Ruby Extensions

Stanko Krtalić Rusendić
@monorkin
Stankec

A question first

Have you guys ever been to IKEA?

They have a huge catalogue of things and prices vary from country to country

How to find the lowest price?

```
Country 1
  Item 1 : description (price in euros)
  Item 2 : description (price in euros)
  Item 3 : description (price in euros)
  Item 4 : description (price in euros)
  Item 5 : description (price in euros)
  Item 6 : description (price in euros)
  Item 7 : description (price in euros)
  Item 8 : description (price in euros)
  Item 9 : description (price in euros)
Country 2
  Item 1 : description (price in euros)
  Item 2 : description (price in euros)
  Item 3 : description (price in euros)
  Item 4 : description (price in euros)
  Item 5 : description (price in euros)
  Item 6 : description (price in euros)
  Item 7 : description (price in euros)
  Item 8 : description (price in euros)
  Item 9 : description (price in euros)
```

- The catalogue is stored as a .TXT file
- The file is organised in a non-standard format, shown on the left
- It has ~750MB
- It contains 8,000,000 line items

Let’s use Ruby to find the lowest price of meatballs!

```ruby
def self.catalogue_to_hash(catalogue)
  active_country = ''
  prices = {}

  catalogue.each do |line|
    data = line.scan(/^\s\s(.+)\s\:\s(.+)\s\((.+)\)$/)
    if data.empty?
      active_country = line
      prices[active_country] ||= {}
    else
      item, description, price = data.first
      prices[active_country][item] = {
        description: description,
        price: price.to_f
      }
    end
  end

  prices
end
```

It takes ~7min to parse this file with Ruby
It uses ~4GB of RAM

Ok… That’s slow!

Let’s try something else

```ruby
require_relative '../ext/CParser/CParser'
include CParser

prices = parse 'ikea_catalogue.txt'

prices.each do |k, v|
  puts "#{k}: #{v['Meatballs'][:price]}"
end
```

Takes ~50s to parse!
Uses ~850MB of RAM!

What kind of black magic is this!?

Let’s take a peek at the code

```
└$ pwd
/Users/Stanko/Desktop/Beyond plain Ruby/ikea_catalogue/ext/CParser
┌Stanko@Stankos-MBP  ~/Desktop/Beyond plain Ruby/ikea_catalogue/ext/CParser ‹2.1.2›
└$ ls -al
total 64
drwxr-xr-x  7 Stanko  staff   238 Mar  7 11:03 .
drwxr-xr-x  3 Stanko  staff   102 Mar  5 18:19 ..
-rwxr-xr-x  1 Stanko  staff  9924 Mar  7 10:27 CParser.bundle
-rw-r--r--  1 Stanko  staff  7066 Mar  7 08:55 Makefile
-rw-r--r--  1 Stanko  staff   153 Mar  5 18:20 extconf.rb
-rw-r--r--  1 Stanko  staff  2156 Mar  7 10:27 parser.c
-rw-r--r--  1 Stanko  staff  4020 Mar  7 10:27 parser.o
┌Stanko@Stankos-MBP  ~/Desktop/Beyond plain Ruby/ikea_catalogue/ext/CParser ‹2.1.2›
└$
```

There is no CParser Ruby file here!?
Where did the module come from?

Ruby Extensions

Ruby Extensions

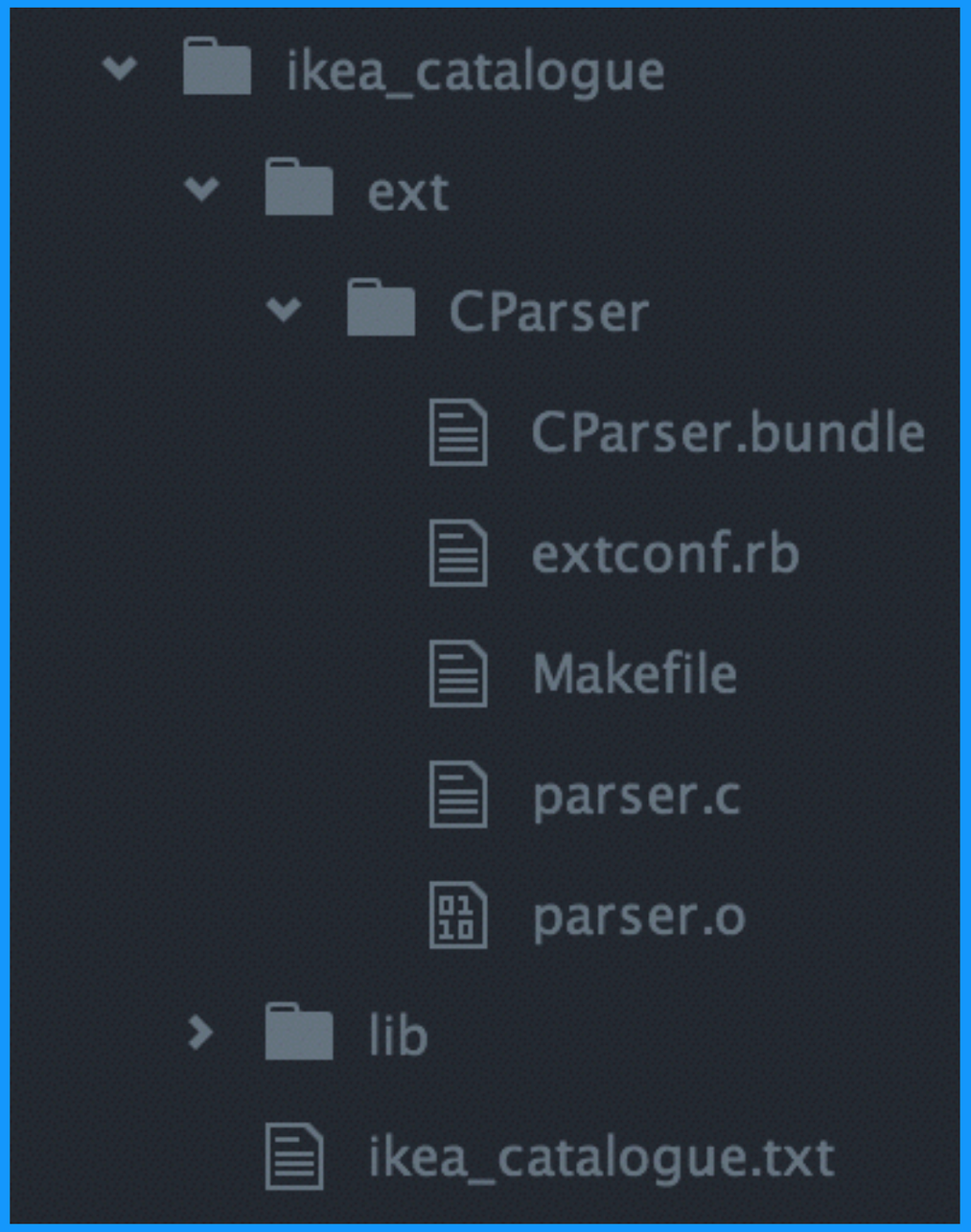
Libraries written in languages other than Ruby, which act and behave like native Ruby objects

But why, you ask.

- Mostly for speed improvements
- Integration with non-ruby libraries
- Integration with closed-source / precompiled / 3rd party libraries

HOW?

/ext - A home for extensions



- All extensions live in the ext folder
- Extensions should be placed in a subfolder with their name to prevent namespace collisions
- Each extension has to have an extconf.rb

extconf.rb

```ruby
require 'mkmf'

# Give it a name
extension_name = 'CParser'

# The destination
dir_config(extension_name)

# Do the work
create_makefile(extension_name)
```

- Defines the name of the extension
- Contains the path to the extension's source
- Auto generates the Makefile

Implementation

```c
#include "ruby.h"
#include <string.h>
#include <stdio.h>
#include <ctype.h>

/* Define module object */
VALUE CParser = Qnil;

/* Define functions */
void Init_CParser();
VALUE method_parse(VALUE self, VALUE file_path);
char *trimwhitespace(char *str);

/* Implement functions */
void Init_CParser() {
  CParser = rb_define_module("CParser");
  rb_define_method(CParser, "parse", method_parse, 1);
}

VALUE method_parse(VALUE self, VALUE file_path) {
  static int array_size = 255;
  FILE *fp;
  char str[array_size];
  char product[array_size];
  char description[array_size];
  double price;
  VALUE active_country;
  VALUE result = rb_hash_new();
  VALUE temp_hash;

  /* opening file for reading, if unable to raise an error */
  fp = fopen(RSTRING_PTR(file_path), "r");
  if(fp == NULL) {
    rb_raise(rb_eIOError, "file not found or unable to open");
  }
```

Implementation

```c
/* Define module object */
VALUE CParser = Qnil;

/* Define functions */
void Init_CParser();
VALUE method_parse(VALUE self, VALUE file_path);
char *trimwhitespace(char *str);

/* Implement functions */
void Init_CParser() {
  CParser = rb_define_module("CParser");
  rb_define_method(CParser, "parse", method_parse, 1);
}
```

Defines a module

Defines a method

At leas a module / class definition is needed

```
VALUE method_parse(VALUE self, VALUE file_path) {
  static int array_size = 255;
  FILE *fp;
  char str[array_size];
  char product[array_size];
  char description[array_size];
  double price;
  VALUE active_country;
  VALUE result = rb_hash_new();
  VALUE temp_hash;

  /* opening file for reading, if unable to raise an error */
  fp = fopen(RSTRING_PTR(file_path), "r");
  if(fp == NULL) {
    rb_raise(rb_eIOError, "file not found or unable to open");
  }

  /* Iterate through file */
  while( fgets(str, array_size, fp) != NULL ) {
    if (str[0] != ' ') {
      *str = *strtok(str, "\n");
      active_country = rb_str_new(str, strlen(str));
      if (TYPE(rb_hash_aref(result, active_country)) == T_NIL) {
        rb_hash_aset(result, active_country, rb_hash_new());
      }
    } else {
      sscanf(str, "  %[^:]: %[^(](%lf)\n", product, description, &price);
      *product = *trimwhitespace(product);
      *description = *trimwhitespace(description);
      temp_hash = rb_hash_new();
      rb_hash_aset(
        temp_hash,
        ID2SYM(rb_intern("description")),
        rb_str_new(description, strlen(description))
      );
      rb_hash_aset(
        temp_hash,
        ID2SYM(rb_intern("price")),
        DBL2NUM((double)price)
      );
      rb_hash_aset(
        rb_hash_aref(result, active_country),
        rb_str_new(product, strlen(product)),
        temp_hash
      );
    }
  }
  fclose(fp);

  return(result);
}
```

Declare Ruby variables

Convert Ruby object to C primitive

Open the file or raise an IOError exception

Iterate through each line

Convert C primitive to Ruby object

Build a Ruby Hash object

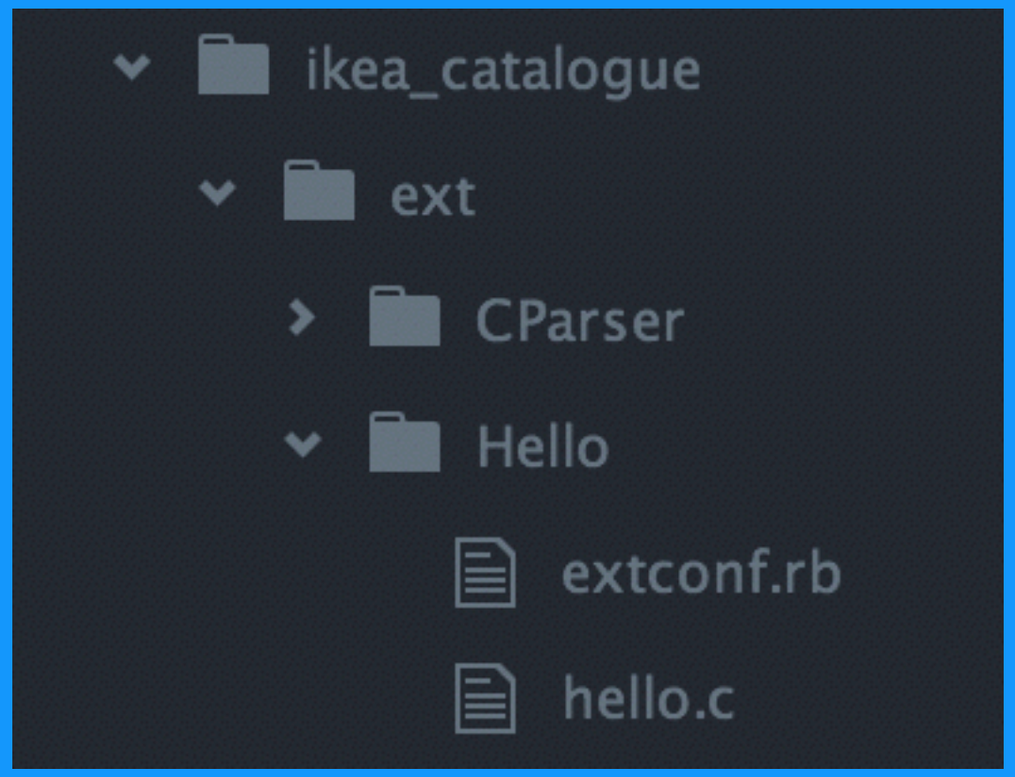
Return result

Makefile

- Run `*ruby extconfig.rb*` to generate a Makefile
- To make the code you wrote useable by Ruby run `*make*`
- The output of this will either be an .o, .so, .dll or .bundle file, depending on your OS
- That is your new Ruby module / class

Hello Ruby Extensions

Create a new subfolder



Create a extconf.rb file

```ruby
require 'mkmf'

# Give it a name
extension_name = 'Hello'

# The destination
dir_config(extension_name)

# Do the work
create_makefile(extension_name)
```

Write your code

```c
#include "ruby.h"
#include <string.h>
#include <stdio.h>
#include <ctype.h>

/* Define module object */
VALUE Hello = Qnil;

/* Define functions */
void Init_Hello();
VALUE method_hello(VALUE self, VALUE name);

/* Implement functions */
void Init_Hello() {
  Hello = rb_define_module("Hello"); // Name of the module
  rb_define_method(Hello, "hello", method_hello, 1); // Name of the method
}

VALUE method_hello(VALUE self, VALUE name) {
  char hello[6] = "Hello ";
  char *result = malloc(strlen(hello) + strlen(RSTRING_PTR(name)) + 1);

  strcpy(result, hello);
  strcat(result, RSTRING_PTR(name));

  return(rb_str_new(result, strlen(result)));
}
```

The result

```
└$ ruby extconf.rb
creating Makefile
┌Stanko@Stankos-MBP  ~/Desktop/Beyond plain Ruby/ikea_catalogue/ext/Hello ‹2.1.2›
└$ make
compiling hello.c
linking shared-object Hello.bundle
┌Stanko@Stankos-MBP  ~/Desktop/Beyond plain Ruby/ikea_catalogue/ext/Hello ‹2.1.2›
└$ pry
[1] pry(main)> require_relative 'Hello'
=> true
[2] pry(main)> include Hello
=> Object
[3] pry(main)> hello("Borko")
=> "Hello Borko"
[4] pry(main)>
```

Some will say:
‘But I don’t like C’

Neither do I !

Many supported languages

Basically anything that can be compiled

C++ example

```
Object method_detect_faces(Rice::String image_path, Rice::String cascade_path) {
  Mat img = imread(image_path.c_str(), 0);

  if(!img.data)
  {
    throw Rice::Exception(rb_eIOError, "Could not open or find image file.");
  }

  equalizeHist(img, img);
  Size s = img.size();

  CascadeClassifier cascade;
  if (!cascade.load(cascade_path.c_str())) {
    throw Rice::Exception(rb_eIOError, "Could not open or find cascade file.");
  }

  std::vector<cv::Rect> faces;
  cascade.detectMultiScale(
    img, faces,
    1.05, 4,
    0 | CV_HAAR_SCALE_IMAGE,
    cvSize(10, 10), cvSize(30, 30)
  );

  Array result;
  for (size_t i = 0; i < faces.size(); i++)
  {
    Rect_<int> r = faces[i];
    Hash hash;
    hash[Symbol('x')] =      to_ruby((double)(r.x / s.width));
    hash[Symbol('y')] =      to_ruby((double)(r.y / s.height));
    hash[Symbol("width")] =  to_ruby((double)(r.width / s.width));
    hash[Symbol("height")] = to_ruby((double)(r.height / s.height));
    result.push(hash);
  }

  img.release();

  return result;
}

extern "C"
void Init_FaceDetector() {
  Module rb_mFaceDetector = define_module("FaceDetector");
  rb_mFaceDetector.define_method("detect_faces", &method_detect_faces);
}
```

Convert Ruby object to C++ object

Raise an IOError exception

Face detection using OpenCV

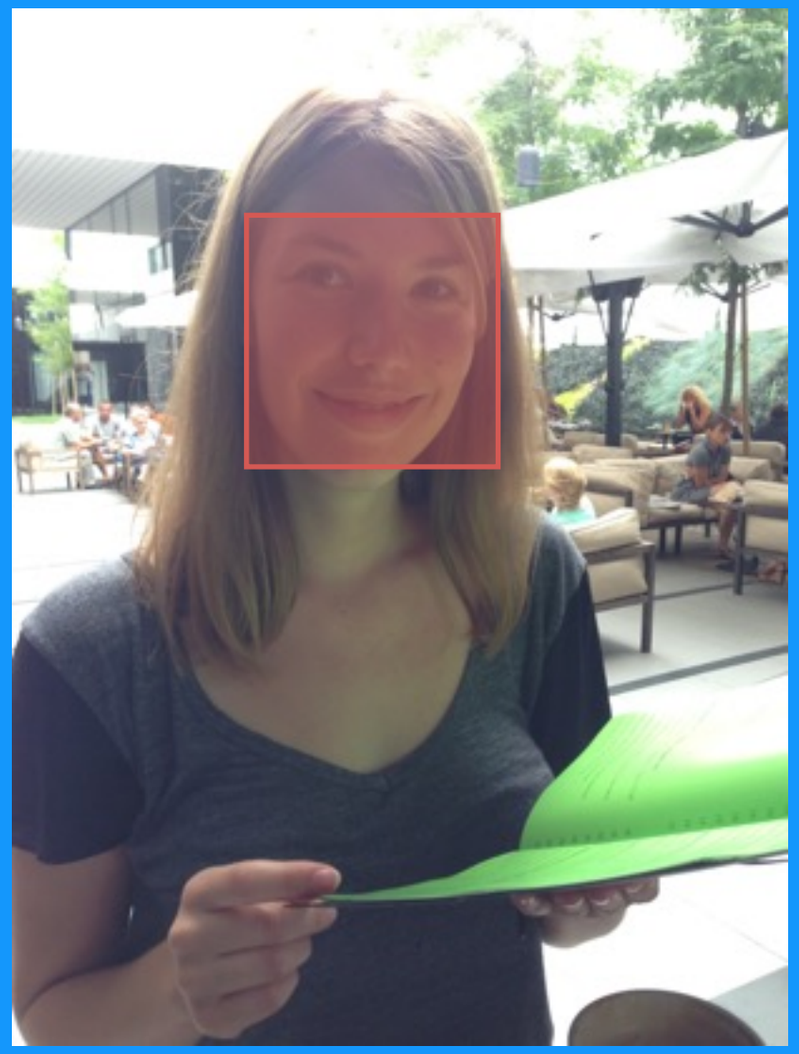
Convert C++ objects to Ruby objects

Define Ruby module

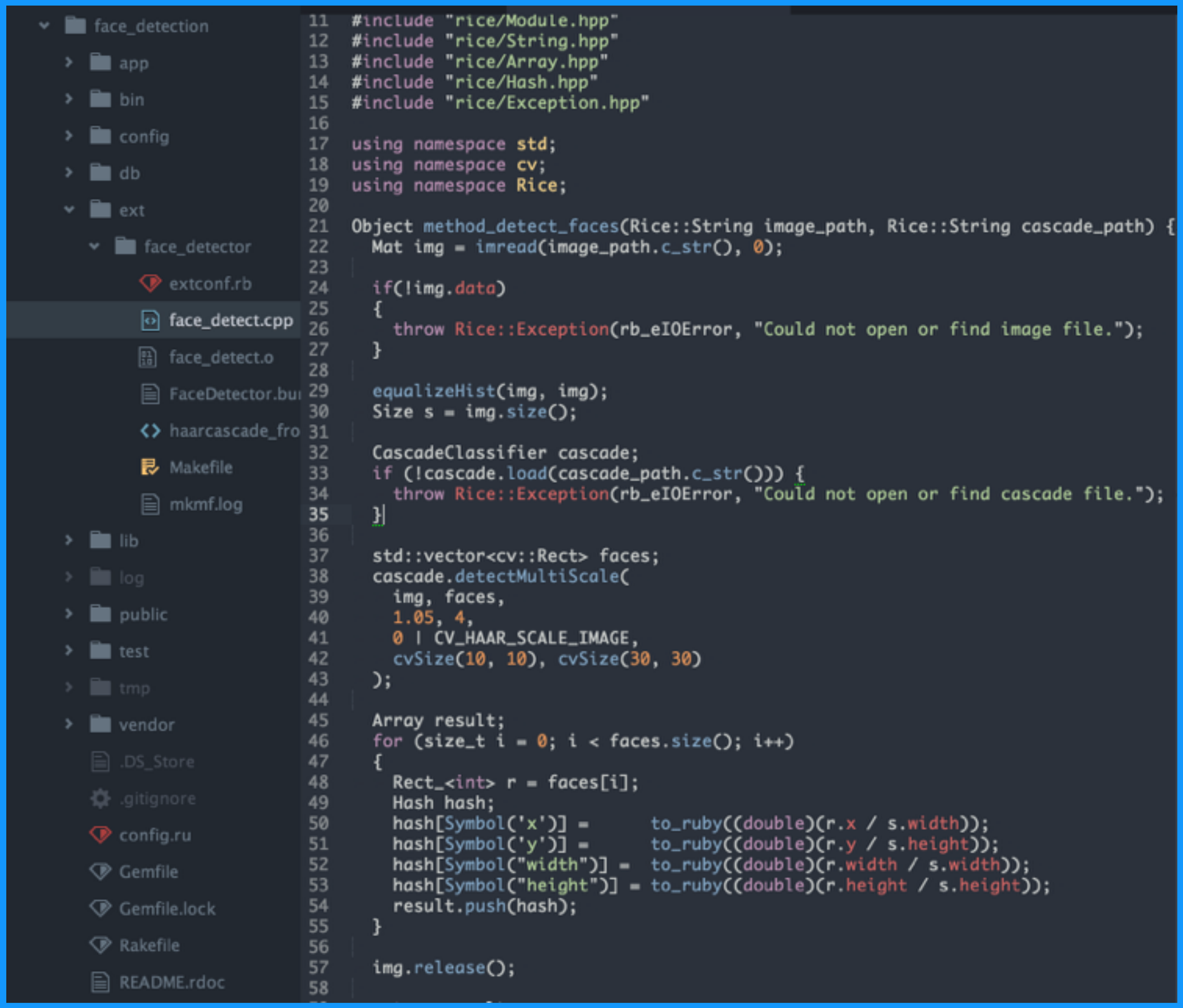
Define Ruby method for module

Result

```
└$ pry
[1] pry(main)> require_relative 'FaceDetector'
=> true
[2] pry(main)> include FaceDetector
=> Object
[3] pry(main)> detect_faces '/Users/Stanko/Desktop/karla.jpg', '/usr/local/share/OpenCV/haarcascades/haarcascade_frontalface_alt.xml'
=> [{:x=>0.3433, :y=>0.1132, :width=>0.2451, :height=>0.2434}]
```



It works with Rails!



That’s all 😘

References

- Chris Lalancette - Writing Ruby extensions in C (http://goo.gl/Ya6vMN)
- Aaron Patterson - Writing Ruby C extensions (http://goo.gl/5LFDhj)
- James Coglan - Your first Ruby Java extension (http://goo.gl/wdsVZ7)
- Brson - Embedding Rust in Ruby (http://goo.gl/xEgIL4)
- Sgonyea - objc_ruby-ext (http://goo.gl/GI2QkV)

Demo source

https://github.com/Stankec/ruby-lectures/tree/master/beyond_plain_ruby