Files

cd08e6b

Go to file

- .devcontainer
- .github
- actioncable
- actionmailbox
- actionmailer
- actionpack
  - bin
  - lib
    - abstract_controller
    - action_controller
    - action_dispatch
      - http
      - journey
      - middleware
        - session
        - templates
        - actionable_exceptions.rb
        - callbacks.rb
        - cookies.rb
        - debug_exceptions.rb
        - debug_locks.rb
        - debug_view.rb
        - exception_wrapper.rb
        - executor.rb
        - flash.rb
        - host_authorization.rb
        - public_exceptions.rb
        - reloader.rb
        - remote_ip.rb
        - request_id.rb
        - server_timing.rb
        - show_exceptions.rb
        - ssl.rb
        - stack.rb
        - static.rb
      - request

rails / actionpack / lib / action_dispatch / middleware / exception_wrapper.rb

alaz and flavorjones Skip logging backtrace when exception is i... dc2d948 · 3 years ago History

Code Blame 196 lines (162 loc) · 6.05 KB Raw

```ruby
# frozen_string_literal: true

require "active_support/core_ext/module/attribute_accessors"
require "rack/utils"

module ActionDispatch
  class ExceptionWrapper
    cattr_accessor :rescue_responses, default: Hash.new(:internal_server_error).merge!(
      "ActionController::RoutingError"                    => :not_found,
      "AbstractController::ActionNotFound"                => :not_found,
      "ActionController::MethodNotAllowed"                => :method_not_allowed,
      "ActionController::UnknownHttpMethod"               => :method_not_allowed,
      "ActionController::NotImplemented"                  => :not_implemented,
      "ActionController::UnknownFormat"                   => :not_acceptable,
      "ActionDispatch::Http::MimeNegotiation::InvalidType" => :not_acceptable,
      "ActionController::MissingExactTemplate"            => :not_acceptable,
      "ActionController::InvalidAuthenticityToken"        => :unprocessable_entity,
      "ActionController::InvalidCrossOriginRequest"       => :unprocessable_entity,
      "ActionDispatch::Http::Parameters::ParseError"      => :bad_request,
      "ActionController::BadRequest"                      => :bad_request,
      "ActionController::ParameterMissing"                => :bad_request,
      "Rack::QueryParser::ParameterTypeError"             => :bad_request,
      "Rack::QueryParser::InvalidParameterError"          => :bad_request
    )

    cattr_accessor :rescue_templates, default: Hash.new("diagnostics").merge!(
      "ActionView::MissingTemplate"            => "missing_template",
      "ActionController::RoutingError"         => "routing_error",
      "AbstractController::ActionNotFound"     => "unknown_action",
      "ActiveRecord::StatementInvalid"         => "invalid_statement",
      "ActionView::Template::Error"            => "template_error",
      "ActionController::MissingExactTemplate" => "missing_exact_template",
    )

    cattr_accessor :wrapper_exceptions, default: [
      "ActionView::Template::Error"
    ]

    cattr_accessor :silent_exceptions, default: [
      "ActionController::RoutingError",
      "ActionDispatch::Http::MimeNegotiation::InvalidType"
    ]

    attr_reader :backtrace_cleaner, :exception, :wrapped_causes, :line_number, :file

    def initialize(backtrace_cleaner, exception)
      @backtrace_cleaner = backtrace_cleaner
      @exception = exception
      @exception_class_name = @exception.class.name
      @wrapped_causes = wrapped_causes_for(exception, backtrace_cleaner)

      expand_backtrace if exception.is_a?(SyntaxError) || exception.cause.is_a?(SyntaxE
    end

    def unwrapped_exception
      if wrapper_exceptions.include?(@exception_class_name)
```

routing
system_testing
testing
journey.rb
railtie.rb
routing.rb

```
57              exception.cause
58            else
59              exception
60            end
61          end
62
63          def rescue_template
64            @@rescue_templates[@exception_class_name]
65          end
66
67          def status_code
68            self.class.status_code_for_exception(unwrapped_exception.class.name)
69          end
70
71          def exception_trace
72            trace = application_trace
73            trace = framework_trace if trace.empty? && !silent_exceptions.include?(@exception
74            trace
75          end
76
77          def application_trace
78            clean_backtrace(:silent)
79          end
80
81          def framework_trace
82            clean_backtrace(:noise)
83          end
84
85          def full_trace
86            clean_backtrace(:all)
87          end
88
89          def traces
90            application_trace_with_ids = []
91            framework_trace_with_ids = []
92            full_trace_with_ids = []
93
94            full_trace.each_with_index do |trace, idx|
95              trace_with_id = {
96                exception_object_id: @exception.object_id,
97                id: idx,
98                trace: trace
99              }
100
101             if application_trace.include?(trace)
102               application_trace_with_ids << trace_with_id
103             else
104               framework_trace_with_ids << trace_with_id
105             end
106
107             full_trace_with_ids << trace_with_id
108           end
109
110           {
111             "Application Trace" => application_trace_with_ids,
112             "Framework Trace" => framework_trace_with_ids,
113             "Full Trace" => full_trace_with_ids
114           }
115         end
116
117         def self.status_code_for_exception(class_name)
118           Rack::Utils.status_code(@@rescue_responses[class_name])
123         end
124
125         def source_extracts
126           backtrace.map do |trace|
127             file, line_number = extract_file_and_line_number(trace)
128
129             {
130               code: source_fragment(file, line_number),
131               line_number: line_number
```

```
132             }
133           end
134         end
135
136         def trace_to_show
137           if traces["Application Trace"].empty? && rescue_template != "routing_error"
138             "Full Trace"
139           else
140             "Application Trace"
141           end
142         end
143
144         def source_to_show_id
145           (traces[trace_to_show].first || {})[:id]
146         end
147
148         private
149           def backtrace
150             Array(@exception.backtrace)
151           end
152
153           def causes_for(exception)
154             return enum_for(__method__, exception) unless block_given?
155
156             yield exception while exception = exception.cause
157           end
158
159           def wrapped_causes_for(exception, backtrace_cleaner)
160             causes_for(exception).map { |cause| self.class.new(backtrace_cleaner, cause) }
161           end
162
163           def clean_backtrace(*args)
164             if backtrace_cleaner
165               backtrace_cleaner.clean(backtrace, *args)
166             else
167               backtrace
168             end
169           end
170
171           def source_fragment(path, line)
172             return unless Rails.respond_to?(:root) && Rails.root
173             full_path = Rails.root.join(path)
174             if File.exist?(full_path)
175               File.open(full_path, "r") do |file|
176                 start = [line - 3, 0].max
177                 lines = file.each_line.drop(start).take(6)
178                 Hash[*(start + 1..(lines.count + start)).zip(lines).flatten]
179               end
180             end
181           end
182
183           def extract_file_and_line_number(trace)
184             # Split by the first colon followed by some digits, which works for both
185             # Windows and Unix path styles.
186             file, line = trace.match(/^(.+?):(\d+).*$/, &:captures) || trace
187             [file, line.to_i]
188           end
189
190           def expand_backtrace
191             @exception.backtrace.unshift(
192               @exception.to_s.split("\n")
193             ).flatten!
194           end
195       end
196     end
```