

Borges

A Web Application Framework

Eric Hodel <drbrain@segment7.net>

<http://segment7.net>



Seaside

- Web application framework written in Smalltalk
- Linear flow control
- Components call and return from each other
- Uses continuations to supports backtracking
- Reusable, embeddable components



Borges

- Why port to Ruby?
- Exposure to more users
- Easy migration path
- Larger language draw



Porting

- Wrote a source-munging porting tool
- Generated mostly-ruby code
- Block-based code easy to understand and port
- Smalltalk is a completely dynamic environment



Porting

```
!WASStoreBrowseFrame methodsFor: 'as yet unclassified'  
    stamp: 'ab 5/30/2003 00:20'!  
actions  
    | actions |  
    actions _ #(home browse).  
    ^ self session hasCart  
        ifTrue: [actions, #(checkout)]  
        ifFalse: [actions]  
!!  
  
def actions()  
    | actions |  
    actions = #(home browse)  
    return self.session hasCart  
        ifTrue: do actions, #(checkout) end  
        ifFalse: do actions end  
end
```



Porting

- Turn class methods into constructors
- No default or variable arguments
- Dictionary becomes Hash
- asLowercase becomes downcase



Borges URL

Represents a point in application execution

`/borges/test/@QdYCrDXpaYAfVXWX/TWsWCSvB`

`/borges`

the mount point

`/test`

the application name

`/@QdYCrDXpaYAfVXWX`

the session ID

`/TWsWCSvB`

the callback ID



Request Handling

1. Dispatcher
2. Application
3. Session or Handler



Dispatcher

- Applications get registered with the Dispatcher
- Determines which Application to call
- Applications can be unregistered



Application

- Manages Sessions and Handlers for an Application
- Creates new Sessions
- Cleans up expired Sessions



Session

- Manages user's state
- Thread spawned per Session instance
- Can filter requests for transactions, authentication



Request Processing

- Action from action key is retrieved
- Block called if simple anchor
- Form field blocks called in order if form
- Continuation saved for backtracking
- Page rendered and returned to user



Session#respond

- Magic
- Used by Components
- Marks where the user is in the application
- Allows backtracking



Session#respond

```
def respond(&block)
  request = callcc do |cc|
    return_response(block.call(
      action_url_for_continuation(cc)))
  end

  @filters.contents.each do |ea|
    ea.handle_request_in(request, self)
  end

  return request
end
```



Filter

- Allows requests to be examined/redirected/checked
- Used for transactions, authentication



Filter

```
class Once < Filter
  def handle_request_in(req, session)
    if @seen_keys.include? req.action_key then
      session.page_expired
    else
      @seen_keys << req.action_key
    end
  end
end
```



Controller

- Building block of applications
- Subclassed by Component and Controller
- Reusable
- Renderable
- Embeddable



#call & #answer

- Controller#call
 - Passes control to a Component
- Controller#answer
 - Returns control from a Component



Component

- Like a widget
- Builds the HTML document



BatchedList

- Component that breaks up a long list of items into pages
- Renders navigation links
- User is responsible for rendering items on the page



BatchedList

```
def initialize(items)
  @batcher = WABatchedList.new(items)
end

def renderContentOn(r)
  r.list_do(@batcher.batch) do |item|
    proc do
      r.anchor(item.title) do choose(item) end
    end
  end

  r.render(@batcher)
end
```



Task

- Guides a user through a set of actions
- Embedded in a Component
 - Typically a TaskFrame



Renderer

- Uses methods and blocks
- Supports CSS



Renderer

```
r.form do
  r.text "Number of items:"
  r.text_input(1) do |i| add_to_cart(5) end
  r.submit_button('Add to cart') do end
end
```

Becomes:

```
<form
  action="/borges/store/@savBDKSftCvkJISh/XPTXARbT"
  method="POST">
  <input name="5"
    type="submit" value="Add to Cart" />
  <input name="6"
    type="submit" value="Done" />
</form>
```



Renderer

```
def style_link(url)
  r.attributes['type'] = 'text/css'
  r.attributes['rel'] = 'stylesheet'
  r.attributes['href'] = url
  head_tag('link')
end
```

Becomes:

```
<link href="/borges/store/@PeoEmvVCHEIWLrdo"
      rel="stylesheet" type="text/css">
```



Documentation

- None, really ☹️
- Smalltalk source is easy to read & browse
- Avi said he would send *pizza* to documentation writers 😊



Drawbacks

- Lack of documentation
- Not designed for simple dynamic content or simple apps
- Overhead



Benefits

- Great for web applications
- Easy upgrade path to Smalltalk with Squeak or VisualWorks
- Useable on many web servers



Questions?

<http://segment7.net/ruby-code/borges/>

