - method which generates the HTML for this component.

class request-context

all the information regarding an http request/response pair

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

generic function service

the methods which do all the work

- Handles an object (an application, a session or a session-frame) within a request-context.

class session-frame

a single interaction with the server

- method call-callback - call the handler associated with an http request parameter.
- window-component - the “root” component. this component must create the HTML for the entire browser window (but will rarely do this without the help of other components)

class session

a set of interactions by the same user with the same application

- method get-value - returns the value in the session associated with a particular key.

class application

set of entry-points and sessions.

- url-prefix - the url space this application controls.
- method make-request-context - create a new request-context object.
- method find-session - given a request-context returns (or creates) the session object.

class server

A UCW instance

- applications - the set of applications living in the server.
- backend - The object which deals with HTTP.
- method handle-request - deal with a pair of http request and response objects.