

Appendix A

Sarah Vessels

March 28, 2011

Contents

1	Qian, Miao, Zeng Model Tool	2
1.1	link.rb	2
1.2	page.rb	3
1.3	pfd.rb	5
1.4	scraper.rb	7
1.5	site.rb	9
2	Atomic Section Model Tool	13
2.1	atomic_section.rb	13
2.2	component_interaction_model.rb	15
2.3	erb_document.rb	16
2.4	erb_document_test.rb	28
2.5	erb_grammar.treetop	33
2.6	erb_node_extensions.rb	35
2.7	erb_output_tag.rb	36
2.8	erb_tag.rb	37
2.9	erb_tag_test.rb	39
2.10	erb_yield.rb	39
2.11	fake_erb_output.rb	40
2.12	form_transition.rb	41
2.13	generator.rb	41
2.14	html_close_tag.rb	42
2.15	html_directive.rb	43
2.16	html_doctype.rb	44
2.17	html_open_tag.rb	44
2.18	html_quoted_value.rb	45
2.19	html_self_closing_tag.rb	47
2.20	html_tag_attribute.rb	47
2.21	html_tag_attributes.rb	48
2.22	link_transition.rb	49
2.23	parser.rb	49
2.24	rails_url.rb	51
2.25	range.rb	52
2.26	redirect_transition.rb	52
2.27	ruby_code.rb	52
2.28	runner.rb	53
2.29	shared_atomic_section_methods.rb	53
2.30	shared_children_methods.rb	58

2.31	shared_erb_methods.rb	59
2.32	shared_html_tag_methods.rb	59
2.33	shared_methods.rb	59
2.34	shared_open_tag_methods.rb	60
2.35	shared_sexp_methods.rb	60
2.36	shared_sexp_parsing.rb	70
2.37	shared_transition_methods.rb	70
2.38	single_file_generator.rb	71
2.39	syntax_node.rb	72
2.40	syntax_node_test.rb	74
2.41	test_helper.rb	75
2.42	text.rb	76
2.43	transition.rb	76
2.44	whitespace.rb	77
3	Shared Files	78
3.1	html_parsing.rb	78
3.2	link_text.rb	81
3.3	uri_extensions.rb	81

1 Qian, Miao, Zeng Model Tool

1.1 link.rb

```

01: require 'uri'
02:
03: class Link
04:   attr_reader :uri1, :uri2, :description
05:   attr_accessor :target_page
06:
07:   def initialize(uri1, uri2, target_page, desc)
08:     unless uri1.respond_to?(:get_uniq_parts) && uri2.respond_to?(:get_uniq_parts)
09:       raise ArgumentError, "Given URIs must respond to .get_uniq_parts() method"
10:     end
11:     unless target_page.respond_to? :uri
12:       raise ArgumentError, "Given target_page must have .uri property"
13:     end
14:     unless uri2.get_uniq_parts() == target_page.uri.get_uniq_parts()
15:       raise ArgumentError,
16:         "Given target_page does not have same URI as given uri2"
17:     end
18:     if desc.nil? || !desc.is_a?(String)
19:       raise ArgumentError, "Expected String description of link, got #{desc.class.name}"
20:     end
21:     @uri1 = uri1
22:     @uri2 = uri2
23:     @target_page = target_page
24:     @description = desc
25:   end
26:

```

```

27: def ==(other)
28:   other.is_a?(Link) && @uri1 == other.uri1 && @uri2 == other.uri2 &&
29:     @target_page == other.target_page
30: end
31:
32: def eql?(other)
33:   self == other
34: end
35:
36: def hash
37:   @uri1.hash ^ @uri2.hash ^ @target_page.hash
38: end
39:
40: def to_s
41:   sprintf("%s => %s via %s", @uri1.request_uri, @uri2.request_uri, @description)
42: end
43: end
44:

```

1.2 page.rb

```

01: require 'rubygems'
02: require 'open-uri'
03: require File.join(File.join(File.expand_path(File.dirname(__FILE__)), '..'), 'html_parsing.rb')
04: require File.join(File.join(File.expand_path(File.dirname(__FILE__)), '..'), 'uri_extensions.rb')
05:
06: class Page
07:   include SharedHtmlParsing
08:   extend SharedHtmlParsing::ClassMethods
09:   attr_reader :uri, :links, :uri_parts
10:   attr_accessor :is_copy, :link_texts
11:
12:   def initialize(raw_uri, html=nil)
13:     if raw_uri.is_a? String
14:       @uri = Page.parse_uri_forgivingly(raw_uri)
15:       if @uri.nil?
16:         raise ArgumentError, "Could not parse given String URI #{raw_uri}"
17:       end
18:     elsif raw_uri.is_a? URI
19:       @uri = raw_uri
20:     else
21:       raise ArgumentError, "Only URI and String instances are allowed for given URI (got
#{raw_uri.class.name})"
22:     end
23:     if html.nil? || !html.is_a?(Nokogiri::HTML::Document)
24:       html = Page.open_uri(raw_uri)
25:       if html.nil?
26:         raise ArgumentError, "Could not open URI for page #{@uri}"
27:       end
28:     end

```

```

29:   @uri_parts = @uri.get_uniq_parts()
30:   @link_texts = (Page.get_link_uris_with_text(@uri, html) +
31:     Page.get_form_uris_with_text(@uri, html)).uniq
32:   @links = []
33:   @is_copy = false
34: end
35:
36: def ==(other)
37:   other.is_a?(Page) && @uri == other.uri
38: end
39:
40: def <=>(other)
41:   @uri <=> other.uri
42: end
43:
44: def eql?(other)
45:   self == other
46: end
47:
48: def hash
49:   @uri.hash
50: end
51:
52: def Page.open_uri(uri)
53:   return nil if uri.nil?
54:   begin
55:     stringio = open(uri.to_s)
56:   rescue => err
57:     printf("Got error '%s' trying to open URI %s, skipping...\n",
58:       err.to_s, uri.to_s)
59:     stringio = nil
60:   end
61:   stringio.nil? || stringio.content_type != 'text/html' ? nil : Nokogiri::HTML(stringio)
62: end
63:
64: def to_s
65:   str = sprintf("Page %s (%d links", @uri.request_uri, @link_texts.length)
66:   unless @links.empty?
67:     str << ' : '
68:     str << @links.map(&:to_s).join(', ')
69:   end
70:   str << ' )'
71:   str
72: end
73: end
74:

```

1.3 pfd.rb

```
001: require 'erb'
002: require 'page.rb'
003: require 'link.rb'
004:
005: class PFD
006:   PFDTemplateFile = 'ptt_template.html.erb'.freeze
007:   attr_reader :pages, :links, :root_uri
008:
009:   def initialize(pages, links, root_uri)
010:     unless pages.respond_to? :each
011:       raise ArgumentError, "Given pages arg must be enumerable"
012:     end
013:     unless links.respond_to? :each
014:       raise ArgumentError, "Given links arg must be enumerable"
015:     end
016:     unless root_uri.is_a? URI
017:       raise ArgumentError, "Given root_uri must be of type URI"
018:     end
019:     @root_uri = root_uri
020:     @pages = pages
021:     @links = links
022:   end
023:
024:   def ==(other)
025:     return false unless other.is_a?(PFD)
026:     if @pages.length != other.pages.length ||
027:        @links.length != other.links.length
028:       return false
029:     end
030:     @pages.each do |page|
031:       return false unless other.pages.include?(page)
032:     end
033:     other.pages.each do |page|
034:       return false unless @pages.include?(page)
035:     end
036:     @links.each do |link|
037:       return false unless other.links.include?(link)
038:     end
039:     other.links.each do |link|
040:       return false unless @links.include?(link)
041:     end
042:     return true
043:   end
044:
045:   def eql?(other)
046:     self == other
047:   end
048:
```

```

049: def get_test_paths
050:   preorder(@pages[0], 0, [], "Start page")
051: end
052:
053: def hash
054:   hash_code = 1
055:   @pages.each do |page|
056:     hash_code = hash_code ^ page.hash
057:   end
058:   @links.each do |link|
059:     hash_code = hash_code ^ link.hash
060:   end
061:   hash_code
062: end
063:
064: def PFD.to_html(site_uri, test_cases)
065:   pfd_erb = ERB.new(IO.readlines(PFDTemplateFile).join, 0, "%<>")
066:   pfd_erb.result(binding)
067: end
068:
069: def to_s
070:   pages_str = @pages.map(&:to_s).join("\n\t")
071:   links_str = @links.map(&:to_s).join("\n\t")
072:   sprintf("Pages (%d):\n\t%s\nLinks (%d):\n\t%s",
073:     @pages.length,
074:     pages_str,
075:     @links.length,
076:     links_str)
077: end
078:
079: private
080:   def preorder(page, level, test_paths, desc_how_got_to_page)
081:     if test_paths.last.length <= level
082:       # Based on level, still appending to the last test case
083:       #test_paths.last << page.uri
084:       test_paths.last << LinkText.new(page.uri, desc_how_got_to_page)
085:     else
086:       # Have finished concatenating test case, so start a new one based
087:       # on the test case we just completed
088:       new_path = test_paths.last.dup[0...level]
089:       #new_path << page.uri
090:       new_path << LinkText.new(page.uri, desc_how_got_to_page)
091:       test_paths << new_path
092:     end
093:     unless page.is_copy
094:       page.links.each do |link|
095:         preorder(link.target_page, level + 1, test_paths, link.description)
096:       end
097:     end
098:     test_paths

```

```
099:     end
100: end
101:
```

1.4 scraper.rb

```
001: #!/usr/bin/env ruby
002: require 'uri'
003: require 'optparse'
004: require 'pfd.rb'
005: require 'page.rb'
006: require 'site.rb'
007: require 'yaml'
008: require 'fileutils'
009:
010: options = {}
011: optparse = OptionParser.new do |opts|
012:   opts.banner = sprintf("Usage: %s [options]", $0)
013:
014:   opts[:uri] = nil
015:   opts.on('-u', '--uri URI', 'URI of site home page') do |uri|
016:     options[:uri] = uri
017:   end
018:
019:   opts[:input_file] = nil
020:   opts.on('-i', '--input FILE', 'YAML input file with site structure') do |file|
021:     options[:input_file] = file
022:   end
023:
024:   opts[:output_file] = nil
025:   opts.on('-o', '--output FILE',
026:     'YAML site structure will be written here') do |file|
027:     options[:output_file] = file
028:   end
029:
030:   opts[:ptt_file] = nil
031:   opts.on('-p', '--ptt FILE', 'File where PTT will be saved') do |file|
032:     options[:ptt_file] = file
033:   end
034:
035:   opts[:test_paths_file] = nil
036:   opts.on('-t', '--tests FILE',
037:     'File in which generated test paths will be stored') do |file|
038:     options[:test_paths_file] = file
039:   end
040:
041:   opts.on('-h', '--help', 'Display this screen') do
042:     puts opts
043:     exit
044:   end
```

```

045: end
046:
047: # Parse command-line parameters and remove all flag parameters from ARGV
048: optparse.parse!
049:
050: should_generate_ptt = true
051: if options[:uri] && options[:input_file]
052:   print "ERR: define only one of --uri, --input\n"
053:   puts optparse
054:   exit
055: elsif options[:uri]
056:   puts "Got URI #{options[:uri]}"
057:   site = Site.new(Page.new(options[:uri]))
058: elsif options[:input_file]
059:   if File.exists? options[:input_file]
060:     yaml = IO.readlines(options[:input_file]).join
061:     user_input = YAML::load(yaml)
062:     if user_input.is_a? Site
063:       site = user_input
064:       printf("Read site from file %s\n", options[:input_file])
065:     elsif user_input.is_a? PFD
066:       ptt = user_input
067:       site = Site.from_pfd(ptt)
068:       should_generate_ptt = false
069:       printf("Read PTT from file %s\n", options[:input_file])
070:     else
071:       printf("ERR: could not get a site or a PTT from the given input " +
072:         "file: %s\n", options[:input_file])
073:       exit
074:     end
075:   else
076:     printf("ERR: given input file %s does not exist\n", options[:input_file])
077:     exit
078:   end
079: else
080:   # Missing necessary params, print help and exit
081:   puts optparse
082:   exit
083: end
084:
085: printf("\n%s\n", site.to_s)
086: if options[:output_file]
087:   printf("\nWriting site to %s...\n", options[:output_file])
088:   File.open(options[:output_file], 'w') do |file|
089:     file.puts YAML::dump(site)
090:   end
091:   puts "File successfully written"
092: end
093: print "\n"
094:

```



```

095: if should_generate_ptt
096:   pfd = site.get_pfd()
097:   ptt = Site.pfd2ptt(pfd)
098: end
099:
100: if options[:ptt_file] && !ptt.nil?
101:   printf("\nWriting PTT to %s...\n", options[:ptt_file])
102:   File.open(options[:ptt_file], 'w') do |file|
103:     file.puts YAML::dump(ptt)
104:   end
105:   print "File successfully written\n\n"
106: end
107:
108: test_paths = ptt.get_test_paths()
109:
110: if options[:test_paths_file] && !test_paths.empty?
111:   printf("Writing test paths to %s...\n", options[:test_paths_file])
112:   File.open(options[:test_paths_file], 'w') do |file|
113:     test_paths.each do |uris|
114:       file.puts uris.map(&:to_s).join(" => ")
115:     end
116:   end
117:   print "File successfully written\n\n"
118: end
119:
120: dir_name = site.home.uri_parts.join('.').gsub(/\//, '_').chomp('_').chomp('.')
121: Dir.mkdir(dir_name)
122: FileUtils.copy('screen.css', dir_name)
123: html_path = dir_name + '/index.html'
124: printf("Writing HTML file with test paths to %s...\n", html_path)
125: File.open(html_path, 'w') do |file|
126:   file.puts PFD.to_html(site.home.uri, test_paths)
127: end
128: puts "File successfully written"
129:

```

1.5 site.rb

```

001: require File.join(File.expand_path(File.dirname(__FILE__)), 'page.rb')
002: require File.join(File.expand_path(File.dirname(__FILE__)), 'link.rb')
003: require File.join(File.expand_path(File.dirname(__FILE__)), 'pfd.rb')
004: require File.join(File.join(File.expand_path(File.dirname(__FILE__)), '..'), 'uri_extensions.rb')
005: require 'pp'
006:
007: class Site
008:   attr_reader :pages, :home
009:
010:   def initialize(home_page, pages=[])
011:     unless home_page.is_a? Page
012:       raise ArgumentError, "Given home page must be a Page instance"

```

```

013:   end
014:   @home = home_page
015:   if pages.empty?
016:     printf("Getting pages for site at %s...\n", @home.uri)
017:     @pages = Site.get_pages(@home, [@home])
018:     printf("Got %d pages for site at %s\n", @pages.length, @home.uri)
019:   else
020:     @pages = pages
021:   end
022: end
023:
024: def Site.from_pfd(pfd)
025:   unless pfd.is_a? PFD
026:     raise ArgumentError, "Expected given pfd param to be of type PFD"
027:   end
028:   all_pages = pfd.pages
029:   home_page_uri_desc = pfd.root_uri.get_uniq_parts()
030:   home_page = all_pages.find do |page|
031:     page.uri_parts == home_page_uri_desc
032:   end
033:   if home_page.nil?
034:     raise "Cannot extract home page with URI " + pfd.root_uri.to_s +
035:       " from pages in PFD"
036:   end
037:   Site.new(home_page, all_pages - [home_page])
038: end
039:
040: def get_pfd
041:   printf("Getting PFD for site %s...\n", @home.uri.to_s)
042:   pages = [@home, @pages].flatten.uniq
043:   links = []
044:   pages.each do |page1|
045:     page1.link_texts.each do |link_text|
046:       page2 = pages.find do |page|
047:         page.uri_parts == link_text.uri_parts
048:       end
049:       if page2.nil?
050:         printf("ERR: cannot find page with URI %s in site\n", link_text.uri.request_uri)
051:       next
052:     end
053:     new_link = Link.new(page1.uri, link_text.uri, page2, link_text.description)
054:     page1.links << new_link
055:     links << new_link unless links.include? new_link
056:   end
057: end
058: PFD.new(pages, links, @home.uri)
059: end
060:
061: def Site.pfd2ptt(pfd)
062:   puts "Converting PFD to PTT..."

```

```

063:   ptt = pfd.dup
064:   first = []
065:   second = []
066:
067:   # Step 1
068:   first << ptt.pages[0]
069:
070:   while !first.empty?
071:     # Step 3
072:     next_page = first[0]
073:
074:     # If pid is within SECOND, then go to (5). Otherwise, add it into the end
075:     # of SECOND
076:     unless second.include? next_page
077:       second << next_page
078:
079:       # Step 4: if pid is linking to other pages:
080:       next_page.links.each do |link|
081:         linked_page = link.target_page
082:
083:         # If some of the other page identifiers are within FIRST or SECOND:
084:         if first.include?(linked_page) || second.include?(linked_page)
085:           # Then generate their copies
086:           copy = linked_page.dup
087:           copy.is_copy = true
088:
089:           # Retain the links between pid and the other pages (or their
090:           # copies) of the PFD
091:           link.target_page = copy
092:
093:           # Add the other page identifiers (or their copies) into the end
094:           # of FIRST
095:           first << copy
096:         else
097:           # Add the other page identifiers (or their copies) into the end
098:           # of FIRST
099:           first << linked_page
100:         end
101:       end
102:     end
103:
104:     # Step 5
105:     first.delete(next_page)
106:   end
107:   ptt
108: end
109:
110: def to_s
111:   sprintf("Pages in site rooted at %s:\n\t%s",
112:     @home.uri.to_s,

```

```

113:     @pages.map(&:uri).map(&:request_uri).join("\n\t")
114: end
115:
116: private
117: def Site.get_pages(root_page, pages, blacklist_uris=[])
118:   existing_uris = pages.map(&:uri_parts)
119:   new_pages = []
120:
121:   # Don't use #each_with_index because we'll also be using #delete_at, and
122:   # that does weird stuff to the iterator.
123:   (root_page.link_texts.length-1).downto(0) do |i|
124:     link_text = root_page.link_texts[i]
125:     uri = link_text.uri
126:     uri_desc = link_text.uri_parts
127:     if blacklist_uris.include?(uri_desc)
128:       # Current URI is already blacklisted, so remove it from this page
129:       # and skip ahead
130:       root_page.link_texts.delete_at(i)
131:       next
132:     elsif existing_uris.include?(uri_desc)
133:       # Current URI is already represented by a Page, so no need to create
134:       # another Page for it; skip ahead
135:       next
136:     end
137:     html = Page.open_uri(uri)
138:     if html.nil?
139:       # Keep track of URIs that give us errors (404 not found, 405 method
140:       # not allowed, etc.) or that aren't HTML pages, so we don't keep
141:       # trying to open them
142:       blacklist_uris << uri_desc
143:       root_page.link_texts.delete_at(i)
144:     else
145:       existing_uris << uri_desc
146:       new_page = Page.new(uri, html)
147:       new_pages << new_page
148:       pages << new_page
149:     end
150:   end
151:   unless new_pages.empty?
152:     num_new = new_pages.length
153:     printf("Got %d new Page%s linked from %s\n", num_new,
154:       num_new == 1 ? '' : 's', root_page.to_s)
155:     new_pages.each do |page|
156:       print '. '
157:       get_pages(page, pages, blacklist_uris)
158:     end
159:   end
160:   pages
161: end
162: end

```

163:
164:

2 Atomic Section Model Tool

2.1 atomic_section.rb

```
001: class AtomicSection
002:   include SharedMethods
003:   include ERBGrammar::SharedTransitionMethods
004:   include SharedChildrenMethods
005:   include SharedSexpParsing
006:   include SharedSexpMethods
007:   extend SharedSexpMethods::ClassMethods
008:   attr_reader :content, :count, :index
009:
010:   def initialize(count=1)
011:     @content = []
012:     @count = count
013:     @index = -1
014:   end
015:
016:   def can_add_node?(node)
017:     return false if node.nil?
018:     return false unless node.browser_output?
019:     return false if node.index.nil?
020:     return true if @content.empty?
021:     last_node = @content.last
022:     last_node.same_atomic_section?(node) && last_node != node
023:   end
024:
025:   def component_expression(seen_children=[])
026:     unless @content.nil? || @content.empty?
027:       # puts "Atomic section has content:"
028:       # pp @content
029:       # puts "-----"
030:       # Necessary to check content of atomic section in case it contains an
031:       # ERBOutputTag that has a render() call, which would be treated as
032:       # aggregation in the component expression
033:       child_str = @content.collect do |node|
034:         if node.respond_to?(:component_expression)
035:           node.component_expression()
036:         else
037:           nil
038:         end
039:       end.compact.select do |expr|
040:         !expr.blank?
041:       end.join(' . ')
042:       unless child_str.blank? || ' . ' == child_str
043:         #puts "Component expr. segment from p#{@count}: " + child_str
```

```

044:     return child_str
045:   end
046: end
047: expr = sprintf("p%d", @count)
048: #puts "Component expr. segment from p#@count: " + expr
049: expr
050: end
051:
052: def get_local_transitions(source)
053:   []
054: end
055:
056: def include?(node)
057:   return false if @content.nil? || @content.empty?
058:   return false if node.nil?
059:   if node.is_a?(ERBGrammar::FakeERBOutput)
060:     ERBGrammar::FakeERBOutput.new(@content.map(&:text_value), @index) == node
061:   else
062:     @content.include?(node)
063:   end
064: end
065:
066: def inspect
067:   to_s
068: end
069:
070: def range
071:   return nil if @content.nil? || @content.empty?
072:   @content.sort! do |a, b|
073:     a.index <=> b.index
074:   end
075:   start_index = @content.first.index
076:   end_index = @content.last.index
077:   if start_index.nil? || end_index.nil?
078:     raise RuntimeError, "Nil start or end index; atomic section has content: " + @con-
tent.inspect
079:   end
080:   (start_index..end_index)
081: end
082:
083: # TODO: remove duplication between this and SharedHTMLTagMethods
084: def ruby_code
085:   @content.collect do |child|
086:     if child.respond_to?(:ruby_code)
087:       child.ruby_code()
088:     else
089:       'puts "' + child.text_value.gsub(/"/, "\\\"") + '"'
090:     end
091:   end.join("\n")
092: end

```

```

093:
094: def save(file_path)
095:   File.open(file_path, 'w') do |file|
096:     @content.each do |node|
097:       file.puts node.text_value
098:     end
099:   end
100: end
101:
102: def to_s(indent_level=0)
103:   to_s_with_prefix(indent_level, sprintf("Atomic Section #%d (indices %s):\n%s",
104:     @count,
105:     range().to_s,
106:     @content.collect do |node|
107:       node.to_s(indent_level+1)
108:     end.join("\n")))
109: end
110:
111: def try_add_node?(node)
112:   if can_add_node?(node)
113:     @index = node.index if @content.empty?
114:     @content << node
115:     if node.respond_to?(:atomic_section_count)
116:       node.atomic_section_count = @count
117:     end
118:     true
119:   else
120:     false
121:   end
122: end
123: end
124:

```

2.2 component_interaction_model.rb

```

01: class ComponentInteractionModel
02:   attr_reader :site_root, :start_page, :component_expression, :atomic_sections, :transitions
03:
04:   def initialize(root_of_site, start_page, comp_expr, sections, trans)
05:     if root_of_site.nil?
06:       raise ArgumentError, "Cannot have a nil site root"
07:     end
08:     if start_page.nil? || start_page.blank?
09:       raise ArgumentError, "Cannot have a nil/blank start page"
10:     end
11:     if comp_expr.nil?
12:       raise ArgumentError, "Cannot have a nil component expression"
13:     end
14:     if sections.nil? || !sections.is_a?(Array) || sections.empty?
15:       raise ArgumentError, "Must give at least 1 atomic section in Array (got #{sections.class.name}

```

```

16:   end
17:   if trans.nil? || !trans.is_a?(Array)
18:     raise ArgumentError, "Must give a non-nil Array of transitions (got #{trans.class.name})"
19:   end
20:   @site_root = root_of_site
21:   @start_page = start_page
22:   @component_expression = comp_expr
23:   @atomic_sections = sections
24:   @transitions = trans
25: end
26:
27: def controller
28:   rails_uri = RailsURL.from_path(@start_page, @site_root)
29:   if rails_uri.nil?
30:     'UNKNOWN'
31:   else
32:     rails_uri.controller || 'UNKNOWN'
33:   end
34: end
35:
36: def start_url
37:   file_name = File.basename(@start_page.downcase)
38:   suffix_start = file_name.index('.')
39:   if suffix_start.nil?
40:     method = file_name
41:   else
42:     method = file_name[0...suffix_start]
43:   end
44:   sprintf("%s/%s/%s", @site_root, controller(), method)
45: end
46:
47: def to_s
48:   tab = '  '
49:   trans_str = @transitions.collect do |trans|
50:     trans.to_s(tab * 2)
51:   end.join("\n")
52:   sprintf("Component Interaction Model\n\tStart page: %s\n\tStart URL: %s\n\tComponent\n\texpression: %s\n\tTransitions:\n%s", @start_page, start_url(), @component_expression, trans_str)
53: end
54: end
55:

```

2.3 erb_document.rb

```

001: require 'rubygems'
002: require 'ruby_parser'
003: root_dir = File.join(File.expand_path(File.dirname(__FILE__)), '..')
004: require File.join(root_dir, 'atomic_section.rb')
005: require File.join(root_dir, 'rails_url.rb')
006: require File.join(root_dir, 'transition.rb')

```



```

007: require File.join(root_dir, 'form_transition.rb')
008: require File.join(root_dir, 'link_transition.rb')
009: require File.join(root_dir, 'redirect_transition.rb')
010: require File.join(root_dir, 'range.rb')
011:
012: module ERBGrammar
013:   class ERBDocument < Treetop::Runtime::SyntaxNode
014:     include Enumerable
015:     include SharedAtomicSectionMethods
016:     extend SharedAtomicSectionMethods::ClassMethods
017:     include SharedChildrenMethods
018:     include SharedTransitionMethods
019:     attr_reader :content, :initialized_content
020:     attr_accessor :source_file
021:     STATEMENT_END = /\[r\n;\]/.freeze
022:
023:     def [](obj)
024:       if obj.is_a?(Fixnum)
025:         each_with_index do |el, i|
026:           return el if el.index == obj || i == obj
027:         end
028:       elsif obj.respond_to?(:include?)
029:         i = 0
030:         select do |el|
031:           is_nil = el.index.nil?
032:           index_match = !is_nil && obj.include?(el.index)
033:           i_match = is_nil && obj.include?(i)
034:           result = index_match || i_match
035:           i += 1
036:           result
037:         end
038:       else
039:         nil
040:       end
041:     end
042:
043:     def compress_content
044:       # Need to go in reverse lest we end up end up with unnested content
045:       (length-1).downto(0) do |i|
046:         element = self[i]
047:         next unless element.respond_to?(:close) &&
048:           !element.close.nil? &&
049:           element.respond_to?(:content)
050:         # element is open tag
051:         range = element.index+1...element.close.index
052:         content = self[range].compact
053:         next if content.nil? || content.empty?
054:         element.content = content.dup
055:         content.each do |consumed_el|
056:           delete_node_check_size(consumed_el)

```

```

057:     end
058:     # Closing element is not part of the content, but it no longer
059:     # needs to appear as a separate element in the tree
060:     delete_node_check_size(element.close)
061: end
062: end
063:
064: def each
065:   if @initialized_content
066:     @content.each { |n| yield n }
067:   else
068:     yield node
069:     if !x.nil? && x.respond_to?(:each)
070:       x.each { |other| yield other }
071:     end
072:   end
073: end
074:
075: def setup_code_units
076:   code_elements = ERBDocument.extract_ruby_code_elements(@content)
077:   ERBDocument.setup_code_units(code_elements, @content)
078: end
079:
080: def get_atomic_sections
081:   get_atomic_sections_recursive((@atomic_sections || []) + (@content || []))
082: end
083:
084: def get_local_transitions(source)
085:   []
086: end
087:
088: def get_transitions
089:   get_transitions_recursive((@atomic_sections || []) + (@content || []))
090: end
091:
092: def identify_atomic_sections
093:   section = AtomicSection.new
094:   @atomic_sections ||= []
095:   create_section = lambda do |cur_sec|
096:     @atomic_sections << cur_sec
097:     AtomicSection.new(cur_sec.count+1)
098:   end
099:   each do |child_node|
100:     if child_node.browser_output?
101:       unless section.try_add_node?(child_node)
102:         section = create_section.call(section)
103:         # section.parent = child_node
104:         section.try_add_node?(child_node)
105:       end
106:     elsif section.content.length > 0

```

```

107:         section = create_section.call(section)
108:     #         section.parent = child_node
109:     end
110: end
111: # Be sure to get the last section appended if it was a valid one,
112: # like in the case of an ERBDocument with a single node
113: @atomic_sections << section if section.content.length > 0
114: end
115:
116: def initialize_content
117:     @initialized_content = false
118:     @content = []
119:     each do |element|
120:         if element.respond_to?(:parent=)
121:             element.parent = self
122:         end
123:         @content << element
124:     end
125:     @initialized_content = true
126: end
127:
128: def initialize_indices
129:     each_with_index do |element, i|
130:         element.index = i
131:     end
132: end
133:
134: def inspect
135:     file_details = sprintf("Source file: %s", @source_file)
136:     sections = get_sections_and_nodes(:to_s)
137:     sprintf("%s\n%s", file_details, sections.join("\n"))
138: end
139:
140: # Returns the number of HTML, ERB, and text nodes in this document
141: def length
142:     if @initialized_content
143:         @content.length
144:     else
145:         1 + (x.respond_to?(:length) ? x.length : 0)
146:     end
147: end
148:
149: def pair_tags
150:     mateless = []
151:     each_with_index do |element, i|
152:         next unless element.respond_to? :pair_match?
153:         # Find first matching mate for this element in the array of mateless
154:         # elements. First matching mate will be latest added element.
155:         mate = mateless.find { |el| el.pair_match?(element) }
156:         if mate.nil?

```

```

157:      # Add mate to beginning of mateless array, so array is sorted by
158:      # most-recently-found to earliest-found.
159:      mateless.insert(0, element)
160:    else
161:      if mate.respond_to? :close
162:        mate.close = element
163:        mateless.delete(mate)
164:      else
165:        raise "Mate found out of order: " + mate.to_s + ", " + element.to_s
166:      end
167:    end
168:  end
169: end
170:
171: def save_atomic_sections(base_dir='.')
172:   all_sections = get_atomic_sections()
173:   if all_sections.nil? || all_sections.empty?
174:     raise "No atomic sections to write to file"
175:   end
176:   dir_name = sprintf("atomic_sections-%s",
177:     File.basename(@source_file).gsub(/\./, '_'))
178:   dir_path = File.join(base_dir, dir_name)
179:   puts sprintf("Creating directory %s...", dir_path)
180:   Dir.mkdir(dir_path)
181:   all_sections.collect do |section|
182:     file_name = sprintf("%04d.txt", section.count)
183:     file_path = File.join(dir_path, file_name)
184:     puts sprintf("Writing atomic section to file %s...", file_name)
185:     section.save(file_path)
186:     file_path
187:   end
188: end
189:
190: def split_out_erb_newlines
191:   index = 0
192:   num_children = @content.length
193:   while index < num_children
194:     child = @content[index]
195:     unless child.is_a?(ERBTag)
196:       index += 1
197:       next
198:     end
199:     code = child.ruby_code()
200:     unless code =~ STATEMENT_END
201:       index += 1
202:       next
203:     end
204:     split_code = code.split(STATEMENT_END)
205:     contained_units = ERBDocument.get_code_units(split_code)
206:     if contained_units.empty?

```

```

207:     contained_units = ERBDocument.split_out_ends(split_code)
208: end
209: #puts "This code:"
210: #pp split_code
211: #puts "Becomes these code units:"
212: #pp contained_units
213: unless contained_units.empty?
214:   child.override_ruby_code = code
215:   contained_units.each do |code_line|
216:     cur_code = child.override_ruby_code
217:     before_chunk, after_chunk =
218:       ERBDocument.get_before_and_after_code(code_line, cur_code, split_code)
219:     replacement_code = if before_chunk.nil? || before_chunk.blank?
220:       child_placement = -1
221:       after_chunk || ''
222:     elsif after_chunk.nil? || after_chunk.blank?
223:       child_placement = 1
224:       before_chunk || ''
225:     else
226:       # New child in between chunks of code.
227:       # Preserve the first chunk in the existing
228:       # node, we'll create a new node with the
229:       # middle chunk, and create a new node with
230:       # the final chunk.
231:       child_placement = 0
232:       before_chunk || ''
233:     end
234:   unless replacement_code.blank?
235:     child.sexp = nil
236:     new_child = child.dup()
237:     new_child.override_ruby_code = code_line
238:     child.override_ruby_code = replacement_code
239:     #puts "Replacing #{cur_code}\nWith #{child.override_ruby_code}"
240:     case child_placement
241:     when -1 then
242:       #puts "Inserting new child before old child:\nNew child: " +
243:       # new_child.to_s + "\nOld child: " + child.to_s
244:       @content.insert(index, new_child)
245:     when 0 then
246:       #puts "Inserting new child after old child:\nOld child: " +
247:       # child.to_s + "\nNew child: " + new_child.to_s
248:       @content.insert(index+1, new_child)
249:       after_child = child.dup()
250:       after_child.override_ruby_code = after_chunk
251:       #puts "Inserting final new child: " + after_child.to_s
252:       @content.insert(index+2, after_child)
253:     when 1 then
254:       #puts "Inserting new child after old child:\nOld child: " +
255:       # child.to_s + "\nNew child: " + new_child.to_s
256:       @content.insert(index+1, new_child)

```

```

257:         end
258:     end
259: end
260: end
261:     index += 1
262: end # while
263: end
264:
265: def to_s(indent_level=0)
266:     map(&:to_s).select { |str| !str.blank? }.join("\n")
267: end
268:
269: private
270:
271: def delete_node_check_size(node_to_del)
272:     size_before = @content.length
273:     del_node_str = node_to_del.to_s
274:     @content.delete(node_to_del)
275:     if size_before - @content.length > 1
276:         raise "Deleted more than one node equaling\n" + del_node_str
277:     end
278: end
279:
280: def self.find_code_start_within_code(needle, haystack, split_needle)
281:     if needle.nil? || !needle.is_a?(String)
282:         raise ArgumentError, "Expected needle to be a string, got " + needle.class.name
283:     end
284:     if haystack.nil? || !haystack.is_a?(String)
285:         raise ArgumentError, "Expected haystack to be a string, got " + haystack.class.name
286:     end
287:     code_start = haystack.index(needle)
288:     return code_start unless code_start.nil?
289:     if split_needle.nil? || !split_needle.is_a?(Array)
290:         raise ArgumentError, "Expected split_needle to be non-nil array, got " +
split_needle.class.name
291:     end
292:     # Lines of code may have been rejoined with \n when originally they were
293:     # only separated with ;, for example
294:     code_starts = split_needle.collect do |needle_piece|
295:         if needle_piece.strip.blank?
296:             :whitespace
297:         else
298:             haystack.index(needle_piece)
299:         end
300:     end
301:     if code_starts.empty?
302:         raise "Given split needle had no pieces:\nNeedle: #{needle}\nHaystack: #{haystack}\nSplit
needle: #{split_needle.inspect}"
303:     end
304:     length_of_separator = 1

```

```

305:      #puts "Code starts: [" + code_starts.collect { |c| (c || 'nil').to_s }.join(', ') + "]"
306: code_starts.each_with_index do |code_start, i|
307:   cur_needle = split_needle[i]
308:   if 0 == i
309:     if code_start.nil?
310:       return nil
311:       #raise "Could not find needle chunk ::#{cur_needle}:: within haystack #{haystack}"
312:     end
313:   elsif :whitespace != code_start
314:     prev_code_start = nil
315:     prev_index = i-1
316:     length_between = 0
317:     while prev_index >= 0 && :whitespace == code_starts[prev_index]
318:       length_between += split_needle[prev_index].length + length_of_separator
319:       prev_index -= 1
320:     end
321:     prev_code_start = code_starts[prev_index]
322:     prev_needle = split_needle[prev_index]
323:
324:     if prev_code_start.nil?
325:       raise "Could not find needle chunk #{prev_needle} within haystack #{haystack}"
326:     end
327:
328:     expected_code_start = prev_code_start + prev_needle.length + length_between +
length_of_separator
329:     #puts "Expected code start #{expected_code_start}, instead got #{code_start}"
330:
331:     if code_start.nil? || code_start != expected_code_start
332:       raise "Could not find ::#{needle}:: within ::#{haystack}:: in order to split
multiple ERB statements in a single ERB tag into separate ERB tags; specifically could not
find #{cur_needle}"
333:     end
334:   end
335: end
336:
337:   #puts "Index of #{needle} within #{haystack} is #{code_starts[0]}"
338:
339:   # We did find all the chunks of the split_needle consecutively within
340:   # the haystack, so we can return the index of where the first chunk of
341:   # the split_needle was found in the haystack
342:   code_starts[0]
343: end
344:
345: def get_atomic_sections_recursive(nodes=[])
346:   sections = []
347:   get_node_sections = lambda do |node|
348:     next if node.nil?
349:     sections << node if node.is_a?(AtomicSection)
350:     if node.respond_to?(:content) && !node.content.nil?
351:       sections += get_atomic_sections_recursive(node.content)

```

```

352:     end
353:     if node.respond_to?(:atomic_sections) && !node.atomic_sections.nil?
354:       sections += node.atomic_sections
355:     end
356:   end
357:   nodes.each(&get_node_sections)
358:   nodes.select do |node|
359:     node.respond_to?(:close) && !node.close.nil?
360:   end.map(&:close).each(&get_node_sections)
361:   sections
362: end
363:
364: def self.get_before_and_after_code(code_line, cur_code, split_code)
365:   default = [nil, nil]
366:   return default if code_line == cur_code
367:   replace_index = find_code_start_within_code(code_line, cur_code, split_code)
368:   return default if replace_index.nil?
369:   replace_index_end = replace_index + code_line.length
370:   before_chunk = cur_code[0...replace_index]
371:   after_chunk = cur_code[replace_index_end+1...cur_code.length]
372:   [before_chunk, after_chunk]
373: end
374:
375: def get_transitions_recursive(nodes=[])
376:   trans = []
377:   nodes.each do |node|
378:     if node.respond_to?(:transitions)
379:       trans += node.transitions || []
380:     end
381:     if node.respond_to?(:content) && !node.content.nil?
382:       trans += get_transitions_recursive(node.content)
383:     end
384:   end
385:   trans
386: end
387:
388: def self.extract_ruby_code_elements(nodes)
389:   code_els = []
390:   nodes.each do |el|
391:     if RubyCodeTypes.include?(el.class)
392:       code_els << el
393:     elsif el.respond_to?(:text_value)
394:       #puts "Converting type " + el.class.name + " to FakeERBOutput"
395:       code_els << FakeERBOutput.new(el.text_value, el.index)
396:     end
397:     if el.respond_to?(:content) && !(content = el.content).nil?
398:       # Recursively check content of this node for other code elements
399:       code_els += extract_ruby_code_elements(content)
400:     end
401:   end

```



```

402:     code_els
403: end
404:
405: def self.test_only_real_code_first?(unit_elements)
406:   return false if unit_elements.nil? || unit_elements.empty?
407:   classes = unit_elements.map(&:class)
408:   return false unless classes.include?(FakeERBOutput)
409:   num_total = classes.length
410:   num_fake = classes.select { |c| c == FakeERBOutput }.length
411:   return false if num_total == num_fake
412:   ratio = (1.0 * num_fake) / num_total
413:   if $DEBUG
414:     printf("%d fake elements / %d elements = %0.2f per cent fake\n",
415:           num_fake, num_total, ratio*100)
416:   end
417:   ratio > 0.5
418: end
419:
420: def self.code_unit_iterator(code_elements, code_method=nil)
421:   unless block_given?
422:     raise ArgumentError, "Block required for code unit iterator"
423:   end
424:   num_elements = code_elements.length
425:   start_index = end_index = 0
426:   parser = RubyParser.new
427:   found_unit = false
428:   while start_index < num_elements
429:     while end_index < num_elements
430:       range = start_index..end_index
431:       unit_elements = code_elements[range]
432:       erb_elements = unit_elements.select do |e|
433:         e.is_a?(ERBTag)
434:       end
435:       try_parse_code(parser, erb_elements, code_method) do |sexp, joined_lines|
436:         found_unit = true
437:       end
438:       if found_unit
439:         found_unit = false
440:         # Try parsing again, but with all the non-ERBTag included
441:         try_parse_code(parser, unit_elements, code_method) do |sexp, joined_lines|
442:           yield(sexp, joined_lines, unit_elements)
443:           start_index += 1
444:           found_unit = true
445:         end
446:       end
447:
448:       if found_unit
449:         end_index = start_index
450:         break
451:       else

```

```

452:         end_index += 1
453:     end
454: end
455:
456:     if found_unit
457:         found_unit = false
458:     else
459:         # Once finding an outer code unit, should then check
460:         # start_index+1 to end_index-1 and so on inward to see if
461:         # any inner, nested code units exist
462:         start_index += 1
463:         end_index = start_index
464:     end
465: end
466: end
467:
468: def self.try_parse_code(parser, unit_elements, code_method)
469:     if code_method.nil?
470:         unit_lines = unit_elements
471:     else
472:         unit_lines = unit_elements.map { |l| l.send(code_method) }
473:     end
474:     joined_lines = unit_lines.join("\n")
475:     # Call #dup because otherwise end up with pound sign added to
476:     # beginning (!):
477:     sexp = parser.parse(joined_lines.dup())
478:     # Since we made it past the parse(), these lines of Ruby code
479:     # are valid together
480:     yield(sexp, joined_lines)
481: rescue Racc::ParseError
482: rescue SyntaxError
483:     # Can occur when lines are split on ; and this happens in the
484:     # middle of a string
485: end
486:
487: def self.setup_code_units(code_elements, content)
488:     #puts "All code elements:"
489:     #pp code_elements
490:     code_unit_iterator(code_elements,
491:         :ruby_code) do |sexp, joined_lines, unit_elements|
492:         setup_code_unit(unit_elements, sexp, content)
493:     end
494: end
495:
496: def self.get_code_units(code_elements)
497:     code_units = []
498:     code_unit_iterator(code_elements) do |sexp, joined_lines, unit_lines|
499:         code_units << joined_lines
500:     end
501:     code_units

```

```

502:   end
503:
504:   def self.setup_code_unit(unit_elements, sexp, content)
505:     len = unit_elements.length
506:     if len < 1
507:       raise "Woah, how can I set up a code unit with no lines of code?"
508:     end
509:     opening = unit_elements.first
510:     opening.sexp = sexp if opening.respond_to?(:sexp=) && opening.sexp.nil?
511:     if len < 2
512:       #puts "--Found code unit:"
513:       #puts opening
514:       return
515:     end
516:     opening_tag_has_close = opening.respond_to?(:close)
517:     opening_tag_has_parent = opening.respond_to?(:parent) && !opening.parent.nil?
518:     if opening_tag_has_close
519:       opening.close = unit_elements.last
520:       if opening.close.respond_to?(:parent=)
521:         opening.close.parent = opening
522:       end
523:       if opening_tag_has_parent
524:         opening.parent.delete_children_in_range(opening.close.index, opening.close.index)
525:       end
526:     end
527:     included_content = content.select do |el|
528:       el.index > opening.index && (!opening_tag_has_close || el.index < opening.close.index)
529:     end
530:     #puts "--Found code unit:"
531:     #puts opening
532:     if opening.respond_to?(:content=)
533:       included_content.each do |child|
534:         if child.respond_to?(:parent=)
535:           child.parent = opening
536:         end
537:       end
538:       included_content.sort! { |a, b| section_and_node_sort(a, b) }
539:       opening.content = included_content
540:       #puts "--Now looking for code units in content:"
541:       #puts extract_ruby_code_elements(opening.content).map(&:to_s).join(",\n")
542:       #puts "---"
543:       setup_code_units(extract_ruby_code_elements(opening.content), opening.content)
544:     end
545:   end
546:
547:   def self.split_out_ends(code_lines)
548:     code_lines.collect do |line|
549:       unless line.nil?
550:         normalized_line = line.strip.downcase
551:         if 'end' == normalized_line || '}' == normalized_line

```

```

552:         line
553:     end
554: end
555: end.compact
556: end
557: end
558: end
559:

```

2.4 erb_document_test.rb

```

001: base_path = File.expand_path(File.dirname(__FILE__))
002: require File.join(base_path, '..', 'parser.rb')
003: require File.join(base_path, 'test_helper.rb')
004:
005: class ERBDocumentTest < Test::Unit::TestCase
006:   def test_elsif_component_expression
007:     assert_component_expression(fixture('elsif.html'),
008:                               'elsif.html.erb',
009:                               '(p1|p2|p3|p4)')
010:   end
011:
012:   def test_nested_case_when_elsif_component_expression
013:     assert_component_expression(fixture('nested_case_when.html'),
014:                               'nested_case_when.html.erb',
015:                               '((p1|p2|p3)|(p4|p5|p6)|p7)')
016:   end
017:
018:   def test_form_transitions
019:     doc = assert_doc(fixture('_add_updates.html'),
020:                     '_add_updates.html.erb')
021:     trans = doc.get_transitions()
022:     assert_not_nil trans
023:     assert_equal 2, trans.length
024:     trans.each do |transition|
025:       assert_equal transition.class, FormTransition
026:     end
027:   end
028:
029:   def test_loop_if_loop_component_expression
030:     assert_component_expression(fixture('game_index1-sans_unless.html'),
031:                               'game_index1-sans_unless.html.erb',
032:                               'p1.(p2|(p3.p4*.p5)).p6')
033:   end
034:
035:   def test_unless_loop_component_expression
036:     assert_component_expression(fixture('unless_loop.html'),
037:                               'unless_loop.html.erb',
038:                               '((p1.p2*.p3)|NULL)')
039:   end

```

```

040:
041: def test_case_when_component_expression
042:   assert_component_expression(fixture('case_when.html'),
043:                               'case_when.html.erb',
044:                               '(p1|p2|p3|p4)')
045: end
046:
047: def test_nested_elsif_component_expression
048:   assert_component_expression(fixture('nested_elsif.html'),
049:                               'nested_elsif.html.erb',
050:                               '((p1|p2)|p3|p4)')
051: end
052:
053: def test_close_branch_component_expression
054:   assert_component_expression(fixture('_add_updates.html'),
055:                               '_add_updates.html.erb',
056:                               '((p1|p2)|NULL)')
057: end
058:
059: def test_lvar_component_expression
060:   # Sometimes the sexp for ERB code can differ based on context, like if
061:   # "thing.blah()" is parsed versus "if thing = junk; thing.blah(); end"
062:   # is parsed.
063:   assert_component_expression(fixture('short_edit.html'),
064:                               'short_edit.html.erb',
065:                               '(p1|p2)')
066: end
067:
068: def test_loops_component_expression
069:   assert_component_expression(fixture('loops.html'),
070:                               'loops.html.erb',
071:                               'p1*.p2*.p3*.p4***')
072: end
073:
074: def test_form_tag_component_expression
075:   assert_component_expression(fixture('login_index.html'),
076:                               'login_index.html.erb',
077:                               'p1')
078: end
079:
080: def test_javascript_component_expression
081:   assert_component_expression(fixture('javascript.html'),
082:                               'javascript.html.erb',
083:                               '(p1|NULL)')
084: end
085:
086: def test_multiple_statements_in_erb_tags_component_expression
087:   assert_component_expression(fixture('multiple_lines_in_erb.html'),
088:                               'multiple_lines_in_erb.html.erb',
089:                               'p1.p2')

```

```

090: end
091:
092: def test_multiple_lines_in_erb_tags_component_expression
093:   assert_component_expression(fixture('nested_loop.html'),
094:                               'nested_loop.html.erb',
095:                               'p1.p2.p3*.p4')
096: end
097:
098: def test_multiple_erb_lines_unequal_ifs_component_expression
099:   assert_component_expression(fixture('_in_progress.html'),
100:                               '_in_progress.html.erb',
101:                               '(((p1|NULL).p2)|p3)|NULL).p4.p5.p6*.p7.p8*.p9.p10.p11.p12.(p13|p14)')
102: end
103:
104: def test_nested_unequal_ifs_component_expression
105:   assert_component_expression(fixture('nested_unequal_ifs.html'),
106:                               'nested_unequal_ifs.html.erb',
107:                               "((p1|NULL).p2)|p3)")
108: end
109:
110: def test_nested_aggregation_component_expression
111:   assert_component_expression(fixture('game_index2.html'),
112:                               'game_index2.html.erb',
113:                               "p1.(p2|(p3.p4*.p5))*p6")
114: end
115:
116: def test_nested_aggregation_selection_component_expression
117:   assert_component_expression(fixture('game_index1.html'),
118:                               'game_index1.html.erb',
119:                               '(p1.(p2|(p3.p4*.p5))*p6)|NULL)')
120: end
121:
122: def test_nested_if_and_aggregation_component_expression
123:   assert_component_expression(fixture('top_records.html'),
124:                               'top_records.html.erb',
125:                               'p1.(p2|(p3.{p4}.p5)).p6.(p7|(p8.{p9}.p10)).p11')
126: end
127:
128: def test_nested_if_and_loop_component_expression
129:   assert_component_expression(fixture('_finished.html'),
130:                               '_finished.html.erb',
131:                               '(p1|p2)|NULL).(p3|NULL).(p4|NULL).p5.p6*.p7')
132: end
133:
134: def test_delete_node
135:   doc = Parser.new.parse(fixture('login_index.html'), 'login_index.html.erb', URI.parse('/'))
136:   assert_not_nil doc
137:   form = doc[0]
138:   assert_not_nil form
139:   assert_equal "ERBGrammar::ERBTag", form.class.name

```

```

140:   old_length = doc.length
141:   deleted_node = doc.content.delete(form)
142:   assert_equal form, deleted_node, "Expected returned deleted_node to match form"
143:   assert_not_equal form, doc[0], "New node in index 0 should not be the same as the one
we just deleted"
144:   new_length = doc.length
145:   assert_equal old_length-1, new_length, "New length of ERBDocument should be 1 less than
old length"
146: end
147:
148: def test_nested_atomic_section
149:   doc = Parser.new.parse(fixture('_finished.html'), '_finished.html.erb', URI.parse('/'))
150:   assert_not_nil doc
151:   # The code in question:
152:   # <% #Check the state of the game and write out the winners, losers, and drawers.
153:   #   #Then display the final scores.
154:   #   if @winner %>
155:   #     <% if @winner.id == session[:user][:id] %>
156:   #       <p class="game_result_positive">You won!</p>
157:   #     <% else %>
158:   #       <p class="game_result_negative"><%= @winner.email %> won!</p>
159:   #     <% end %>
160:   # <% end %>
161:   if_winner = doc[2]
162:   assert_not_nil if_winner
163:   assert_equal "ERBGrammar::ERBTag", if_winner.class.name, "Wrong type of node in slot 0
of ERBDocument"
164:   assert_not_nil if_winner.content, "Nil content in if-winner ERBTag\nTree: " + doc.to_s
165:   nodes = if_winner.get_sections_and_nodes()
166:   assert_equal 1, nodes.length, "Expected one ERBTag child node of if-winner ERBTag"
167:   if_winner_equal = nodes.first
168:   sections = if_winner_equal.get_sections_and_nodes().select do |child|
169:     child.is_a?(AtomicSection)
170:   end
171:   assert_equal 1, sections.length, "Expected one atomic section child of if-winner-equal
ERBTag: " + sections.inspect
172:   assert_not_nil if_winner_equal.branch_content, "Expected non-nil branch_content for if-winner-equal
ERBTag"
173:   assert_not_equal 1, if_winner_equal.branch_content.length
174:   end_tag = if_winner_equal.close
175:   assert_not_nil end_tag, "Expected 'end' to be close of: " + if_winner_equal.to_s
176:   assert_equal "end", end_tag.ruby_code()
177: end
178:
179: def test_square_bracket_accessor_fixnum
180:   doc = Parser.new.parse(fixture('login_index.html'), 'login_index.html.erb', URI.parse('/'))
181:   assert_not_nil doc
182:   form = doc[0]
183:   assert_not_nil form
184:   assert_equal "ERBGrammar::ERBTag", form.class.name

```

```

185:   label = form.content.find { |c| 7 == c.index }
186:   assert_not_nil label
187:   assert_equal "ERBGrammar::HTMLOpenTag", label.class.name
188:   assert_equal "label", label.name
189:   assert_equal 1, label.attributes.length,
190:     "Should be one attribute on HTML label tag: " + label.to_s
191:   assert_equal 'for', label.attributes.first.name,
192:     "First attribute on label tag should be 'for'"
193:   end_of_block = doc[doc.length-1]
194:   assert_not_nil end_of_block
195:   assert_equal "ERBGrammar::ERBTag", end_of_block.class.name
196: end
197:
198: def test_square_bracket_accessor_range
199:   doc = Parser.new.parse(fixture('login_index.html'), 'login_index.html.erb', URI.parse('/'))
200:   assert_not_nil doc
201:   elements = doc[0..1]
202:   assert_equal Array, elements.class, "Expected Array return value"
203:   assert_equal 1, elements.length, "Expected one element"
204:   assert_equal "ERBGrammar::ERBTag", elements[0].class.name
205:   assert_equal 0, elements[0].index
206: end
207:
208: def test_length
209:   doc = Parser.new.parse(fixture('login_index.html'), 'login_index.html.erb', URI.parse('/'))
210:   assert_not_nil doc
211:   assert_equal 1, doc.length,
212:     "ERB document has all nodes nested within a form_tag, so doc should have length
213: 1"
214: end
215: private
216: def assert_component_expression(erb, file_name, expected)
217:   doc = assert_doc(erb, file_name)
218:   actual = doc.component_expression()
219:   assert_equal expected, actual, "Wrong component expression for " + file_name
220: end
221:
222: def assert_doc(erb, file_name)
223:   rails_path = File.join("app", "views", "test", file_name)
224:   doc = Parser.new.parse(erb, rails_path, URI.parse('http://example.com/'))
225:   assert_not_nil doc
226:   doc
227: end
228: end
229:

```


2.5 erb_grammar.treetop

```
001: # Thanks to http://erector.rubyforge.org/ for their RHTML.treetop and
002: # http://github.com/threedaymonk/treetop-example/blob/master/complex\_html.treetop
003: # for the basis of this grammar.
004: grammar ERBGrammar
005:   rule document
006:     whitespace node whitespace x:document? <ERBDocument>
007:   end
008:
009:   rule node
010:     erb_yield / erb_output / erb / html_doctype / html_directive / html_self_closing_tag
011:     / html_close_tag / html_open_tag / text
012:   end
013:
014:   rule erb_yield
015:     '<%= ' whitespace 'yield' whitespace erb_close_bracket <ERBYield>
016:   end
017:
018:   rule erb_output
019:     '<%= ' whitespace code:ruby_code whitespace erb_close_bracket <ERBOutputTag>
020:   end
021:
022:   rule erb
023:     '<% ' whitespace code:ruby_code whitespace erb_close_bracket <ERBTag>
024:   end
025:
026:   rule erb_close_bracket
027:     '%>'
028:   end
029:
030:   rule newline
031:     [\n\r]
032:   end
033:
034:   rule ruby_code
035:     (('%' '!' '>' ' ') / ['^%'])* <RubyCode>
036:   end
037:
038:   rule tab
039:     "\t"
040:   end
041:
042:   rule whitespace
043:     (newline / tab / [\s])* <Whitespace>
044:   end
045:
046:   rule text
047:     (((<>|!(html_tag_name / ['%!'])) / [^<>]))+ <Text>
048:   end
```

```

048:
049: rule html_open_tag
050:   '<' tag_name:html_tag_name attrs:html_tag_attributes? whitespace '>' <HTMLOpenTag>
051: end
052:
053: rule html_self_closing_tag
054:   '<' tag_name:html_tag_name attrs:html_tag_attributes? whitespace '/>' <HTMLSelfClosingTag>
055: end
056:
057: rule html_close_tag
058:   "&</" tag_name:html_tag_name ">" <HTMLCloseTag>
059: end
060:
061: rule html_tag_name
062:   [A-Za-z0-9_-]+
063: end
064:
065: rule html_doctype
066:   '<!DOCTYPE' [^>]* '>' <HTMLDoctype>
067: end
068:
069: rule html_directive
070:   '<!' [^>]* '>' <HTMLDirective>
071: end
072:
073: rule html_tag_attributes
074:   head:html_tag_attribute tail:html_tag_attributes* <HTMLTagAttributes>
075: end
076:
077: rule html_tag_attribute
078:   whitespace n:(html_tag_name) whitespace '=' whitespace v:quoted_value whitespace <HTMLTagAttribute>
079: end
080:
081: rule quoted_value
082:   (('"' val:(["]*) '"' / ('\'\' val:([']*) '\'\')) {
083:     def convert
084:       extract_erb(val.text_value)
085:     end
086:
087:     def parenthesize_if_necessary(s)
088:       return s if s.strip =~ /\^(.|\.)$/ || s =~ /\^[A-Z0-9_]*$/i
089:       "(" + s + ")"
090:     end
091:
092:     def extract_erb(s, parenthesize = true)
093:       if s =~ /\^(.|\.)?<%=(.|\.)?%>(.|\.)?$/
094:         pre, code, post = $1, $2, $3
095:         out = ""
096:         out = "'#{pre}'" + " unless pre.length == 0
097:         out += parenthesize_if_necessary(code.strip)

```

```

098:     unless post.length == 0
099:       post = extract_erb(post, false)
100:       out += " + #{post}"
101:     end
102:     out = parenthesize_if_necessary(out) if parenthesize
103:     out
104:   else
105:     "\"" + s + "\""
106:   end
107: end
108:
109: def to_s(indent_level=0)
110:   to_s_with_prefix(indent_level, val.text_value)
111: end
112: }
113: end
114: end
115:

```

2.6 erb_node_extensions.rb

```

01: module ERBGrammar
02:   BasePath = File.expand_path(File.dirname(__FILE__))
03:   require File.join(BasePath, 'shared_atomic_section_methods.rb')
04:   require File.join(BasePath, 'shared_children_methods.rb')
05:   require File.join(BasePath, 'shared_methods.rb')
06:   require File.join(BasePath, 'shared_erb_methods.rb')
07:   require File.join(BasePath, 'shared_html_tag_methods.rb')
08:   require File.join(BasePath, 'shared_open_tag_methods.rb')
09:   require File.join(BasePath, 'shared_sexp_methods.rb')
10:   require File.join(BasePath, 'shared_sexp_parsing.rb')
11:   require File.join(BasePath, 'shared_transition_methods.rb')
12:   require File.join(BasePath, 'erb_document.rb')
13:   require File.join(BasePath, 'erb_output_tag.rb')
14:   require File.join(BasePath, 'fake_erb_output.rb')
15:   require File.join(BasePath, 'erb_tag.rb')
16:   require File.join(BasePath, 'html_open_tag.rb')
17:   require File.join(BasePath, 'html_close_tag.rb')
18:   require File.join(BasePath, 'html_self_closing_tag.rb')
19:   require File.join(BasePath, 'html_tag_attributes.rb')
20:   require File.join(BasePath, 'html_tag_attribute.rb')
21:   require File.join(BasePath, 'html_quoted_value.rb')
22:   require File.join(BasePath, 'ruby_code.rb')
23:   require File.join(BasePath, 'text.rb')
24:   require File.join(BasePath, 'whitespace.rb')
25:   require File.join(BasePath, 'html_directive.rb')
26:   require File.join(BasePath, 'html_doctype.rb')
27:   require File.join(BasePath, 'erb_yield.rb')
28:   require File.join(BasePath, 'syntax_node.rb')
29: end

```

30:

2.7 erb_output_tag.rb

```
01: module ERBGrammar
02:   class ERBOutputTag < Treetop::Runtime::SyntaxNode
03:     include SharedAtomicSectionMethods
04:     extend SharedAtomicSectionMethods::ClassMethods
05:     include SharedERBMethods
06:     include SharedSexpMethods
07:     extend SharedSexpMethods::ClassMethods
08:     include SharedSexpParsing
09:     include SharedTransitionMethods
10:     LINK_METHODS = [:link_to, :link_to_remote, :link_to_unless_current,
11:       :link_to_unless, :link_to_if, :link_to_function].freeze
12:     attr_accessor :atomic_section_count
13:
14:     def content
15:       nil
16:     end
17:
18:     def get_local_transitions(source)
19:       set_sexp() if @sexp.nil?
20:       get_link_transitions(source)
21:     end
22:
23:     def inspect
24:       sprintf("%s (%d): %s", self.class, @index, ruby_code())
25:     end
26:
27:     def ruby_code
28:       code.content_removing_trims()
29:     end
30:
31:     def to_s(indent_level=0)
32:       to_s_with_prefix(indent_level, '<%= ' + ruby_code())
33:     end
34:
35:     private
36:     def get_link_transitions(source)
37:       transitions = []
38:       LINK_METHODS.each do |link_method|
39:         link_args = ERBOutputTag.get_sexp_for_call_args(sexp, link_method)
40:         unless link_args.nil?
41:           sink = get_target_page_from_sexp(link_args)
42:           unless sink.nil?
43:             transitions << LinkTransition.new(source, sink, ruby_code())
44:           end
45:         end
46:       end
47:     end
48:   end
49: end
```

```

47:     transitions
48:   end
49: end
50: end
51:

```

2.8 erb_tag.rb

```

01: module ERBGrammar
02:   class ERBTag < Treetop::Runtime::SyntaxNode
03:     include SharedAtomicSectionMethods
04:     extend SharedAtomicSectionMethods::ClassMethods
05:     include SharedChildrenMethods
06:     include SharedERBMethods
07:     include SharedSexpMethods
08:     extend SharedSexpMethods::ClassMethods
09:     include SharedTransitionMethods
10:     include SharedOpenTagMethods
11:     FORM_METHODS = [:form_tag, :form_remote_tag, :form_for, :remote_form_for,
12:       :form_remote_for].freeze
13:     REDIRECT_METHODS = [:redirect_to, :redirect_to_full_url].freeze
14:     attr_accessor :content, :parent, :close, :branch_content, :overridden_ruby_code
15:
16:     def atomic_section_str(indent_level=0)
17:       if @atomic_sections.nil?
18:         ''
19:       else
20:         @atomic_sections.collect do |section|
21:           section.to_s(indent_level)
22:         end.join("\n") + "\n"
23:       end + close_str(indent_level)
24:     end
25:
26:     def get_local_transitions(source)
27:       set_sexp() if @sexp.nil?
28:       trans = get_form_transitions(source)
29:       trans += get_redirect_transitions(source)
30:       trans
31:     end
32:
33:     def inspect
34:       sprintf("%s (%d): %s\n%s", self.class, @index, ruby_code, content_str())
35:     end
36:
37:     def ruby_code
38:       if @overridden_ruby_code.nil?
39:         code.content_removing_trims()
40:       else
41:         @overridden_ruby_code
42:       end

```

```

43:   end
44:
45:   def to_s(indent_level=0)
46:     sections = get_sections_and_nodes(:to_s, indent_level+2)
47:     prefix = '  '
48:     #branch_str = @branch_content.nil? ? '' : sprintf("\nBranches:\n%s", @branch_content.map(&:inspect).join("\n"))
49:     content_prefix = sections.empty? ? '' : sprintf("%sContent and sections:\n", prefix *
(indent_level+1))
50:     close_string = close_str(indent_level+2)
51:     close_prefix = close_string.blank? ? '' : sprintf("%sClose:\n", prefix * (indent_level+1))
52:     to_s_with_prefix(indent_level,
53:       sprintf("%s\n%s%s\n%s%s", ruby_code, content_prefix,
54:         sections.join("\n"), close_prefix, close_string))
55:   end
56:
57:   private
58:   def get_form_transitions(source)
59:     transitions = []
60:     FORM_METHODS.each do |form_method|
61:       form_args = ERBTag.get_sexp_for_call_args(@sexp, form_method)
62:       unless form_args.nil?
63:         sink = get_target_page_from_sexp(form_args, source)
64:         unless sink.nil?
65:           transitions << FormTransition.new(source, sink, ruby_code())
66:         end
67:       end
68:     end
69:     transitions
70:   end
71:
72:   def get_redirect_transitions(source)
73:     transitions = []
74:     REDIRECT_METHODS.each do |redirect_method|
75:       redirect_args = ERBTag.get_sexp_for_call_args(@sexp, redirect_method)
76:       unless redirect_args.nil?
77:         sink = get_target_page_from_sexp(redirect_args, source)
78:         unless sink.nil?
79:           transitions << RedirectTransition.new(source, sink, ruby_code())
80:         end
81:       end
82:     end
83:     transitions
84:   end
85: end
86: end
87:

```

2.9 erb_tag_test.rb

```
01: base_path = File.expand_path(File.dirname(__FILE__))
02: require File.join(base_path, '..', 'parser.rb')
03: require File.join(base_path, 'test_helper.rb')
04:
05: class ERBTagTest < Test::Unit::TestCase
06:   def test_iteration?
07:     doc = Parser.new.parse(fixture('iteration.html'), 'iteration.html.erb', URI.parse('/'))
08:     assert_not_nil doc
09:     loops = [doc[0], doc[4], doc[5], doc[8], doc[10]]
10:     loops.each do |loop_tag|
11:       assert_not_nil loop_tag
12:       assert_equal "ERBGrammar::ERBTag", loop_tag.class.name
13:       assert loop_tag.iteration?, "Expected #{loop_tag} to be a loop of some sort"
14:     end
15:     non_loops = [doc[13], doc[16]]
16:     non_loops.each do |non_loop_tag|
17:       assert_not_nil non_loop_tag
18:       assert_equal "ERBGrammar::ERBTag", non_loop_tag.class.name
19:       assert !non_loop_tag.iteration?, "Expected #{non_loop_tag} to not be a loop of any kind"
20:     end
21:   end
22: end
23:
```

2.10 erb_yield.rb

```
01: module ERBGrammar
02:   class ERBYield < Treetop::Runtime::SyntaxNode
03:     include SharedAtomicSectionMethods
04:     extend SharedAtomicSectionMethods::ClassMethods
05:     include SharedERBMethods
06:     include SharedSexpMethods
07:     extend SharedSexpMethods::ClassMethods
08:     include SharedSexpParsing
09:
10:     def ruby_code
11:       'yield'
12:     end
13:
14:     def to_s(indent_level=0)
15:       to_s_with_prefix(indent_level, 'yield')
16:     end
17:   end
18: end
19:
```

2.11 fake_erb_output.rb

```
01: module ERBGrammar
02:   class FakeERBOutput
03:     include SharedSexpMethods
04:     extend SharedSexpMethods::ClassMethods
05:     include SharedMethods
06:     attr_reader :index, :lines_of_code, :content
07:
08:     def initialize(code, index)
09:       if code.is_a?(String)
10:         @lines_of_code = [code]
11:       elsif code.is_a?(Array)
12:         @lines_of_code = code
13:       else
14:         raise ArgumentError, "Expected String or Array code"
15:       end
16:       unless index.is_a?(Fixnum)
17:         raise ArgumentError, "Expected Fixnum index"
18:       end
19:       @content = nil
20:       @index = index
21:     end
22:
23:     def ==(other)
24:       return false if other.nil?
25:       other.respond_to?(:lines_of_code) && !other.lines_of_code.nil? &&
26:         other.lines_of_code == @lines_of_code
27:     end
28:
29:     def inspect
30:       to_s
31:     end
32:
33:     # Need a way of encapsulating non-ERB content in Ruby tags so it can be
34:     # recognized by the parser relative to the rest of the ERB code. Wrap
35:     # HTML tags, etc. in a Ruby string and 'puts' it, so it can be seen,
36:     # for example, that this particular HTML was within the 'else' portion
37:     # of an if/else block.
38:     def ruby_code
39:       @lines_of_code.collect do |code|
40:         'puts "' + FakeERBOutput.escape_value(code) + "'"
41:       end.join("\n")
42:     end
43:
44:     def to_s(indent_level=0)
45:       to_s_with_prefix(indent_level, "FakeERBOutput " + @lines_of_code.join("\n"))
46:     end
47:
48:     private
```



```

49:     def self.escape_value(value)
50:         return nil if value.nil?
51:         value.gsub(/"/, "\\\"")
52:     end
53: end
54: end
55:

```

2.12 form_transition.rb

```

01: class FormTransition < Transition
02:     # TODO: include GET/POST method
03:     def initialize(src, snk, c)
04:         super(src, snk, c)
05:     end
06:
07:     def to_s(prefix='')
08:         sprintf("%sForm Transition\n%s", prefix, super(prefix))
09:     end
10: end
11:

```

2.13 generator.rb

```

01: #!/usr/bin/env ruby
02: root_dir = File.expand_path(File.dirname(__FILE__))
03: require File.join(root_dir, 'parser.rb')
04: require 'find'
05: require File.join(root_dir, 'component_interaction_model.rb')
06:
07: unless ARGV.length == 2
08:     printf("Usage: %s path_to_rails_app_root root_url_of_site\n", $0)
09:     exit
10: end
11:
12: rails_root_path = ARGV.shift
13: begin
14:     root_url = URI.parse(ARGV.shift)
15: rescue URI::InvalidURIError => err
16:     printf("ERROR: could not parse given root URI: %s", err)
17:     exit
18: end
19:
20: app_path = File.join(rails_root_path, 'app')
21: unless File.exists?(app_path)
22:     printf("ERROR: expected app directory does not exist at %s", app_path)
23:     exit
24: end
25:
26: views_path = File.join(app_path, 'views')
27: unless File.exists?(views_path)

```

```

28:  printf("ERROR: expected app/views directory does not exist at %s", views_path)
29:  exit
30: end
31:
32: ERB_FILE_TYPES = ['rhtml', 'erb'].freeze
33: EXCLUDED_DIRS = ['.svn'].freeze
34: cims = []
35:
36: Find.find(views_path) do |path|
37:   if FileTest.directory?(path)
38:     dir_name = File.basename(path.downcase)
39:     if EXCLUDED_DIRS.include?(dir_name)
40:       Find.prune # Don't look in this directory
41:     else
42:       printf("Looking in directory %s\n", path)
43:     end
44:   else # Found a file
45:     file_type = File.basename(path.downcase).split('.').last
46:     if ERB_FILE_TYPES.include?(file_type)
47:       erb = IO.readlines(path).join
48:       if erb.nil? || erb.blank?
49:         printf("No data in file %s, skipping\n", path)
50:         next
51:       end
52:       ast = Parser.new.parse(erb, path, root_url)
53:       if ast.nil?
54:         printf("Could not parse file %s, skipping\n", path)
55:         next
56:       end
57:       expr = ast.component_expression()
58:       sections = ast.get_atomic_sections()
59:       trans = ast.get_transitions()
60:       cim = ComponentInteractionModel.new(root_url, path, expr, sections, trans)
61:       puts cim.to_s + "\n"
62:     end
63:   end
64: end
65:

```

2.14 html_close_tag.rb

```

01: module ERBGrammar
02:   class HTMLCloseTag < Treetop::Runtime::SyntaxNode
03:     include SharedHTMLTagMethods
04:     include SharedSexpParsing
05:     include SharedSexpMethods
06:     extend SharedSexpMethods::ClassMethods
07:
08:     def ==(other)
09:       super(other) && prop_eq?(other, :name)

```

```

10:     end
11:
12:   def hash
13:     prop_hash(:name)
14:   end
15:
16:   def name
17:     tag_name.text_value.downcase
18:   end
19:
20:   def inspect
21:     sprintf("%s %d: %s", self.class, @index, name)
22:   end
23:
24:   def pair_match?(other)
25:     opposite_type_same_name?(HTMLOpenTag, other)
26:   end
27:
28:   def to_s(indent_level=0)
29:     to_s_with_prefix(indent_level, '/' + name)
30:   end
31: end
32: end
33:

```

2.15 html_directive.rb

```

01: module ERBGrammar
02:   class HTMLDirective < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     include SharedHTMLTagMethods
06:     extend SharedSexpMethods::ClassMethods
07:
08:     def ==(other)
09:       super(other) && prop_eql?(other, :text_value)
10:     end
11:
12:     def hash
13:       prop_hash(:text_value)
14:     end
15:
16:     def inspect
17:       to_s
18:     end
19:
20:     def to_s(indent_level=0)
21:       to_s_with_prefix(indent_level, text_value)
22:     end
23:   end

```

```
24: end
25:
```

2.16 html_doctype.rb

```
01: module ERBGrammar
02:   class HTMLDoctype < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     include SharedHTMLTagMethods
06:     extend SharedSexpMethods::ClassMethods
07:
08:     def to_s(indent_level=0)
09:       to_s_with_prefix(indent_level, text_value)
10:     end
11:   end
12: end
13:
```

2.17 html_open_tag.rb

```
01: require File.join(
02:   File.expand_path(File.join(File.dirname(__FILE__), '..', '..'),
03:   'html_parsing.rb')
04: )
05: require 'rubygems'
06: require 'nokogiri'
07:
08: module ERBGrammar
09:   class HTMLOpenTag < Treetop::Runtime::SyntaxNode
10:     include SharedOpenTagMethods
11:     include SharedHTMLTagMethods
12:     include SharedSexpParsing
13:     include SharedSexpMethods
14:     extend SharedSexpMethods::ClassMethods
15:     include SharedTransitionMethods
16:     include SharedHtmlParsing
17:     extend SharedHtmlParsing::ClassMethods
18:     attr_accessor :content, :close
19:
20:     def ==(other)
21:       super(other) && prop_eq?(other, :name, :attributes_str)
22:     end
23:
24:     def attributes
25:       attrs.empty? ? [] : attrs.to_a
26:     end
27:
28:     def attributes_str
29:       attrs.empty? ? '' : attrs.to_s
30:     end
31:   end
32: end
```

```

31:
32: def get_local_transitions(source)
33:   trans = []
34:   tag_name = name()
35:   if source.is_a?(RailsURL)
36:     source_uri = source.to_uri()
37:   else
38:     source_uri = source
39:   end
40:   doc = Nokogiri::HTML(text_value)
41:   HTMLOpenTag.get_link_uris(source_uri, doc).each do |sink|
42:     trans << LinkTransition.new(source, RailsURL.from_uri(sink), text_value)
43:   end
44:   HTMLOpenTag.get_form_uris(source_uri, doc).each do |sink|
45:     trans << FormTransition.new(source, RailsURL.from_uri(sink), text_value)
46:   end
47:   trans
48: end
49:
50: def hash
51:   prop_hash(:name, :attributes_str)
52: end
53:
54: def name
55:   tag_name.text_value.downcase
56: end
57:
58: def inspect
59:   sprintf("%s (%d): %s %s", self.class, @index, name, attributes_str)
60: end
61:
62: def pair_match?(other)
63:   opposite_type_same_name?(HTMLCloseTag, other)
64: end
65:
66: def to_s(indent_level=0)
67:   to_s_with_prefix(indent_level, sprintf("%s %s\n%s",
68:     name, attributes_str, content_str(indent_level+1)))
69: end
70: end
71: end
72:

```

2.18 html_quoted_value.rb

```

01: module ERBGrammar
02:   class HTMLQuotedValue < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     extend SharedSexpMethods::ClassMethods

```

```

06: include SharedHTMLTagMethods
07:
08:     def ==(other)
09:         super(other) && prop_eql?(other, :value)
10:     end
11:
12:     def hash
13:         prop_hash(:value)
14:     end
15:
16:     def inspect
17:         sprintf("%s: %s", self.class, value)
18:     end
19:
20:     def to_s(indent_level=0)
21:         to_s_with_prefix(indent_level, value)
22:     end
23:
24:     def value
25:         val.text_value
26:     end
27:
28:     def convert
29:         extract_erb(val.text_value)
30:     end
31:
32:     def parenthesize_if_necessary(s)
33:         return s if s.strip =~ /\^(.*\)$ / || s =~ /\^[A-Z0-9_]*$/i
34:         "(" + s + ")"
35:     end
36:
37:     def extract_erb(s, parenthesize = true)
38:         if s =~ /\^(.*?)<%(.*?)%>(.*?)$/
39:             pre, code, post = $1, $2, $3
40:             out = ""
41:             out = "'#{pre}' + " unless pre.length == 0
42:             out += parenthesize_if_necessary(code.strip)
43:             unless post.length == 0
44:                 post = extract_erb(post, false)
45:                 out += " + #{post}"
46:             end
47:             out = parenthesize_if_necessary(out) if parenthesize
48:             out
49:         else
50:             "'" + s + "'"
51:         end
52:     end
53: end
54: end
55:

```

2.19 html_self_closing_tag.rb

```
01: module ERBGrammar
02:   class HTMLSelfClosingTag < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     extend SharedSexpMethods::ClassMethods
06:     include SharedHTMLTagMethods
07:
08:     def ==(other)
09:       super(other) && prop_eql?(other, :name, :attributes_str)
10:     end
11:
12:     def attributes
13:       attrs.empty? ? [] : attrs.to_a
14:     end
15:
16:     def attributes_str
17:       attrs.empty? ? '' : attrs.to_s
18:     end
19:
20:     def hash
21:       prop_hash(:name, :attributes_str)
22:     end
23:
24:     def name
25:       tag_name.text_value.downcase
26:     end
27:
28:     def inspect
29:       sprintf("%s: %s %s", self.class, name, attributes_str)
30:     end
31:
32:     def to_s(indent_level=0)
33:       to_s_with_prefix(indent_level, name + ' ' + attributes_str)
34:     end
35:   end
36: end
37:
```

2.20 html_tag_attribute.rb

```
01: module ERBGrammar
02:   class HTMLTagAttribute < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     extend SharedSexpMethods::ClassMethods
06:     include SharedHTMLTagMethods
07:
08:     def ==(other)
09:       super(other) && prop_eql?(other, :name, :value)
```

```

10:     end
11:
12:   def hash
13:     prop_hash(:name, :value)
14:   end
15:
16:   def name
17:     n.text_value.downcase
18:   end
19:
20:   def value
21:     v.text_value
22:   end
23:
24:   def inspect
25:     sprintf("%s: %s => %s", self.class, name, value)
26:   end
27:
28:   def to_s(indent_level=0)
29:     sprintf("%s => %s", name, value)
30:   end
31: end
32: end
33:

```

2.21 html_tag_attributes.rb

```

01: module ERBGrammar
02:   class HTMLTagAttributes < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     extend SharedSexpMethods::ClassMethods
06:     include SharedHTMLTagMethods
07:
08:     def ==(other)
09:       return false unless super(other) && index_eq?(other)
10:       this_arr = to_a
11:       other_arr = other.to_a
12:       return false if this_arr.length != other_arr.length
13:       this_arr.each_with_index do |el, i|
14:         return false unless el == other_arr[i]
15:       end
16:       true
17:     end
18:
19:     def hash
20:       h = prop_hash()
21:       to_a.each do |el|
22:         h = h ^ el.hash
23:       end

```



```

24:     h
25: end
26:
27: def to_a
28:   arr = [head]
29:   unless tail.empty?
30:     arr += tail.elements.first.to_a
31:   end
32:   arr
33: end
34:
35: def to_h
36:   hash = {}
37:   hash[head.name] = head.value
38:   unless tail.empty?
39:     hash.merge!(tail.elements.first.to_h)
40:   end
41:   hash
42: end
43:
44: def to_s(indent_level=0)
45:   to_a.map(&:to_s).join(' ', ' ')
46: end
47: end
48: end
49:

```

2.22 link_transition.rb

```

1: class LinkTransition < Transition
2:   def initialize(src, snk, c)
3:     super(src, snk, c)
4:   end
5:
6:   def to_s(prefix='')
7:     sprintf("%sLink Transition\n%s", prefix, super(prefix))
8:   end
9: end
10:

```

2.23 parser.rb

```

01: require 'rubygems'
02: require 'treetop'
03:
04: base_path = File.expand_path(File.dirname(__FILE__))
05: require File.join(base_path, 'nodes', 'erb_node_extensions.rb')
06: Treetop.load(File.join(base_path, 'erb_grammar.treetop'))
07:
08: class Parser
09:   @@parser = ERBGrammarParser.new

```

```

10:
11: def parse(data, file_name, root_url, debug_on=false)
12:   printf("Parsing ERB file %s...\n", file_name)
13:   tree = @@parser.parse data
14:   unless tree.nil?
15:     puts "Initializing content..." if debug_on
16:     tree.initialize_content()
17:     puts "Splitting out ERB newlines..." if debug_on
18:     tree.split_out_erb_newlines()
19:     puts "Initializing indices..." if debug_on
20:     tree.initialize_indices()
21:     #pp tree
22:     #puts '-----'
23:     #puts "Pairing HTML tags..."
24:     #tree.pair_tags
25:     puts "Setting up code units..." if debug_on
26:     tree.setup_code_units()
27:     puts "Identifying atomic sections..." if debug_on
28:     tree.identify_atomic_sections()
29:     puts "Nesting atomic sections..." if debug_on
30:     tree.nest_atomic_sections()
31:     puts "Splitting branches..." if debug_on
32:     tree.split_branches()
33:     puts "Removing duplicate children..." if debug_on
34:     tree.remove_duplicate_children()
35:     puts "Identifying transitions..." if debug_on
36:     src_rails_url = RailsURL.from_path(file_name, root_url)
37:     if src_rails_url.nil?
38:       printf("Could not interpret path %s as a Rails URL, skipping transition identification\n",
file_name)
39:     else
40:       tree.identify_transitions(src_rails_url, root_url)
41:     end
42:     tree.source_file = file_name
43:   end
44:   tree
45: rescue Racc::ParseError => err
46:   printf("Failed to parse %s at offset %d: %s\n", file_name, @@parser.index, err)
47: rescue SyntaxError => err
48:   printf("Failed to parse %s at offset %d: %s\n", file_name, @@parser.index, err)
49: end
50:
51: def parse_and_compress(data)
52:   tree = parse(data)
53:   puts "Compressing content..."
54:   tree.compress_content()
55:   tree
56: end
57: end
58:

```

2.24 rails_url.rb

```
01: require 'uri'
02: require File.join(File.expand_path(File.join(File.dirname(__FILE__), '..'), 'html_parsing.rb'))
03:
04: class RailsURL
05:   extend SharedHtmlParsing::ClassMethods
06:   attr_reader :action, :controller, :raw_url, :site_root
07:
08:   def initialize(ctrlr, act, raw, root='')
09:     if (ctrlr.nil? || ctrlr.blank?) && (act.nil? || act.blank?) && (raw.nil? || raw.blank?)
10:       raise ArgumentError, "Must provide at least one non-null part of URL"
11:     end
12:     @controller = (ctrlr || '').strip.downcase
13:     @action = (act || '').strip.downcase
14:     @raw_url = (raw || '').strip.downcase
15:     @site_root = root.to_s.strip.downcase
16:   end
17:
18:   def RailsURL.from_path(path, site_root='')
19:     return nil if path.nil?
20:     path_parts = path.split(File::ALT_SEPARATOR)
21:     path = File.join(path_parts)
22:     controller_prefix = File.join('app', 'views')
23:     prefix_start = path.index(controller_prefix)
24:     return nil if prefix_start.nil?
25:     controller_index = prefix_start + controller_prefix.length
26:     with_ext = path[controller_index...path.length]
27:     ext_start = with_ext.index('.') || with_ext.length
28:     without_ext = with_ext[0...ext_start]
29:     controller = File.dirname(without_ext).gsub(/^\//, '')
30:     action = File.basename(without_ext)
31:     RailsURL.new(controller, action, nil, site_root)
32:   end
33:
34:   def RailsURL.from_uri(uri)
35:     if uri.nil? || !uri.is_a?(URI)
36:       raise ArgumentError, "Expected non-nil URI, got #{uri.class.name}"
37:     end
38:     RailsURL.new(nil, nil, uri.to_s)
39:   end
40:
41:   def relative?
42:     uri = to_uri()
43:     uri.nil? ? false : uri.relative?
44:   end
45:
46:   def url
47:     if @raw_url.blank?
48:       sprintf("%s/%s/%s", @site_root, @controller, @action)
```

```

49:   else
50:     # TODO: prefix with @site_root if necessary (relative URL)
51:     @raw_url
52:   end
53: end
54:
55: def to_s
56:   #sprintf("%sRailsURL\n\t%sController: %s\n\t%sAction: %s\n\t%sRaw URL: %s",
57:   #        prefix, prefix, @controller, prefix, @action, prefix, @raw_url)
58:   url()
59: end
60:
61: def to_uri
62:   RailsURL.parse_uri_forgivingly(url())
63: end
64: end
65:

```

2.25 range.rb

```

1: class Range
2:   def <=>(other)
3:     return -1 if other.nil? || !other.respond_to?(:begin)
4:     self.begin <=> other.begin
5:   end
6: end
7:

```

2.26 redirect_transition.rb

```

1: class RedirectTransition < Transition
2:   def initialize(src, snk, c)
3:     super(src, snk, c)
4:   end
5:
6:   def to_s(prefix='')
7:     sprintf("%sRedirect Transition\n%s", prefix, super(prefix))
8:   end
9: end
10:

```

2.27 ruby_code.rb

```

01: module ERBGrammar
02:   class RubyCode < Treetop::Runtime::SyntaxNode
03:     def ==(other)
04:       super(other) && prop_eq?(other, :content_removing_trims)
05:     end
06:
07:     def hash
08:       prop_hash(:content_removing_trims)

```

```

09:   end
10:
11:   def content_removing_trims
12:     text_value.strip.gsub(/\s*\s*\s*$/, '')
13:   end
14:
15:   def to_s(indent_level=0)
16:     to_s_with_prefix(indent_level, text_value.strip)
17:   end
18: end
19: end
20:

```

2.28 runner.rb

```

1: #!/usr/bin/env ruby
2: # See http://ruby-doc.org/stdlib/libdoc/test/unit/rdoc/classes/Test/Unit.html
3: require 'test/unit'
4: base_path = File.expand_path(File.dirname(__FILE__))
5: require File.join(base_path, 'erb_tag_test')
6: require File.join(base_path, 'syntax_node_test')
7: require File.join(base_path, 'erb_document_test')
8:

```

2.29 shared_atomic_section_methods.rb

```

001: require 'set'
002: module ERBGrammar
003:   module SharedAtomicSectionMethods
004:     module ClassMethods
005:       def section_and_node_sort(a, b)
006:         comparison = a.range <=> b.range
007:         equal = 0 == comparison
008:         a_atomic = a.is_a?(AtomicSection)
009:         b_atomic = b.is_a?(AtomicSection)
010:         # Sort AtomicSections first so we don't end up repeating nodes that are
011:         # accounted for in an AtomicSection
012:         if equal && a_atomic && !b_atomic
013:           -1
014:         elsif equal && !a_atomic && b_atomic
015:           1
016:         else
017:           comparison
018:         end
019:       end
020:     end
021:     attr_reader :atomic_sections
022:
023:     def add_atomic_section(section)
024:       return if section.nil?
025:       section_index = section.index

```

```

026:     if section_index <= @index
027:       raise ArgumentError, "Cannot set section #{section} to be child of #{self}--index
is too low"
028:     end
029:     if @atomic_sections.nil? || @atomic_sections.empty?
030:       @atomic_sections = [section]
031:     else
032:       prev_section_index = @atomic_sections.index { |s| s.index > section_index }
033:       if prev_section_index.nil?
034:         @atomic_sections << section
035:       else
036:         @atomic_sections.insert(prev_section_index, section)
037:       end
038:     end
039: end
040:
041: def get_code_units_for_nesting
042:   @content.select do |el|
043:     "ERBGrammar::ERBTag" == el.class.name && !el.content.nil? && !el.content.empty?
044:   end
045: end
046:
047: def get_current_state
048:   unless respond_to?(:iteration?) && respond_to?(:aggregation?) && respond_to?(:selection?)
049:     return nil
050:   end
051:   if aggregation?
052:     :aggr
053:   elsif iteration?
054:     :iter
055:   elsif selection?
056:     :sel
057:   else
058:     nil
059:   end
060: end
061:
062: def component_expression(seen_children=[])
063:   cur_state = get_current_state()
064:   seen_children << self unless seen_children.include?(self)
065:   if :sel == cur_state
066:     expr = selection_component_expression(seen_children)
067:     return expr
068:   end
069:   children = get_sections_and_nodes()
070:   child_str = children.collect do |node|
071:     if node.respond_to?(:component_expression) && !seen_children.include?(node)
072:       seen_children << node
073:       node.component_expression(seen_children)
074:     else

```

```

075:     nil
076:   end
077: end.compact.select do |expr|
078:   !expr.blank?
079: end.join(' . ')
080: case cur_state
081:   when :iter:
082:     if child_str.nil? || child_str.blank?
083:       nil
084:     else
085:       has_single_child = (1 == children.length)
086:       open_paren = has_single_child ? ' ' : '('
087:       close_paren = has_single_child ? ' ' : ')'
088:       sprintf("%s%s%s*", open_paren, child_str, close_paren)
089:     end
090:   when :aggr:
091:     if respond_to?(:atomic_section_count) && children.empty?
092:       sprintf("{p%s}", atomic_section_count)
093:     else
094:       sprintf("{%s}", child_str)
095:     end
096:   else
097:     child_str
098:   end
099: end
100:
101: def get_sections_and_nodes(method_sym_to_call=nil, *args)
102:   atomic_sections_covered = []
103:   should_call_method = !method_sym_to_call.nil?
104:   details = (@atomic_sections || []) + (@content || [])
105:   details.sort! { |a, b| self.class.section_and_node_sort(a, b) }
106:   details.collect do |section_or_node|
107:     cur_range = section_or_node.range
108:     if atomic_sections_covered.include?(cur_range.begin)
109:       nil
110:     else
111:       atomic_sections_covered += cur_range.to_a
112:       if should_call_method
113:         section_or_node.send(method_sym_to_call, *args)
114:       else
115:         section_or_node
116:       end
117:     end
118:   end.compact
119: end
120:
121: def nest_atomic_sections
122:   code_units = get_code_units_for_nesting()
123:   return if code_units.empty?
124:   reversed_code_units = code_units.reverse

```

```

125:   (@atomic_sections.length-1).downto(0) do |i|
126:     section = @atomic_sections[i]
127:     section_range = section.range
128:     find_parent_code = lambda do |code_unit|
129:       code_range = code_unit.range
130:       code_range.include?(section_range.begin) &&
131:       code_range.include?(section_range.end)
132:     end
133:     parent_code = code_units.find(&find_parent_code)
134:     unless parent_code.nil?
135:       # Maybe the parent has another child that should actually be
136:       # the parent of this atomic section
137:       #parent_code_units = parent_code.get_code_units_for_nesting()
138:       #if parent_code_units.length > 0
139:       # parent_code = parent_code_units.find(&find_parent_code) || parent_code
140:       #end
141:       #puts "Adding atomic section"
142:       #puts section.to_s
143:       #puts "To parent"
144:       #puts parent_code.to_s
145:       #puts "
146:       parent_code.add_atomic_section(section)
147:       @atomic_sections.delete_at(i)
148:       parent_code.nest_atomic_sections()
149:     end
150:   end
151: end
152:
153: def split_branches
154:   # TODO: should I split innermost branches first, to get nested if's?
155:   # Thus should I do branch_processor on @content before self?
156:   if respond_to?(:split_branch) && respond_to?(:selection?) && selection?
157:     split_branch()
158:   end
159:   (@content || []).select do |child|
160:     !child.nil? && child.respond_to?(:split_branches)
161:   end.each do |child|
162:     child.split_branches()
163:   end
164:   if !@close.nil? && @close.respond_to?(:split_branches)
165:     @close.split_branches()
166:   end
167: end
168:
169: private
170: def selection_component_expression(seen_children=[])
171:   if !respond_to?(:branch_content)
172:     # End up here when, for example, there's an if statement within an ERBOutputTag,
173:     # e.g., <%= (user.id == session[:user][:id]) ? 'you' : user.email %>
174:     return nil

```



```

175: end
176: if @branch_content.nil?
177:   return "Has split branch, but no @branch_content is set"
178: end
179: child_selector = lambda do |n|
180:   if seen_children.include?(n) || !n.respond_to?(:component_expression)
181:     nil
182:   else
183:     seen_children << n
184:     n.component_expression(seen_children)
185:   end
186: end
187: branches_exprs = @branch_content.collect do |cur_branch_content|
188:   if cur_branch_content.nil? || cur_branch_content.empty?
189:     nil
190:   else
191:     cur_branch_exprs = cur_branch_content.map(&child_selector).compact.select do |expr|
192:       !expr.blank?
193:     end
194:     if cur_branch_exprs.empty?
195:       nil
196:     else
197:       cur_branch_str = cur_branch_exprs.join(' ')
198:       one_child = cur_branch_exprs.length == 1
199:       open_paren = one_child ? ' ' : '('
200:       close_paren = one_child ? ' ' : ')'
201:       sprintf("%s%s%s", open_paren, cur_branch_str, close_paren)
202:     end
203:   end
204: end.compact
205: return nil if branches_exprs.empty?
206: if branches_exprs.length == 1
207:   branches_exprs << 'NULL'
208: end
209: branches_str = branches_exprs.join('|')
210: branches_str = 'NULL' if branches_str.blank?
211: sprintf("(%s)", branches_str)
212: end
213:
214: def check_true_and_false_sections(true_sections, false_sections)
215:   true_set = Set.new(true_sections)
216:   false_set = Set.new(false_sections)
217:   true_and_false_sections = true_set.intersection(false_set)
218:   unless true_and_false_sections.empty?
219:     raise RuntimeError, "Should not have the same AtomicSection(s) in " +
220:       "both the true and false branch of selection:\n" +
221:       true_and_false_sections.to_a.map(&:to_s).join(', ')
222:   end
223: end
224:

```

```

225:     def content_to_atomic_sections(content, atomic_sections)
226:       content.collect do |node|
227:         atomic_sections.select do |section|
228:           section.is_a?(AtomicSection) && section.include?(node)
229:         end.first
230:       end.compact.uniq
231:     end
232: end
233: end
234:

```

2.30 shared_children_methods.rb

```

01: module SharedChildrenMethods
02:   def delete_children_in_range(start_index, end_index)
03:     should_delete_child = lambda do |el|
04:       if el.index >= start_index && el.index <= end_index
05:         #puts "Deleting element #{el}"
06:         true
07:       else
08:         false
09:       end
10:     end
11:     if respond_to?(:atomic_sections) && !self.atomic_sections.nil?
12:       self.atomic_sections.delete_if(&should_delete_child)
13:     end
14:     (@content || []).delete_if(&should_delete_child)
15:     (@branch_content || []).delete_if do |arr|
16:       arr.delete_if(&should_delete_child)
17:       arr.empty?
18:     end
19:     #puts "Now there are #{@content.length} children"
20:     if respond_to?(:parent) && !self.parent.nil? && self.parent.respond_to?(:delete_children_in_range)
21:       self.parent.delete_children_in_range(start_index, end_index)
22:     end
23:     #puts "NOW PARENT IS #{to_s}"
24:     #puts "-----\n\n"
25:   end
26:
27:   def remove_duplicate_children
28:     return if @content.nil?
29:     @content.each do |child|
30:       #puts "Looking at child #{child.class.name}: #{child.range}"
31:       has_content = child.respond_to?(:content) && !child.content.nil? && !child.content.empty?
32:       has_close = child.respond_to?(:close) && !child.close.nil?
33:       if has_content && has_close
34:         #puts "Deleting range #{child.content.first.index}..#{child.close.index}"
35:         delete_children_in_range(child.content.first.index, child.close.index)
36:       end
37:       if child.respond_to?(:remove_duplicate_children)

```

```

38:     child.remove_duplicate_children()
39:   end
40: end
41: end
42: end
43:

```

2.31 shared_erb_methods.rb

```

1: module SharedERBMethods
2:   def ==(other)
3:     super(other) && prop_eql?(other, :ruby_code)
4:   end
5:
6:   def hash
7:     prop_hash(:ruby_code)
8:   end
9: end
10:

```

2.32 shared_html_tag_methods.rb

```

1: module SharedHTMLTagMethods
2:   def opposite_type_same_name?(opp_type, other)
3:     !other.nil? && other.is_a?(opp_type) && name == other.name
4:   end
5:
6:   def ruby_code
7:     'puts "' + text_value.gsub(/"/, "\\\"") + "'"
8:   end
9: end
10:

```

2.33 shared_methods.rb

```

01: module SharedMethods
02:   def eql?(other)
03:     return false unless other.is_a?(self.class)
04:     self == other
05:   end
06:
07:   def index_eql?(other)
08:     return false if other.nil?
09:     @index.nil? && other.index.nil? || @index == other.index
10:   end
11:
12:   def prop_eql?(other, *property_names)
13:     property_names.each do |prop_name|
14:       return false unless self.send(prop_name) == other.send(prop_name)
15:     end
16:     true

```

```

17: end
18:
19: def prop_hash(*property_names)
20:   hash_code = 0
21:   property_names << :index unless property_names.include? :index
22:   property_names.each do |prop_name|
23:     prop_value = self.send(prop_name)
24:     hash_code = hash_code ^ prop_value.hash unless prop_value.nil?
25:   end
26:   hash_code
27: end
28:
29: def to_s_with_prefix(indent_level=0, suffix='', prefix='')
30:   close_str = if !respond_to?(:close) || @close.nil?
31:     ''
32:   elsif @close.respond_to?(:range)
33:     sprintf("-%d", @close.range.to_a.last)
34:   else
35:     sprintf("-%d", @close.index)
36:   end
37:   sprintf("%s%d%s: %s", prefix * indent_level, @index, close_str, suffix)
38: end
39: end
40:

```

2.34 shared_open_tag_methods.rb

```

01: module SharedOpenTagMethods
02:   def close_str(indent_level=0)
03:     @close.nil? ? '' : @close.to_s(indent_level)
04:   end
05:
06:   def content_str(indent_level=0)
07:     if @content.nil?
08:       ''
09:     else
10:       @content.collect do |el|
11:         el.to_s(indent_level)
12:       end.join("\n") + "\n"
13:     end + close_str(indent_level)
14:   end
15: end
16:

```

2.35 shared_sexp_methods.rb

```

001: require 'pp'
002:
003: module SharedSexpMethods
004:   ITERATION_METHODS = [:each, :each_with_index, :each_cons, :each_entry,
005:     :each_slice, :each_with_object, :upto, :downto, :times].freeze

```

```

006: URL_METHODS = [:url_for].freeze
007: attr_accessor :sexp, :has_been_split
008:
009: module ClassMethods
010:   # If the given Sexp contains a call to the given method name, the Sexp
011:   # representing the arguments passed in that method call will be returned.
012:   # Otherwise, nil is returned.
013:   def get_sexp_for_call_args(sexp, method_name)
014:     unless method_name.is_a?(Symbol)
015:       raise ArgumentError, "method_name must be a Symbol"
016:     end
017:     return nil if sexp.nil? || method_name.nil? || !sexp.is_a?(Enumerable)
018:     if :call == sexp.first && (!sexp[1].nil? && method_name == sexp[1][2] || method_name ==
sexp[2])
019:       args = sexp[3]
020:       #puts "Found args:"
021:       #pp args
022:       #puts "
023:       return nil if args.nil?
024:       return nil if :arglist != args[0]
025:       args[1...args.length]
026:     elsif sexp.is_a?(Sexp)
027:       get_sexp_for_call_args(sexp[1], method_name)
028:     else
029:       nil
030:     end
031:   end
032:
033:   def sexp_include_call?(sexp, method_name)
034:     # e.g., sexp =
035:     # s(:iter,
036:     #   s(:call, s(:ivar, :@names), :each, s(:arglist)),
037:     #   s(:lasgn, :blah),
038:     #   s(:call, nil, :puts, s(:arglist, s(:lvar, :blah))))
039:     # Another sexp example:
040:     # s(:call, nil, :render, s(:arglist,
041:     #   s(:hash, s(:lit, :partial), s(:str, "top_list"),
042:     #   s(:lit, :collection), s(:ivar, :@wins),
043:     #   s(:lit, :as), s(:lit, :outcome))))
044:     unless method_name.is_a?(Symbol)
045:       raise ArgumentError, "method_name must be a Symbol"
046:     end
047:     return false if sexp.nil? || method_name.nil? || !sexp.is_a?(Enumerable)
048:     if :call == sexp.first && (!sexp[1].nil? && method_name == sexp[1][2] || method_name ==
sexp[2])
049:       true
050:     else
051:       sexp_include_call?(sexp[1], method_name)
052:     end
053:   end

```

```

054:
055: def get_sexp_hash_value(sexp, key)
056:   return nil if sexp.nil? || !sexp.is_a?(Enumerable) || :hash != sexp.first
057:   if key.nil?
058:     raise ArgumentError, "Expected non-nil hash key to look up in Sexp"
059:   end
060:   key = key.to_s.downcase
061:   key_value_pairs = sexp[1...sexp.length]
062:   # Sample:
063:   # s(:hash,
064:   # s(:lit, :action),
065:   # s(:str, "try_login"),
066:   # s(:lit, :method),
067:   # s(:lit, :post))
068:   # Keys are on even indices, values on the following odd index
069:   num_pairs = key_value_pairs.length
070:   key_value_pairs.each_with_index do |key_or_value, i|
071:     if i.even? && key_or_value[1].to_s.downcase == key && i+1 < num_pairs
072:       value_sexp = key_value_pairs[i+1]
073:       if value_sexp.length == 2
074:         return value_sexp[1]
075:       else
076:         return value_sexp
077:       end
078:     end
079:   end
080:   nil
081: end
082:
083: def sexp_outer_call?(sexp, method_name)
084:   unless method_name.is_a?(Symbol)
085:     raise ArgumentError, "method_name must be a Symbol"
086:   end
087:   return false if sexp.nil? || method_name.nil? || !sexp.is_a?(Enumerable)
088:   :call == sexp.first && (!sexp[1].nil? && method_name == sexp[1][2] || method_name ==
sexp[2])
089: end
090:
091: def sexp_outer_keyword?(sexp, keyword)
092:   unless keyword.is_a?(Symbol)
093:     raise ArgumentError, "keyword must be a Symbol"
094:   end
095:   return false if sexp.nil? || keyword.nil? || !sexp.is_a?(Enumerable)
096:   keyword == sexp.first
097: end
098: end
099:
100: # Tries to extract a URL from the given Sexp. Looks for calls to the Rails
101: # method url_for(), as well as plain string URLs, as well as
102: # controller/action hashes.

```

```

103: def get_target_page_from_sexp(sexp_args, source=nil)
104:   if source.nil? || !source.respond_to?(:controller)
105:     src_controller = nil
106:   else
107:     src_controller = source.controller
108:   end
109:   sexp_args.each do |sexp|
110:     if sexp.is_a?(Enumerable) && !sexp.empty?
111:       if :ivar == sexp[0] && sexp.length >= 2
112:         var_name = sexp[1].to_s.gsub(/@/, '')
113:         return RailsURL.new(src_controller, nil, var_name)
114:       end
115:     end
116:     controller = self.class.get_sexp_hash_value(sexp, :controller) || src_controller
117:     action = self.class.get_sexp_hash_value(sexp, :action)
118:     unless action.nil?
119:       return RailsURL.new(controller, action, nil)
120:     end
121:     url = self.class.get_sexp_hash_value(sexp, :url)
122:     unless url.nil?
123:       return RailsURL.new(nil, nil, url) if url.is_a?(String)
124:       if url.is_a?(Sexp)
125:         URL_METHODS.each do |url_method|
126:           url_args = self.class.get_sexp_for_call_args(url, url_method)
127:           unless url_args.nil?
128:             return get_target_page_from_sexp(url_args, source)
129:           end
130:         end
131:       end
132:     end
133:   end
134:   nil
135: end
136:
137: def sexp_include_call?(sexp, method_name)
138:   # e.g., sexp =
139:   # s(:iter,
140:   #   s(:call, s(:ivar, :@names), :each, s(:arglist)),
141:   #   s(:lasgn, :blah),
142:   #   s(:call, nil, :puts, s(:arglist, s(:lvar, :blah))))
143:   # Another sexp example:
144:   # s(:call, nil, :render, s(:arglist,
145:   #   s(:hash, s(:lit, :partial), s(:str, "top_list")),
146:   #   s(:lit, :collection), s(:ivar, :@wins),
147:   #   s(:lit, :as), s(:lit, :outcome))))
148:   unless method_name.is_a?(Symbol)
149:     raise ArgumentError, "method_name must be a Symbol"
150:   end
151:   return false if sexp.nil? || method_name.nil? || !sexp.is_a?(Enumerable)
152:   if :call == sexp.first && (!sexp[1].nil? && method_name == sexp[1][2] || method_name ==

```

```

sexp[2])
153:     true
154:   else
155:     sexp_include_call?(sexp[1], method_name)
156:   end
157: end
158:
159: def set_sexp
160:   #puts "Setting sexp for " + to_s
161:   return unless @sexp.nil?
162:   parser = RubyParser.new
163:   begin
164:     # Call dup() otherwise ERBTag Ruby comments end up with multiple pound
165:     # signs at the beginning (!)
166:     @sexp = parser.parse(ruby_code().dup())
167:   rescue Racc::ParseError
168:     @sexp = :invalid_ruby
169:   rescue SyntaxError
170:     # Can occur when lines are split on ; and this happens in the
171:     # middle of a string
172:     @sexp = :invalid_ruby
173:   end
174:   #pp @sexp
175:   #puts "
176: end
177:
178: # p -> p1 | p2 (conditionals)
179: def selection?
180:   set_sexp() if @sexp.nil?
181:   return false if :invalid_ruby == @sexp
182:   [:if, :case, :when].each do |keyword|
183:     return true if self.class.sexp_outer_keyword?(@sexp, keyword)
184:   end
185:   false
186: end
187:
188: def split_branch
189:   return unless @has_been_split.nil?
190:   @has_been_split = true
191:   # Return here when, for example, there's an if statement within an ERBOutputTag,
192:   # e.g., <%= (user.id == session[:user][:id]) ? 'you' : user.email %>
193:   return if !selection? || !respond_to?(:branch_content=) || :invalid_ruby == @sexp
194:   # Expect non-ERBTag content to be contained in AtomicSections, so
195:   # only get ERBTags who might have nested AtomicSections within them,
196:   # as opposed to HTMLOpenTags and whatnot that would be duplicated
197:   # within AtomicSections we've already got
198:   erb_content = (@content || []).select do |child|
199:     child.set_sexp() if child.sexp.nil?
200:     child.is_a?(ERBGrammar::ERBTag) && child.respond_to?(:parent) && self == child.parent
201:   end

```



```

202:   # Split branches on contents first, in case there are nested case-when's
203:   erb_content.map(&:split_branch)
204:
205:   @branch_content ||= []
206:   atomic_sections = @atomic_sections || []
207:   if erb_content.empty?
208:     @branch_content << atomic_sections
209:     return
210:   end
211:
212:   # Find all invalid Ruby, and assume it's the pivot points
213:   pivots = erb_content.select do |child|
214:     :invalid_ruby == child.sexp
215:   end.sort { |a, b| a.index <=> b.index }
216:   if pivots.empty?
217:     add_branch_content(atomic_sections, erb_content)
218:     return
219:   end
220:   prev_pivot = pivots[0]
221:   prev_index = prev_pivot.index
222:
223:   select_first_branch = lambda do |child|
224:     child.index > @index && child.index < prev_index
225:   end
226:   add_branch_content(atomic_sections.select(&select_first_branch),
227:     erb_content.select(&select_first_branch))
228:
229:   if :invalid_ruby == pivots[0].sexp
230:     prev_pivot = pivots[0]
231:   else
232:     prev_pivot = self
233:   end
234:   prev_index = prev_pivot.index
235:   pivots.each do |condition_pivot|
236:     next if prev_pivot == condition_pivot
237:     cond_pivot_index = condition_pivot.index
238:     is_branch_child = lambda do |child|
239:       child.index > prev_index && child.index < cond_pivot_index
240:     end
241:     branch_erb = erb_content.select(&is_branch_child)
242:     branch_sections = atomic_sections.select(&is_branch_child)
243:     next if branch_erb.empty? && branch_sections.empty?
244:     if prev_pivot != self
245:       copy_branch_content(branch_sections, branch_erb, prev_pivot, cond_pivot_index)
246:       @branch_content << [prev_pivot]
247:     end
248:     prev_pivot = condition_pivot
249:     prev_index = prev_pivot.index
250:   end
251:

```

```

252:   if !@close.nil?
253:     close_index = get_close_index(prev_pivot) || @close.index
254:     select_last_branch = lambda do |child|
255:       child.index > prev_index && child.index < close_index
256:     end
257:     branch_sections = atomic_sections.select(&select_last_branch)
258:     branch_erb = erb_content.select(&select_last_branch)
259:     copy_branch_content(branch_sections, branch_erb, prev_pivot, close_index)
260:     add_branch_content(branch_sections, branch_erb)
261:   end
262: end
263:
264: def get_close_index(prev_pivot)
265:   if prev_pivot.respond_to?(:parent) && !prev_pivot.parent.nil?
266:     prev_parent = prev_pivot.parent
267:     if prev_parent.respond_to?(:close) && !prev_parent.close.nil?
268:       prev_parent.close.index
269:     else
270:       nil
271:     end
272:   else
273:     nil
274:   end
275: end
276:
277: # p -> p1* (loops)
278: def iteration?
279:   set_sexp() if @sexp.nil?
280:   if :invalid_ruby == @sexp
281: #     puts "Invalid ruby for:\n" + to_s
282:     return false
283:   end
284:   # For cases like the following sexp:
285:   # s(:iter,
286:   #   s(:call,
287:   #     s(:call, s(:ivar, :@game), :get_sorted_scores, s(:arglist, s(:true))),
288:   #     :each,
289:   #     s(:arglist)),
290:   #   s(:lasgn, :score))
291:   if self.class.sexp_outer_keyword?(@sexp, :iter) &&
292:     sexp_calls_enumerable_method?(@sexp[1])
293:     #puts "Sexp has a call to :each in iterator--iteration!\n"
294:     return true
295:   end
296:   [:while, :for, :until].each do |keyword|
297: #     puts "Looking for key word '" + keyword.to_s + "' in "
298: #     pp @sexp
299:     if self.class.sexp_outer_keyword?(@sexp, keyword)
300: #       puts "Found it!\n"
301:       return true

```

```

302:     end
303: end
304: if sexp_calls_enumerable_method?(@sexp)
305: #   puts "Sexp has a call to :each--iteration!\n"
306:     return true
307: end
308: false
309: end
310:
311: # p -> p1{p2} (file inclusion, function calls in p1)
312: def aggregation?
313:     set_sexp() if @sexp.nil?
314:     return false if :invalid_ruby == @sexp
315:     # TODO: go out and fetch the component expression for the thing
316:     # being rendered, if possible?
317:     return true if self.class.sexp_outer_call?(@sexp, :render)
318:     false
319: end
320:
321: private
322: def add_branch_content(sections, erb)
323:     branch_content = sections + erb
324:     return if branch_content.empty?
325:     branch_content.sort! { |a, b| self.class.section_and_node_sort(a, b) }
326:     @branch_content << branch_content
327: end
328:
329: def copy_branch_content(sections, erb, parent, final_index)
330:     copy_atomic_sections(sections, parent)
331:     copy_content(erb, parent) unless erb.empty?
332:     branch_content = erb + sections
333:     return if branch_content.empty?
334:     branch_content.sort! { |a, b| self.class.section_and_node_sort(a, b) }
335:     delete_children_in_range(branch_content.first.index, final_index-1)
336: end
337:
338: def copy_atomic_sections(sections, parent)
339:     sections.each do |section|
340:         parent.add_atomic_section(section)
341:     end
342: end
343:
344: def copy_content(new_content, parent)
345:     return if parent.nil?
346:     (new_content || []).each do |child|
347:         if child.index <= parent.index
348:             raise ArgumentError, "Cannot set element #{child} to be child of #{parent}--index
is too low"
349:         end
350:     end

```

```

351:     parent.content = new_content
352: end
353:
354: def sexp_calls_enumerable_method?(sexp)
355:   ITERATION_METHODS.each do |method_name|
356:     return true if self.class.sexp_outer_call?(sexp, method_name)
357:   end
358:   false
359: end
360:
361: def lines_consecutive_in_sexp?(needle, haystack)
362:   return false if haystack.nil?
363:   found_each_line_consecutively = true
364:   index = 0
365:   prev_index = -1
366:   num_lines = needle.length
367:   while index < num_lines && found_each_line_consecutively && !prev_index.nil?
368:     line = needle[index]
369:     #puts "Previous matching index: #{prev_index}"
370:     #puts "Looking for line ##{index} #{line}"
371:     matching_index = haystack.index { |s| line == s }
372:     #puts "Found at index ##{matching_index || 'nil'}"
373:     found_each_line_consecutively = !matching_index.nil? && (-1 == prev_index || matching_index-
1 == prev_index)
374:     #puts "Found each line consecutively: #{found_each_line_consecutively}"
375:     prev_index = matching_index
376:     index += 1
377:   end
378:   found_each_line_consecutively
379: end
380:
381: def contained_or_equal?(needle, haystack)
382:   return false if haystack.nil? || !haystack.is_a?(Sexp)
383:   if haystack.include?(needle) || haystack == needle
384:     return true
385:   end
386:   haystack.each do |haystack_child|
387:     if contained_or_equal?(needle, haystack_child)
388:       return true
389:     end
390:   end
391:   false
392: end
393:
394: def replace_lvars(sexp_arr, so_far=[])
395:   return so_far if sexp_arr.nil?
396:   unless sexp_arr.is_a?(Array)
397:     raise ArgumentError, "Expected Array, got #{sexp_arr.class.name}"
398:   end
399:   first_item = sexp_arr[0]

```

```

400:     if :lvar == first_item
401:       name = sexp_arr[1]
402:       replacement = [:call, nil, name, [:arglist]]
403:       so_far += replacement
404:     else
405:       if first_item.is_a?(Array)
406:         replacement = replace_lvars(first_item, [])
407:         so_far << replacement unless replacement.nil?
408:       else
409:         so_far << first_item
410:       end
411:       next_part = sexp_arr[1...sexp_arr.length]
412:       unless next_part.empty?
413:         replace_lvars(next_part, so_far)
414:       end
415:     end
416:     so_far
417:   end
418:
419:   def sexp_contains_sexp?(needle, haystack)
420:     return false if haystack.nil? || needle.nil? || :invalid_ruby == needle
421:     unless needle.is_a?(Sexp)
422:       raise ArgumentError, "Expected parameter to be of type Sexp, got " + needle.class.name
423:     end
424:     unless haystack.is_a?(Sexp)
425:       raise ArgumentError, "Expected parameter to be of type Sexp, got " + haystack.class.name
426:     end
427:     set_sexp() if @sexp.nil?
428:     if !selection?
429:       puts "Not a selection"
430:       return false
431:     end
432:     #puts "Looking for"
433:     #pp needle
434:     #puts "In"
435:     #pp haystack
436:     #puts "
437:     return true if contained_or_equal?(needle, haystack)
438:     #if self.class.sexp_outer_keyword?(haystack, :block)
439:     # haystack = haystack[1...haystack.length]
440:     #end
441:     if self.class.sexp_outer_keyword?(needle, :block)
442:       needle = needle[1...needle.length]
443:     end
444:     return true if contained_or_equal?(needle, haystack)
445:     if haystack.to_a.flatten.include?(:lvar)
446:       # Example:
447:       # haystack =
448:       #s(:call,
449:       # nil,

```

```

450:      # :distance_of_time_in_words_to_now,
451:      # s(:arglist, s(:call, s(:lvar, :l), :updated_at, s(:arglist))))
452:      #
453:      # needle =
454:      #s(:call,
455:      # nil,
456:      # :distance_of_time_in_words_to_now,
457:      # s(:arglist,
458:      # s(:call, s(:call, nil, :l, s(:arglist)), :updated_at, s(:arglist))))
459:      new_needle = needle.to_a
460:      new_haystack = replace_lvars(haystack.to_a)
461:      if self.class.sexp_outer_keyword?(new_haystack, :block)
462:        new_haystack = new_haystack[1...new_haystack.length]
463:      end
464:      return true if lines_consecutive_in_sexp?(new_needle, new_haystack)
465:      return true if contained_or_equal?(new_needle, new_haystack)
466:    else
467:      return true if lines_consecutive_in_sexp?(needle, haystack)
468:    end
469:    false
470:  end
471: end
472:

```

2.36 shared_sexp_parsing.rb

```

01: module SharedSexpParsing
02:   attr_reader :parsed_sexp
03:
04:   def sexp
05:     return @parsed_sexp unless @parsed_sexp.nil?
06:     parser = RubyParser.new
07:     begin
08:       @parsed_sexp = parser.parse(ruby_code)
09:     rescue Racc::ParseError
10:       @parsed_sexp = :invalid_ruby
11:     end
12:     @parsed_sexp
13:   end
14: end
15:

```

2.37 shared_transition_methods.rb

```

01: module ERBGrammar
02:   module SharedTransitionMethods
03:     attr_reader :transitions
04:
05:     def identify_transitions(source_rails_url, root_url)
06:       if source_rails_url.relative?
07:         source_rails_url = RailsURL.new(source_rails_url.controller,

```

```

08:             source_rails_url.action,
09:             source_rails_url.raw_url,
10:             root_url)
11:     end
12:     @transitions = get_local_transitions(source_rails_url)
13:     children = []
14:     children += @content || [] if respond_to?(:content)
15:     children += @atomic_sections || [] if respond_to?(:atomic_sections)
16:     children.each do |child|
17:         #puts "Identifying transitions for child: " + child.to_s
18:         if child.respond_to?(:identify_transitions)
19:             child.identify_transitions(source_rails_url, root_url)
20:         end
21:     end
22: end
23: end
24: end
25:

```

2.38 single_file_generator.rb

```

01: #!/usr/bin/env ruby
02: root_dir = File.expand_path(File.dirname(__FILE__))
03: require File.join(root_dir, 'parser.rb')
04: require 'optparse'
05: require 'pp'
06: require File.join(root_dir, 'component_interaction_model.rb')
07:
08: options = {}
09: optparse = OptionParser.new do |opts|
10:   opts.banner = sprintf("Usage: %s [options]", $0)
11:
12:   options[:debug] = false
13:   $DEBUG = false
14:   opts.on('-d', '--debug', 'Turn debug messages on') do
15:     options[:debug] = true
16:     $DEBUG = true
17:   end
18: end
19:
20: # Parse command-line parameters and remove all flag parameters from ARGV
21: optparse.parse!
22:
23: unless ARGV.length == 2
24:   printf("Usage: %s [-d] path_to_erb_file root_url_of_site\n", $0)
25:   exit
26: end
27:
28: path = ARGV.shift
29: begin

```

```

30: root_url = URI.parse(ARGV.shift)
31: rescue URI::InvalidURIError => err
32:   printf("ERROR: could not parse given root URI: %s", err)
33:   exit
34: end
35: erb = IO.readlines(path).join
36: ast = Parser.new.parse(erb, path, root_url, options[:debug])
37: pp ast
38: expr = ast.component_expression()
39: sections = ast.get_atomic_sections()
40: trans = ast.get_transitions()
41: cim = ComponentInteractionModel.new(root_url, path, expr, sections, trans)
42: puts cim.to_s + "\n"
43:

```

2.39 syntax_node.rb

```

001: module ERBGrammar
002:   class Treetop::Runtime::SyntaxNode
003:     include Enumerable
004:     include SharedMethods
005:     PlainHTMLTypes = [HTMLDirective, HTMLOpenTag, HTMLCloseTag, Whitespace, Text,
HTMLDoctype, HTMLQuotedValue, HTMLSelfClosingTag, HTMLTagAttribute].freeze
006:     ERBOutputTypes = [ERBOutputTag, ERBYield].freeze
007:     BrowserOutputTypes = (PlainHTMLTypes + ERBOutputTypes).freeze
008:     RubyCodeTypes = ([ERBTag] + ERBOutputTypes).freeze
009:     attr_accessor :index
010:     alias_method :old_to_s, :to_s
011:
012:     def [](obj)
013:       if obj.is_a?(Fixnum)
014:         each_with_index do |el, i|
015:           return el if i == obj
016:         end
017:       end
018:     end
019:
020:     def ==(other)
021:       # Necessary to check other.class to prevent comparing a SyntaxNode with a
022:       # TrueClass instance, for example
023:       return false unless other.is_a?(self.class) &&
024:         length == other.length &&
025:         index.eql?(other)
026:       if nonterminal?
027:         elements.each_with_index do |el, i|
028:           return false unless el == other[i]
029:         end
030:       end
031:       true
032:     end

```



```

033:
034:   def each
035:     if nonterminal?
036:       elements.each { |el| yield el }
037:     end
038:   end
039:
040:   def browser_output?
041:     BrowserOutputTypes.include?(self.class)
042:   end
043:
044:   def length
045:     nonterminal? ? elements.length : 0
046:   end
047:
048:   def range
049:     start_index = @index
050:     end_index = (!respond_to?(:close) || @close.nil?) ? start_index : @close.index
051:     (start_index..end_index)
052:   end
053:
054:   def same_atomic_section?(other)
055:     return false if other.nil? || @index.nil? || other.index.nil?
056:     index_diff = (@index - other.index).abs
057:     return false if 1 != index_diff
058:
059:     # If both nodes are just HTML, they can be part of the same atomic
060:     # section
061:     is_plain_html = PlainHTMLTypes.include?(self.class)
062:     other_is_plain_html = PlainHTMLTypes.include?(other.class)
063:     return true if is_plain_html && other_is_plain_html
064:
065:     # If one node is an ERBTag and the other is not, they should not
066:     # be in the same atomic section--ERBTags split apart atomic sections
067:     is_erb = self.is_a?(ERBTag)
068:     other_is_erb = other.is_a?(ERBTag)
069:     return false if !is_erb && other_is_erb || is_erb && !other_is_erb
070:
071:     class1_is_output = self.is_a?(ERBOutputTag)
072:     class2_is_output = other.is_a?(ERBOutputTag)
073:     if class1_is_output && class2_is_output
074:       # Two ERBOutputTags
075:       class1_is_render = ERBOutputTag.sexp_include_call?(self.sexp, :render)
076:       class2_is_render = ERBOutputTag.sexp_include_call?(other.sexp, :render)
077:       if !class1_is_render && !class2_is_render
078:         return true
079:       else
080:         # One ERBOutputTag is a render() and the other is not, or they are
081:         # both render() calls--thus they are two separate atomic sections,
082:         # using aggregation

```

```

083:     return false
084:   end
085:   elsif class1_is_output && ERBOutputTag.sexp_include_call?(self.sexp, :render)
086:     return false
087:   elsif class2_is_output && ERBOutputTag.sexp_include_call?(other.sexp, :render)
088:     return false
089:   end
090:   true
091: end
092:
093: # Thanks to https://github.com/aarongough/koi-reference-parser/blob/
094: # development/lib/parser/syntax_node_extensions.rb
095: def to_h
096:   hash = {}
097:   hash[:offset] = interval.first
098:   hash[:text_value] = text_value
099:   hash[:name] = self.class.name.split("::").last
100:   if elements.nil?
101:     hash[:elements] = nil
102:   else
103:     hash[:elements] = elements.map do |element|
104:       element.to_h
105:     end
106:   end
107:   hash
108: end
109:
110: def new_to_s(indent_level=0)
111:   to_s_with_prefix(indent_level, old_to_s)
112: end
113:
114: alias_method :to_s, :new_to_s
115: end
116: end
117:

```

2.40 syntax_node_test.rb

```

01: base_path = File.expand_path(File.dirname(__FILE__))
02: require File.join(base_path, '..', 'parser.rb')
03: require File.join(base_path, 'test_helper.rb')
04: require File.join(base_path, '..', 'nodes', 'erb_node_extensions.rb')
05:
06: class SyntaxNodeTest < Test::Unit::TestCase
07:   include ERBGrammar
08:
09:   def test_same_atomic_section?
10:     nodes = get_test_nodes()
11:     # Impose our own order on the elements in the document:
12:     nodes[:html_tags][0].index = 0

```

```

13: nodes[:erb_output_tags][0].index = 1
14: nodes[:html_tags][1].index = 2
15: nodes[:html_tags][2].index = 3
16: nodes[:erb_tags][0].index = 4
17: nodes[:erb_output_tags][1].index = 5
18: nodes[:erb_tags][1].index = 6
19:
20: assert nodes[:html_tags][0].same_atomic_section?(nodes[:erb_output_tags][0])
21: assert nodes[:erb_output_tags][0].same_atomic_section?(nodes[:html_tags][1])
22: assert nodes[:html_tags][1].same_atomic_section?(nodes[:html_tags][2])
23: assert !nodes[:html_tags][2].same_atomic_section?(nodes[:erb_tags][0]), "Expected following to
not be in same atomic section:\n(#{nodes[:html_tags][2].class.name}) " + nodes[:html_tags][2].to_s
+ "\n\n(#{nodes[:erb_tags][0].class.name}) " + nodes[:erb_tags][0].to_s
24: assert !nodes[:erb_tags][0].same_atomic_section?(nodes[:erb_output_tags][1])
25: assert !nodes[:erb_output_tags][1].same_atomic_section?(nodes[:erb_tags][1])
26: end
27:
28: private
29: def get_test_nodes
30:   doc = Parser.new.parse(fixture('login_index.html'), 'login_index.html.erb', URI.parse('/'))
31:   form_tag = doc[0]
32:   assert_equal ERBTag, form_tag.class
33:   section = form_tag.atomic_sections[0]
34:   html_tag_1 = section.content[0]
35:   html_tag_2 = section.content[1]
36:   html_tag_3 = section.content[3]
37:   assert_equal HTMLOpenTag, html_tag_1.class, "Tree:\n" + doc.to_s
38:   assert_equal HTMLOpenTag, html_tag_2.class
39:   assert_equal HTMLCloseTag, html_tag_3.class
40:   output_tag_1 = section.content[9]
41:   output_tag_2 = section.content[15]
42:   assert_equal ERBOutputTag, output_tag_1.class, output_tag_1.to_s
43:   assert_equal ERBOutputTag, output_tag_2.class, output_tag_2.to_s
44:   erb_tag_1 = form_tag
45:   erb_tag_2 = form_tag.close
46:   assert_equal ERBTag, erb_tag_2.class
47:   assert !erb_tag_1.browser_output?
48:   assert !erb_tag_2.browser_output?
49:   {:html_tags => [html_tag_1, html_tag_2, html_tag_3],
50:    :erb_output_tags => [output_tag_1, output_tag_2],
51:    :erb_tags => [erb_tag_1, erb_tag_2]}
52: end
53: end
54:

```

2.41 test_helper.rb

```

01: require 'test/unit'
02:
03: class Test::Unit::TestCase

```

```

04: BasePath = File.expand_path(File.dirname(__FILE__)).freeze
05:
06: # Returns contents of ERB file with the given prefix
07: def fixture(file_name_prefix)
08:   path = File.join(BasePath, 'fixtures', sprintf("%s.erb", file_name_prefix))
09:   IO.readlines(path).join
10: end
11: end
12:

```

2.42 text.rb

```

01: module ERBGrammar
02:   class Text < Treetop::Runtime::SyntaxNode
03:     include SharedSexpParsing
04:     include SharedSexpMethods
05:     extend SharedSexpMethods::ClassMethods
06:
07:     def ==(other)
08:       super(other) && prop.eql?(other, :text_value)
09:     end
10:
11:     def hash
12:       prop_hash(:text_value)
13:     end
14:
15:     # TODO: remove duplication between this and SharedHTMLTagMethods
16:     def ruby_code
17:       'puts "' + text_value.gsub(/"/, "\\\"") + "'"
18:     end
19:
20:     def to_s(indent_level=0)
21:       stripped = text_value.strip
22:       to_s_with_prefix(
23:         indent_level,
24:         if stripped.empty?
25:           ''
26:         else
27:           stripped.gsub(/\'/, "\\\'")
28:         end
29:       )
30:     end
31:   end
32: end
33:

```

2.43 transition.rb

```

01: require 'uri'
02:
03: class Transition

```

```

04: attr_reader :source, :sink, :code
05:
06: def initialize(src, snk, c)
07:   if src.nil?
08:     raise ArgumentError, "Given source of transition cannot be nil"
09:   end
10:   if src.is_a?(String)
11:     @source = URI.parse(src)
12:   else
13:     @source = src
14:   end
15:   if snk.nil? || !snk.is_a?(RailsURL)
16:     raise ArgumentError, "Given sink of transition cannot be nil, and must be a RailsURL
(got #{snk.class.name})"
17:   end
18:   @sink = snk
19:   if c.nil? || !c.is_a?(String) || c.blank?
20:     raise ArgumentError, "Given transition code cannot be blank or nil, and must be a
String (got #{c.class.name})"
21:   end
22:   @code = c
23: end
24:
25: def inspect
26:   to_s
27: end
28:
29: def to_s(prefix='')
30:   tab = ' '
31:   sprintf("%s<%s> --> <%s>\n%sUnderlying code:\n%s%s%s",
32:     tab, prefix, @source, @sink, prefix, tab,
33:     prefix, tab, tab, (@code || '').strip)
34: end
35: end
36:

```

2.44 whitespace.rb

```

1: module ERBGrammar
2:   class Whitespace < Treetop::Runtime::SyntaxNode
3:     def to_s(indent_level=0)
4:       to_s_with_prefix(indent_level, "Whitespace")
5:     end
6:   end
7: end
8:

```

3 Shared Files

3.1 html_parsing.rb

```
001: require 'uri'
002: require 'rubygems'
003: require 'nokogiri'
004: require File.join(File.expand_path(File.dirname(__FILE__)), 'link_text.rb')
005:
006: module SharedHtmlParsing
007:   module ClassMethods
008:     TransitionURITypes = [URI::HTTP, URI::FTP].freeze
009:     SubmitButtonTypes = ['submit', 'image'].freeze
010:
011:     def get_uri_for_host(str, host_uri)
012:       unless str.is_a?(String)
013:         raise ArgumentError, "Expected URI string, got #{src.class.name}"
014:       end
015:       unless host_uri.is_a?(URI)
016:         raise ArgumentError, "Expected host URI, got #{host_uri.class.name}"
017:       end
018:       if host_uri.relative?
019:         raise ArgumentError, "Expected absolute URI for host URI, got relative URI #{host_uri}"
020:       end
021:       return nil if str.length < 1
022:       rel_uri = parse_uri_forgivingly(str)
023:       if rel_uri.nil?
024:         if str.include?('##')
025:           # Try to clean up badly formed URIs like http://example.com/#comments#add_comment
026:           pound_index = str.index('##')
027:           str2 = str[0...pound_index]
028:           rel_uri = parse_uri_forgivingly(str2)
029:           if rel_uri.nil?
030:             return nil
031:           end
032:         else
033:           return nil
034:         end
035:       end
036:       absolutize_uri(rel_uri, host_uri)
037:     end
038:
039:     def parse_uri_forgivingly(str)
040:       begin
041:         URI.parse(str)
042:       rescue URI::InvalidURIError
043:         nil
044:       end
045:     end
046:
```

```

047:     def get_form_uris(root_uri, doc)
048:       get_form_uris_with_text(root_uri, doc).map(&:uri)
049:     end
050:
051:   def get_form_uris_with_text(root_uri, doc)
052:     if root_uri.nil? || !root_uri.is_a?(URI)
053:       raise ArgumentError, "Expected URI, got #{root_uri.class.name}"
054:     end
055:     target_host = root_uri.host
056:     extract_uris_on_host(
057:       doc.css('form').select do |form|
058:         if form['action'].nil?
059:           false
060:         else
061:           !get_submit_buttons(form.css('input')).empty?
062:         end
063:       end).collect do |form|
064:         uri = get_uri_for_host(form['action'], root_uri)
065:         if include_uri?(uri)
066:           desc = get_submit_buttons(form.css('input')).join(', ')
067:           LinkText.new(uri, desc)
068:         else
069:           nil
070:         end
071:       end,
072:       target_host
073:     ).uniq
074:   end
075:
076:   def get_link_uris(root_uri, doc)
077:     get_link_uris_with_text(root_uri, doc).map(&:uri)
078:   end
079:
080:   def get_link_uris_with_text(root_uri, doc)
081:     if root_uri.nil? || !root_uri.is_a?(URI)
082:       raise ArgumentError, "Expected URI, got #{root_uri.class.name}"
083:     end
084:     target_host = root_uri.host
085:     all_uris = doc.css('a').select do |link|
086:       !link['href'].nil?
087:     end.collect do |link|
088:       uri = get_uri_for_host(link['href'], root_uri)
089:       if include_uri?(uri)
090:         LinkText.new(uri, link.children.to_s)
091:       else
092:         nil
093:       end
094:     end
095:     extract_uris_on_host(all_uris, target_host).uniq
096:   end

```

```

097:
098: private
099:   def absolutize_uri(relative_uri, root_uri)
100:     if relative_uri.nil? || !relative_uri.is_a?(URI::Generic)
101:       raise ArgumentError, "Expected a relative URI, got #{relative_uri.class.name}"
102:     end
103:     if root_uri.nil? || !root_uri.is_a?(URI::Generic)
104:       raise ArgumentError, "Expected a root URI, got #{root_uri.class.name}"
105:     end
106:     if root_uri.relative?
107:       raise ArgumentError, "Expected absolute root URI, got a relative root URI
#{root_uri}"
108:     end
109:     return relative_uri unless relative_uri.relative?
110:     rel_uri_str = relative_uri.to_s || ''
111:     slash = rel_uri_str.start_with?(' / ') ? ' / ' : '/'
112:     abs_path = sprintf("%s://%s%s%s", root_uri.scheme, root_uri.host, slash, relative_uri.to_s)
113:     parse_uri_forgivingly(abs_path)
114:   end
115:
116:   def get_submit_buttons(inputs)
117:     (inputs || []).select do |input|
118:       !input.nil? && !input['type'].nil? && SubmitButtonTypes.include?(input['type'].downcase)
119:     end.collect do |input|
120:       value = input['value']
121:       if value.nil? || value.length < 1
122:         src = input['src']
123:         id = input['id']
124:         src_id = sprintf("source %s, ID %s", src, id)
125:         sprintf("%s button - %s", input['type'], src_id)
126:       else
127:         value
128:       end
129:     end
130:   end
131:
132:   def extract_uris_on_host(link_texts, target_host)
133:     link_texts.compact.select do |link_text|
134:       uri = link_text.uri
135:       target_host == uri.host || uri.relative?
136:     end.uniq
137:   end
138:
139:   def include_uri?(uri)
140:     !uri.nil? && TransitionURITypes.include?(uri.class)
141:   end
142: end
143: end
144:

```


3.2 link_text.rb

```
01: require File.join(File.expand_path(File.dirname(__FILE__)), 'uri_extensions.rb')
02:
03: class LinkText
04:   attr_reader :uri, :uri_parts, :description
05:
06:   def initialize(u, desc)
07:     if u.nil? || !u.is_a?(URI)
08:       raise ArgumentError, "Expected URI, got #{u.class.name}"
09:     end
10:     @uri = u
11:     if desc.nil? || !desc.is_a?(String)
12:       raise ArgumentError, "Expected String of link description, got #{desc.class.name}"
13:     end
14:     @description = desc
15:     @uri_parts = @uri.get_uniq_parts()
16:   end
17:
18:   def ==(other)
19:     other.is_a?(LinkText) && @uri_parts == other.uri_parts && @description == other.description
20:   end
21:
22:   def <=>(other)
23:     @description <=> other.description
24:   end
25:
26:   def eql?(other)
27:     self == other
28:   end
29:
30:   def hash
31:     @uri.hash ^ @description.hash
32:   end
33: end
34:
```

3.3 uri_extensions.rb

```
01: require 'uri'
02:
03: class URI::FTP
04:   # E.g. ["ftp", "blah", "test/", "query=yes"] for URI ftp://blah/test/?query=yes
05:   def get_uniq_parts
06:     [scheme, host, path, query]
07:   end
08: end
09:
10: class URI::HTTP
11:   # Use scheme (e.g. http), host (e.g. google.com), and request_uri,
12:   # which includes parameters such as ?query=whew but not #comments
```

```
13: def get_uniq_parts
14:   [scheme, host, request_uri.gsub(/\//, ' ')]
15: end
16: end
17:
```