

solargraph-dead_end

Beware the Dreaded Dead End!

By @schneems



```

    Solargraph::Logging.logger.info "Adding #{model_attrs.count} attributes a
model_attrs.map
    Solargraph::Pin::Method.new(
      name: attr[:name],
      comments: "@return [#{type_translation.fetch(attr[:type], attr[:type])}
      location: attr[:location],
      closure: Solargraph::Pin::Namespace.new(name: module_names.join('::'))
      _name}"),
      scope: :instance,
      attribute: true
    )
  end
end

```

.map with no do:
line 81

```

def ruby_matchers
  [
    MetaSource::Association::BelongsToMatcher.new,
    MetaSource::Association::HasManyMatcher.new,
    MetaSource::Association::HasOneMatcher.new,
    MetaSource::Association::HasAndBelongsToManyMatcher.new
  ]
end

def col_with_type(line)
end

def type_translation
  {
    'decimal' => 'BigDecimal',
    'integer' => 'Integer',
    'date' => 'Date',
    'datetime' => 'ActiveSupport::TimeWithZone',
    'string' => 'String',
    'boolean' => 'Boolean',
    'float' => 'Float',
    'text' => 'String'
  }
end
end
end
end

```

helpful feedback:
line 120

```
└─ solargraph-rails [main] ⚡ ruby lib/solargraph/rails/pin_creator.rb  
lib/solargraph/rails/pin_creator.rb:119: syntax error, unexpected `end', expecting end-of-input  
└─ solargraph-rails [main] ⚡ █
```

[RELEASES](#)[BLOG](#)[GEMS](#)

dead_end *3.1.1*

When you get an "unexpected end" in your syntax this gem helps you find it

```
└─ solargraph-rails [main] ⚡ dead_end lib/solargraph/rails/pin_creator.rb  
--> /Users/frederickmeissner/devprojects/solargraph-rails/lib/solargraph/rails/pin_creator.rb
```

```
Unmatched `end', missing keyword (`do', `def`, `if`, etc.) ?
```

```
5 module Solargraph  
6   module Rails  
7     class PinCreator  
15      def create_pins  
> 80      Solargraph::Logging.logger.info "Adding #{model_attrs.count} attributes as pins"  
> 81      model_attrs.map  
> 90      end  
91    end  
117  end  
118 end  
119 end
```



actually helpful
feedback

CLI is inconvenient

```

    Solargraph::Logging.logger.info "Adding #{model_attrs.count} attributes a
model_attrs.map
    Solargraph::Pin::Method.new(
      name: attr[:name],
      comments: "@return [#{type_translation.fetch(attr[:type], attr[:type])
location: attr[:location],
      closure: Solargraph::Pin::Namespace.new(name: module_names.join('::')
      _name}"),
      scope: :instance,
      attribute: true
    )
  end
end

```

```

def ruby_matchers
  [
    MetaSource::Association::BelongsToMatcher.new,
    MetaSource::Association::HasManyMatcher.new,
    MetaSource::Association::HasOneMatcher.new,
    MetaSource::Association::HasAndBelongsToManyMatcher.new
  ]
end

```

```

def col_with_type(line)
end

```

```

def type_translation
  {
    'decimal' => 'BigDecimal',
    'integer' => 'Integer',
    'date' => 'Date',
    'datetime' => 'ActiveSupport::TimeWithZone',
    'string' => 'String',
    'boolean' => 'Boolean',
    'float' => 'Float',
    'text' => 'String'
  }
end
end
end
end

```

appears while I
type

Can we get dead_end into our editors?

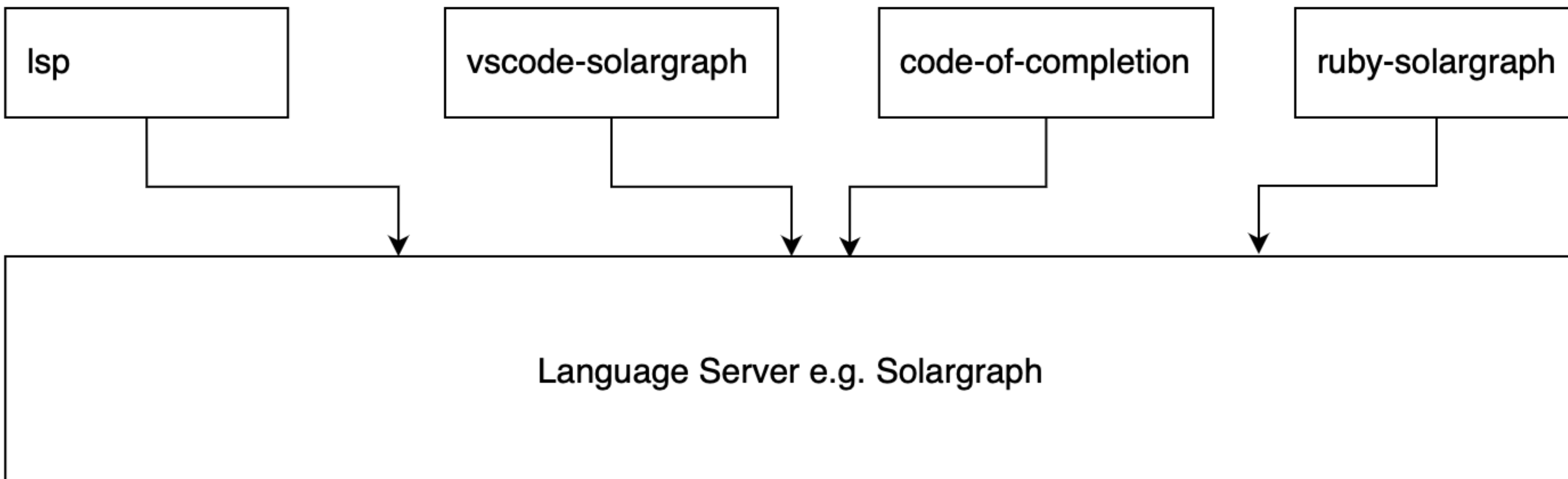
Yes!

Solargraph

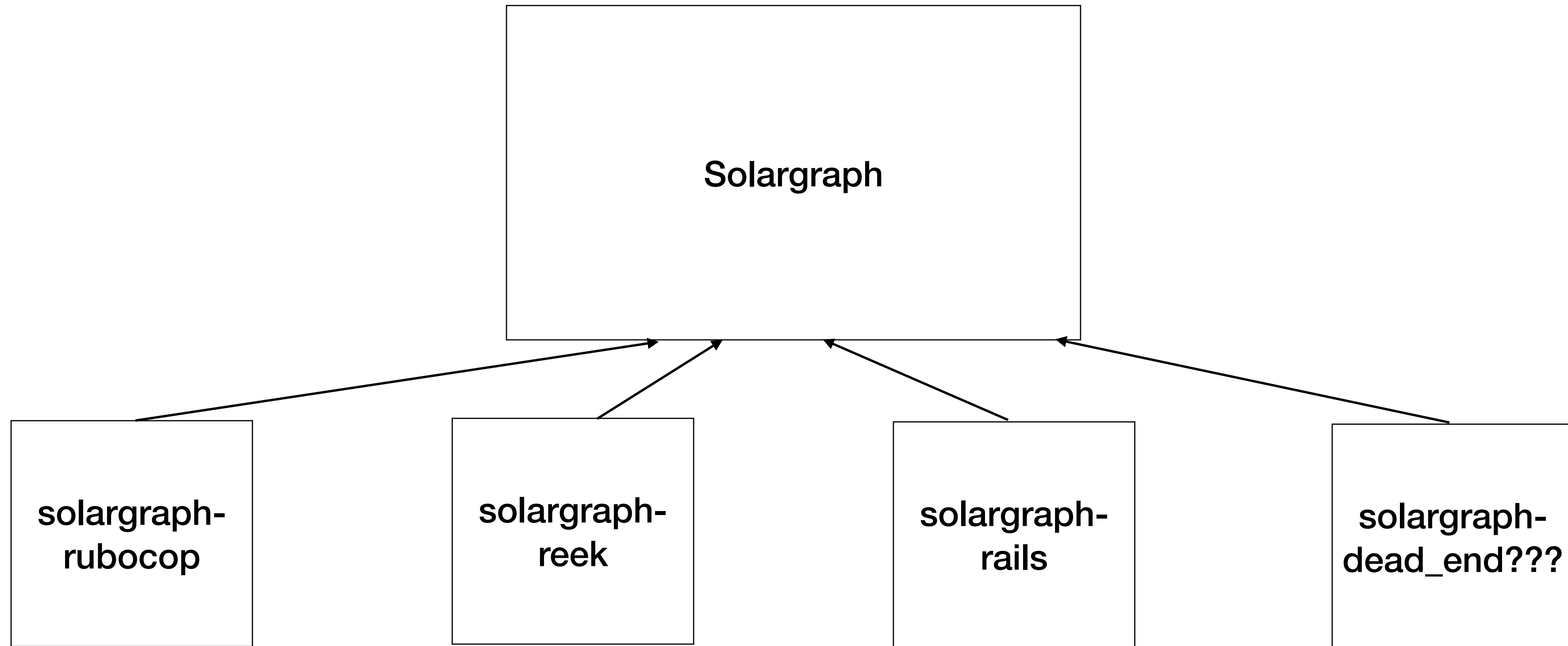
A computer brain for your Ruby

- A gem
 - ... that runs a server
 - ... that reads your code
 - ... that can talk to your editor
- Knows about methods, constants, return types etc.
 - "Where is the definition of this method"
 - "What methods are available on this object"

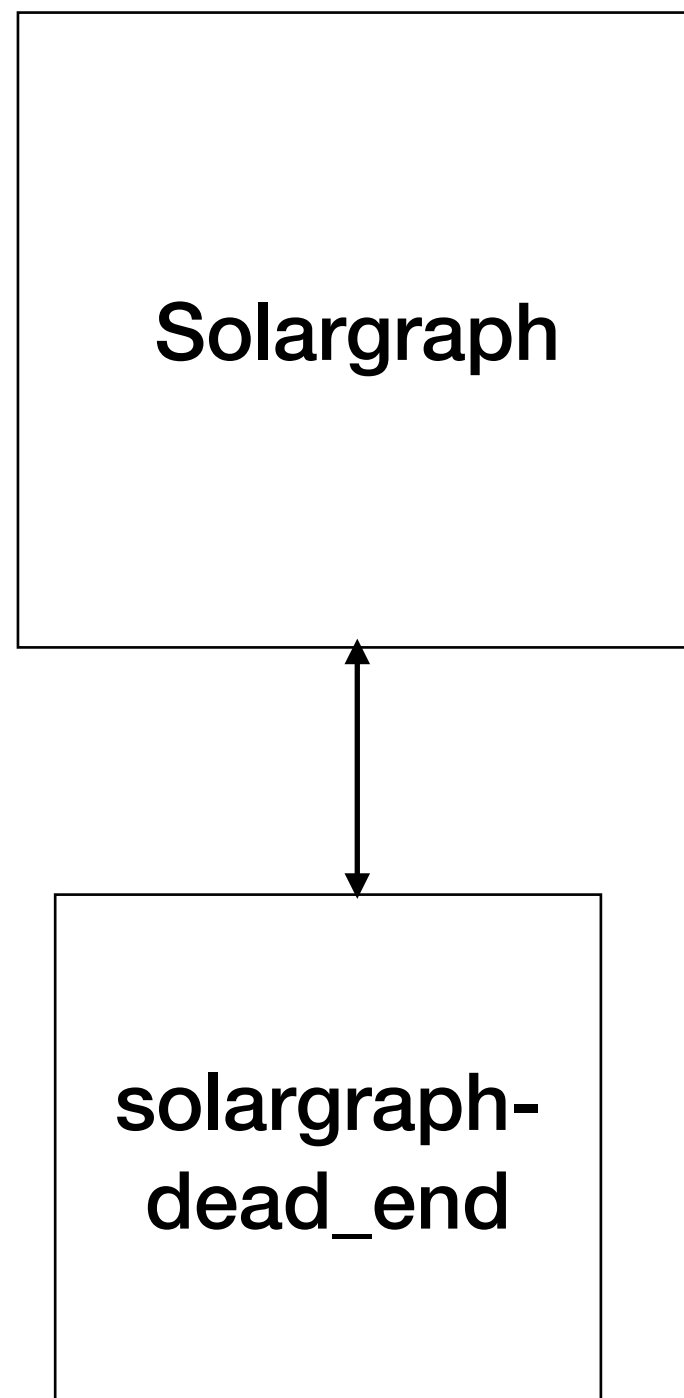
Solargraph



Plugins

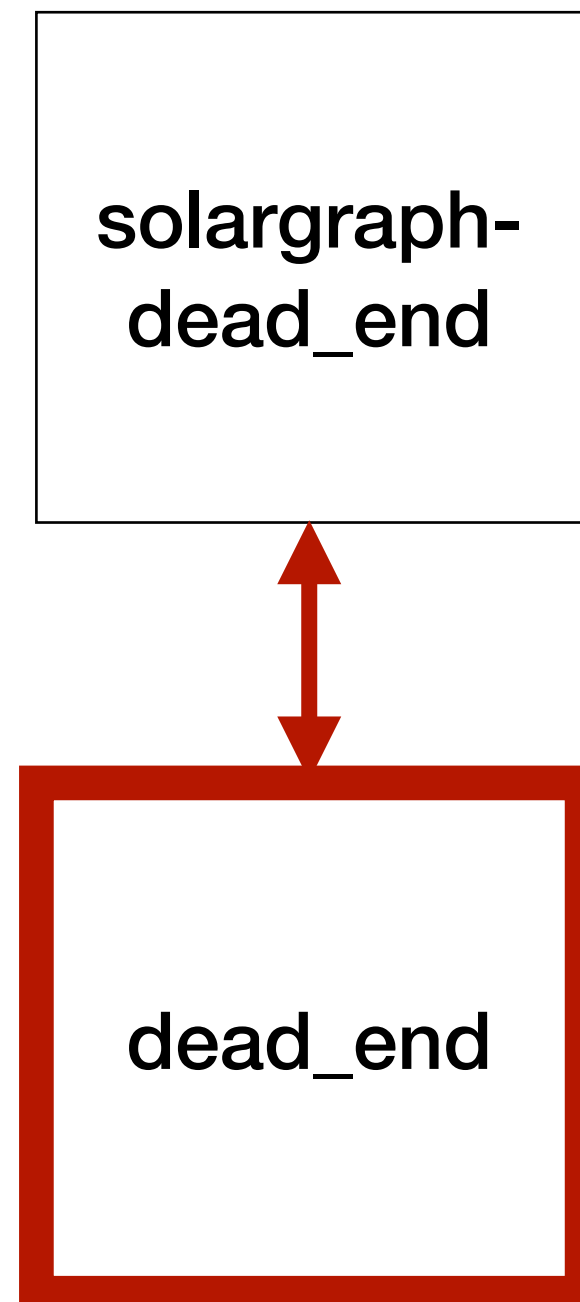


Easy Part



```
{
  message: explain.errors.first + "\n" + document,
  range: {
    start: {
      line: relevant_lines.first.line_number - 1,
      character: 0
    },
    end: {
      line: relevant_lines.last.line_number - 1,
      character: relevant_lines.last.line.length
    }
  },
  severity: 1,
  source: "DeadEnd",
  code: "Syntax"
}
```

Hard Part



```
search = ::DeadEnd::CodeSearch.new(source.code, record_dir: nil).call

blocks = search.invalid_blocks

return [] if blocks.empty?

result = []
blocks.each do |block|
  explain = ::DeadEnd::ExplainSyntax.new(
    code_lines: block.lines
  ).call

  lines = ::DeadEnd::CaptureCodeContext.new(
    blocks: block,
    code_lines: search.code_lines
  ).call

  document = ::DeadEnd::DisplayCodeWithLineNumbers.new(
    lines: lines,
    terminal: @terminal,
    highlight_lines: block.lines
  ).call
```

```

    Solargraph::Logging.logger.info "Adding #{model_attrs.count} attributes a
model_attrs.map
    Solargraph::Pin::Method.new(
      name: attr[:name],
      comments: "@return [#{type_translation.fetch(attr[:type], attr[:type])
location: attr[:location],
      closure: Solargraph::Pin::Namespace.new(name: module_names.join('::')
_name}"),
      scope: :instance,
      attribute: true
    )
  end
end

def ruby_matchers
  [
    MetaSource::Association::BelongsToMatcher.new,
    MetaSource::Association::HasManyMatcher.new,
    MetaSource::Association::HasOneMatcher.new,
    MetaSource::Association::HasAndBelongsToManyMatcher.new
  ]
end

def col_with_type(line)
end

def type_translation
  {
    'decimal' => 'BigDecimal',
    'integer' => 'Integer',
    'date' => 'Date',
    'datetime' => 'ActiveSupport::TimeWithZone',
    'string' => 'String',
    'boolean' => 'Boolean',
    'float' => 'Float',
    'text' => 'String'
  }
end
end
end
end
end

```

not so helpful
feedback, while
you edit

Unmatched `end`, missing keyword (`do`, `def`, `if`, etc.)

```
• Solargraph::Logging.logger.info "Adding #{model_attrs.count} attributes as pins"
  model_attrs.map
    Solargraph::Pin::Method.new(
      name: attr[:name],
      comments: "@return [#{type_translation.fetch(attr[:type], attr[:type])}]",
      location: attr[:location],
      closure: Solargraph::Pin::Namespace.new(name: module_names.join('::') + "::#{model_name}"),
      scope: :instance,
      attribute: true
    )
  end
end

def ruby_matchers
  [
    MetaSource::Association::BelongsToMatcher.new,
    MetaSource::Association::HasManyMatcher.new,
    MetaSource::Association::HasOneMatcher.new,
    MetaSource::Association::HasAndBelongsToManyMatcher.new
  ]
end

def col_with_type(line)
end

def type_translation
{
  'decimal' => 'BigDecimal',
  'integer' => 'Integer',
  'date' => 'Date',
  'datetime' => 'ActiveSupport::TimeWithZone',
  'string' => 'String',
  'boolean' => 'Boolean',
  'float' => 'Float',
  'text' => 'String'
}
```

better!

```

class MyClass
  def a_thing
    "hello world"

  def my_method
    puts a_thing
  end

  # some more code
end

```

```

class MyOtherClass
  def another_method
    [1, 2, 3].each do |x|
  end
end

```

Unmatched keyword, missing `end' ?

```

> 12 class MyOtherClass
> 13   def another_method
> 14     [1, 2, 3].each do |x|
> 15   end
> 16 end

```


DeadEnd(Syntax)

[View Problem](#) No quick fixes available

```


class MyOtherClass
  def another_method
    [1, 2, 3].each do |x|
  end
end

```



Search or jump to... /

Pull requestsIssuesMarketplaceExplore

 iftheshoefritz / solargraph-dead_end

Public

<> Code

Issues

Pull requests

Actions

Projects


Wiki

Security

main

1 branch

2 tags

 iftheshoefritz

Update authors



Search Gems... 🔍

RELEASESBLOGGEM

solargraph-dead_end 0.2.0

Solargraph DeadEnd reporter tells you where you're missing a closing `end` for your method definitions or blocks

VERSIONS:

0.2.0 - December 13, 2021 (261 KB)
0.1.0 - December 10, 2021 (7.5 KB)

RUNTIME DEPENDENCIES (2):

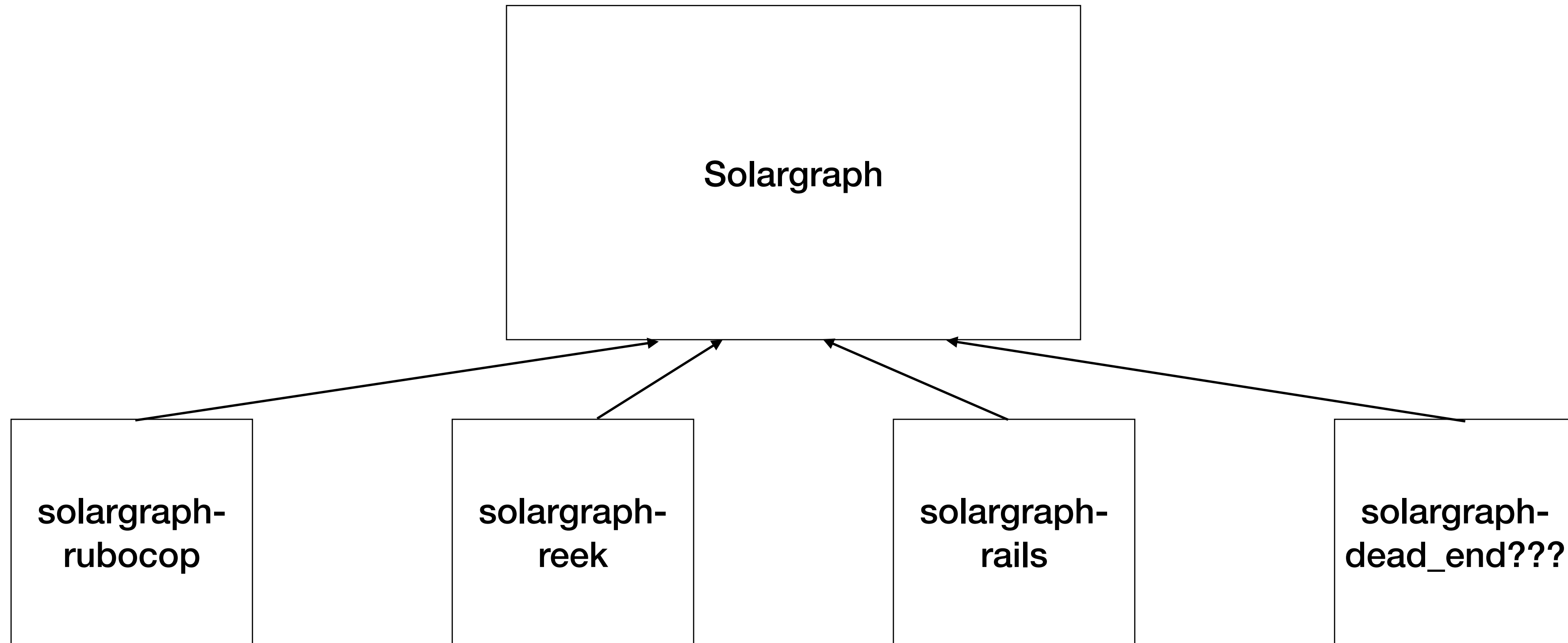
dead_end `>= 0`

solargraph `>= 0.39.1`

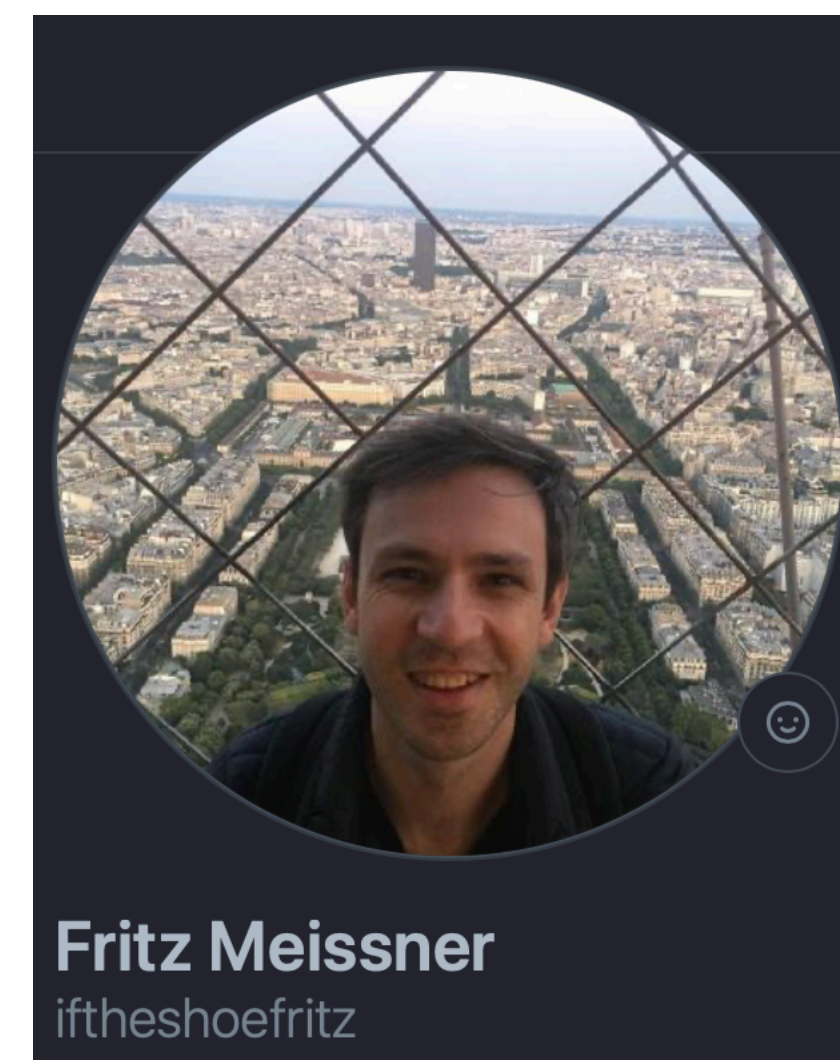
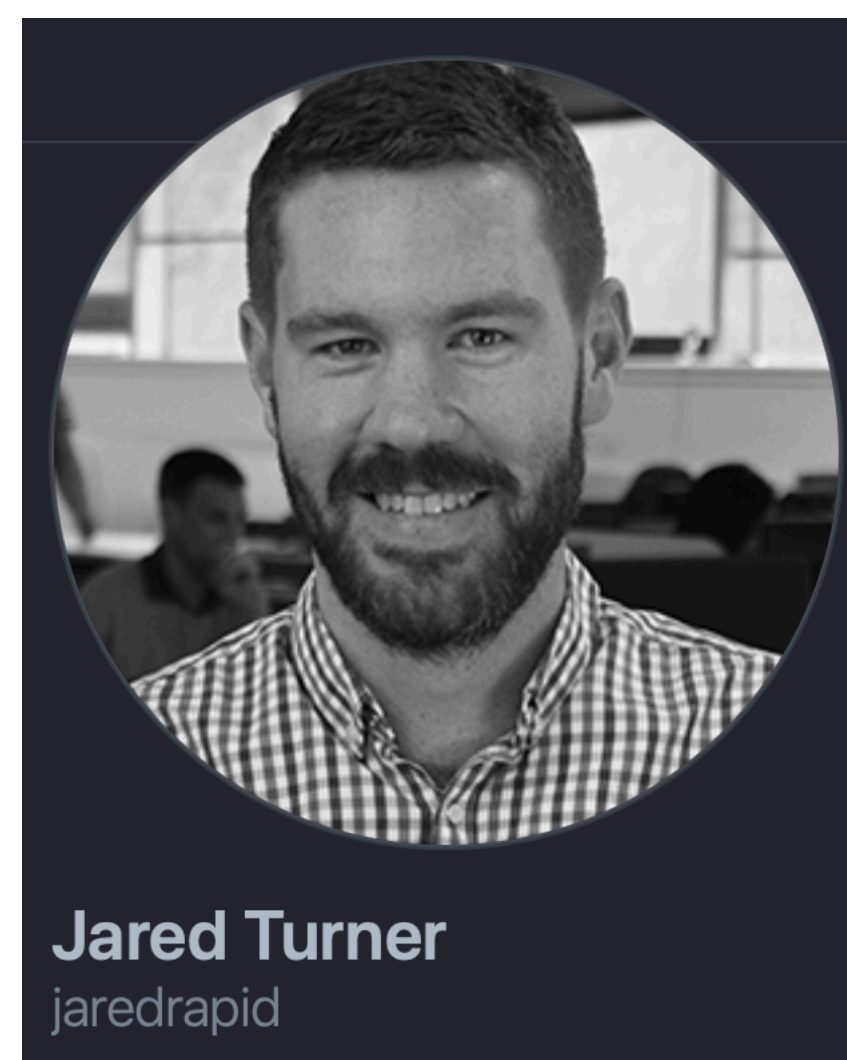
[Show all transitive dependencies →](#)

OWNERS:

Plugins



https://github.com/ifttheshoefritz/solargraph-dead_end



thoughtbot