

Using IRB to Drive Internet Explorer

```
D:\rubyclass>irb
irb(main):001:0> load 'iecrb'
=> true
irb(main):002:0> start_ie("http://localhost:8080")
=> #<CLabs::IEC::ClIEController:0x2b0cf70 @ie=#<WIN32OLE:0x2b0cf10>>
irb(main):003:0> show_forms
action: login
=> nil
irb(main):004:0> login_form = get_forms[0]
=> #<CLabs::IEC::IEDomFormWrapper:0x2b03d78
@form=#<WIN32OLE:0x2b03d90>>
irb(main):005:0> show_elements(login_form)
name: name value:
=> nil
irb(main):006:0> login_form.name = "brian"
=> nil
irb(main):007:0> login_form.submit
=> nil
irb(main):008:0> show_forms
action: job
=> nil
irb(main):009:0> job_form = get_forms[0]
=> #<CLabs::IEC::IEDomFormWrapper:0x2af2c78
@form=#<WIN32OLE:0x2af2c90>>
irb(main):010:0> show_elements(job_form)
name: session value: 24199864
name: name value:
name: background value: true
=> nil
irb(main):011:0> job_form.name = "misc"
=> nil
irb(main):012:0> job_form.submit
=> nil
```

Start irb.
Load the file "iecrb".

Start Internet Explorer.
Internet Explorer appears.
See what forms are on the page.
There is one form: login.

Store the form in a variable.

What is the name of the text field?
Its name is "name."

Enter a new user's name in it.

Submit the form: like hitting return.
A new page asks for a background job.
Show the forms.
One form, with an action of "job."

Assign to a variable.

Look at the controls.
What's this? It's a hidden control.
The text field.
The check box.

Set the name of the background job.

Submit the form.
The main timeclock page appears.

(continued)

```
irb(main):013:0> show_forms
action: start_day
action: start
action: job
action: start
=> nil
irb(main):014:0> start_form = get_form_by_action("start")
=> #<CLabs::IEC::IEDomFormWrapper:0x2adae40
@form=#<WIN32OLE:0x2adaea0>>
irb(main):015:0> show_elements(start_form)
name: session value: 24199864
name: name value: misc
=> nil
irb(main):016:0> get_element_by_value(start_form,"misc").click
=> nil
irb(main):017:0> show_forms
action: pause_or_stop_day
action: start
action: job
action: refresh
action: pause_or_stop_job
action: start
=> nil
irb(main):018:0> refresh_form = get_form_by_action("refresh")
=> #<CLabs::IEC::IEDomFormWrapper:0x2acd448
@form=#<WIN32OLE:0x2acd4a8>>
irb(main):019:0> show_elements(refresh_form)
name: session value: 24199864
name: refresh value: Refresh
=> nil
irb(main):020:0> refresh_form.elements("refresh").click
=> nil
```

Show the forms.

Two forms have the same action. Hmm.

Use the first "start" form.

What is the name of the "misc" button?

Name is "name"; value is "misc".
We'll use the value.
Click on the button.

Some new forms have appeared.

The "refresh" form.

What is the button name?

This time, we'll use the name.

Click it.

```
# iec.rb - General utility functions useful for Timeclock
require 'cl/iec'

def start_ie(url)
  "Start the IE Controller at the specified URL."
  @iec = ClIEController.new(true)
  @iec.navigate(url)
  return @iec
end

def get_forms()
  "Return the Forms on the current page of IE as an array."
  page_forms = []
  for form in @iec.document.forms
    page_forms << IEDomFormWrapper.new(form)
  end
  return page_forms
end

def show_forms()
  "Print the actions for each of the forms on the current page."
  page_forms = []
  for form in @iec.document.forms
    puts "action: " + form.action
  end
end

def get_form_by_action(action)
  "Return the first Form on the current page that has the specified Action."
  for form in @iec.document.forms
    if form.action == action
      return IEDomFormWrapper.new(form)
    end
  end
end

def show_elements(form)
  "Print the Name and Value of the Elements in the Form."
  for element in form.elements
    puts "name: " + element.name + " value: " + element.value
  end
end

def get_element_by_value(form, value)
  "Return the first Element of the Form with the specified Value"
  for element in form.elements
    if element.value == value
      return element
    end
  end
end
```

```
def get_html
  "Return the full html of the current page."
  @iec.document.getElementsByTagName("HTML").item(0).outerHtml
end

# This allows us to access the Form OLE object wrapped by the class.
class IEDomFormWrapper
  def form
    return @form
  end
end

def show_ole_methods(ole_object)
  "Print the ole/com methods for the specified object."
  for method in ole_object.ole_methods
    puts method.name
  end
end
```

Internet Explorer Cheat Sheet

Getting Started

| | |
|---|---------------------------------|
| <code>require 'cl/iec'</code> | Load the IEController library |
| <code>iec = ClIEController.new(true)</code> | Start IE |
| <code>iec.navigate("url")</code> | Go to specified URL. |
| <code>iec.wait</code> | Wait until IE is no longer busy |

Forms

| | |
|--|--|
| <code>form = IEDomFormWrapper.new(iec.form)</code> | Access the form on the current web page. |
| <code>form.name</code> | Return the name of the form |
| <code>form.action</code> | Return the action of the form |
| <code>iec.document.forms.each { x puts x.name}</code> | Display the names of all the forms on the page |
| <code>iec.document.forms(x)</code> | Access a specific form on a page, where x is a number (zero-index) |
| <code>iec.document.forms('name')</code> | Access a specific form by name |

Controls

| | |
|--|--|
| <code>form.elements.each { x puts x.name}</code> | Print the names of the controls in a form (in order) |
| <code>form.name = value</code> | Set the text field named "name" to the specified value. |
| <code>form.name = true</code> | Check the check box named "name". (Uncheck: <i>false</i>) |
| <code>form.elements('name').click</code> | Click the button with the specified name. |
| <code>form.name</code> or <code>form.elements('name')</code> | Return the current value of the named control. |
| <code>form.elements.each { x return x if x.value == 'value'}</code> | Return the control with the specified value |

Internet Explorer Cheat Sheet

Page 2

Accessing the Document Object Model

| | |
|--|--|
| <pre>dv = ClIEDomViewer.new(ie) root = dv.htmlRootNode dv.buildNodeWrapperTree(root)</pre> | Create a DOM viewer for the current page. |
| <pre>dv.outputDom</pre> | Display the nodes and values of the current document |
| <pre>message = dv.getNodeWrapperFromPath('nodename') message.node.nodeValue</pre> | Return the value of a specific node |

Accessing the HTML

| |
|---|
| <pre>ie.document.getElementsByTagName("HTML").item(0).outerHtml</pre> |
| Extract the HTML for the current page |

Methods for Objects

| | |
|--|---|
| <pre>object.methods</pre> | Return the Ruby methods for the given object. |
| <pre>object.ole_methods</pre> | Return the OLE methods for a given WIN32OLE object. |
| For full documentation of the OLE methods for Internet Explorer objects see http://msdn.microsoft.com/workshop/browser/webbrowser/reference/Objects/InternetExplorer.asp | |

IeController

[HomePage](#) | [RecentChanges](#) | [Preferences](#) | [RubyGarden](#)

[IeController](#) is a package by [Wiki:ChrisMorris](#) that drives [Wiki:InternetExplorer](#) and commits testing against it. It compares favorably to the Perl library <http://samie.sf.net>

The latest version is at the bottom of this:

<http://clabs.org/ruby.htm>

Make sure you also install two other packages found there: clutil and clxmlserial. (6/23/03 - Latest cvs no longer requires these -- this was a weird dependency that snuck in.)

It's rough and in progress, but if you're interested, let me know. Go to my name page [[ChrisMorris](#)] for contact info. (You can browse the cvs of it here: <http://clabs.org/iec>)

This correspondence inspired a hello-world example for it:

> *Where's a "hello world" example?*

> <http://samie.sf.net> , *your competition in Perl, does this:*

```
my $URL = "http://www.amazon.com";
Navigate($URL);
WaitForBusy();
$IEDocument = GetDocument();
$seconds = WaitForDocumentComplete();
print "Amazon took $seconds seconds to load\n";
SetListBoxItem("url", "index=baby");
SetEditBox("field-keywords", "rattlesnake");
ClickFormImage("Go");
$seconds = WaitForDocumentComplete();
print "Diaper page took $seconds seconds to load\n";
```

My version would be as follows:

```
require 'cl/iec'

VISIBLE = true
iec = ClIEController.new(VISIBLE)
iec.navigate('http://www.amazon.com')
form = IEDomFormWrapper.new(iec.form)
form.url = 'Baby'
form.invoke('field-keywords').value = 'rattlesnake'
form.submit
iec.wait
```

Comments:

- the navigate method automatically calls the wait method, form.submit does not (but I see now it should).
- IEDomFormWrapper? basically has a method_missing routine which handles a bunch of dirty work. In prepping this, I found I usually have add a 'def form...' method to my test scripts which will automatically take care of creating the wrapper -- that helper function needs to be moved into the iec lib.
- The built-in drop down wrapper allows you to set the drop down based on the display value, not the internal value ('index=baby').
- Because Amazon put a hyphen in their edit box input name, the slick method_missing approach in the form wrapper is thwarted. Normally, you can do:

```
iec.form.field-keywords = 'rattlesnake'
```

- ... but the hyphen is parsed as a method call so only the term 'field' is caught by method missing -- so the lib tries to find a form field called 'field' and it fails to do so. invoke is a WIN32OLE method that allows you to force a call through to the WIN32OLE instance. Because we're also not getting to use the built in text box wrapper, we have to also reference the .value property.
- The iec.form call takes an index argument that defaults to 0, but you can set it to reference other forms on a multiple form page.

Here's another example, using Google and helper routines that probably need to find their way into the lib:


```
require 'cl/iecontroller'

def form
  IEDomFormWrapper.new(@iecontroller.form)
end

def submit
  form.submit
  @iecontroller.wait
end

VISIBLE = true
@iecontroller = ClIEController.new(VISIBLE)
@iecontroller.navigate 'http://www.google.com'
form.q = 'Programming Ruby'
submit
```

Neither of these websites seems to have easily testable output. But, ugly as it is, here's a way to get the first returned link and click it using the `ClIEDomViewer`. First time requires a dump to find the path to the first link:

```
dv = ClIEDomViewer.new(@iecontroller)
File.open('dom.dump.txt', 'w') do |f| dv.outputDom(f) end
```

Digging through `dom.dump.txt`, we find the first link here:

```
[snip]
nodeName: -HTML-BODY1-DIV1
nodeName: -HTML-BODY1-DIV1-P1
nodeName: -HTML-BODY1-DIV1-P1-A1
nodeName: -HTML-BODY1-DIV1-P1-A1-B1
nodeName: -HTML-BODY1-DIV1-P1-A1-B1-#text1
nodeValue: Programming
nodeName: -HTML-BODY1-DIV1-P1-A1-#text1
nodeValue:
nodeName: -HTML-BODY1-DIV1-P1-A1-B2
nodeName: -HTML-BODY1-DIV1-P1-A1-B2-#text1
nodeValue: Ruby
nodeName: -HTML-BODY1-DIV1-P1-A1-#text2
nodeValue: : The Pragmatic Programmer's Guide
[snip]
```

so, we now have a path to the first link: "-HTML-BODY1-DIV1-P1-A1" and we can do:

```
root = dv.htmlRootNode
dv.buildNodeWrapperTree(root)
link = dv.getNodeWrapperFromPath('HTML-BODY1-DIV1-P1-A1', root)
link.node.click
@iec.wait
```

Obviously, this is fragile should Google ever change their layout, but without consistent ids on the <a> tags, or other somesuch, I don't know of another way to do it.

Chris

<http://clabs.org>

[HomePage](#) | [RecentChanges](#) | [Preferences](#) | [RubyGarden](#)

[Edit text of this page](#) | [View other revisions](#)

Last edited June 23, 2003 10:04 am ([diff](#))

Search: