Ruby Explorations IV

Mark Keane...CSI...UCD

Lets's build a program

Ruby iTunes:

A: designing object model and actions

B: implementing the world-world objects

C: implementing the world-world actions

D: the program-world objects & actions

E: some issues arising...

Ruby iTunes

- let's create a Ruby based iTunes
- it won't actually do anything but we can pretend
- later we will use rails to put an interface on it

Part A:

designing the object model and actions

Design: Objects

- what is the fundamental object unit ?
- the listener, buyer, seller, database, album, artist(s), song, genre, favourites ?
- we need to model the music CD world, the program world for handling data, and the buyer-seller world

Some Objects

Some Actions

- Jackson System Development (JSD) argued that you model the whole world

Some Objects Attributes

- these are not given, may be inherent or passing
- actions might change them
- overlapping attributes invite subsumption
- are some more fundamental; song, actor

Think of related tables

buying/selling moves ownership
from one actor to another

Design: Fundamentals

- the song looks like a good fundamental object
- could state its attributes simply as:

  :name, :album, :artist, :time, :owners, :id...
- we could build an album object from a set of songs, build the songs I own from ownership, could build a greatest hits for an artist, create a back catalogue, and so on

Design: Program Objects/Actions

- we have looked at objects & actions in the world-world
- there are equally important objects & actions in our program-world; like readers and error-handlers etc...
- we will define a few: error objs, reader objs, utility actions and so on

Part B:

implementing the world-world objects

Actor

- a standard class with two attributes
- actor making method takes 1 args and builds values for 2 attributes

```
class Actor
 attr_accessor :name, :id
 def initialize(name)
     @name = name
     @id = name.object_id
 end
end

p Actor.new("john")
```

actor.rb

id made up

```
#<Actor:0x2745c @id=80680,@name="john">
```

Song

```
class Song
 attr_accessor :name, :album, :artist, :time, :owners, :id
 def initialize(name, album, artist, time, owners, id)
     @name = name
     @album = album
     @time = time
     @artist = artist
     @owners = owners
    @id = id
 end
end
```

song.rb

Album

```
class Album
 attr_accessor :name, :tracks, :length, :artist, :id
 def initialize(name, tracks, length, artist, owners)
     @name = name
     @tracks = tracks
     @length = length
     @artist = artist
     @owners = owners
     @id = name.object_id
  end
end
```

id made up

album.rb

Part C:

implementing the world-world methods

Comment

- we need to be able to look at the objects
- we need to be able to play a song and buy it
- we need to be able to create albums
  from collections of songs
- for now...lets not worry where they come from

```
class Song
 attr_accessor :name, :album, :artist, :time, :owners, :id
 def initialize(name, album, artist, time, owners, id)
     @name = name
     @album = album
     @time = time
     @artist = artist
     @owners = owners
    @id = id
 end

 def to_s
     puts "<< #{@name} >> by #{@artist} in their album
           #{@album} is owed by #{@owners} .\n"
 end

 def isa?
     instance_of?(Song)
 end

 def play_song
     no = rand(10)
     no.times {print "#{@name} do be do..."}
     puts "\n"
 end

end
```

song.rb

```
class Song
 attr_accessor :name, :album, :artist, :time, :owners, :id
 def initialize(name, album, artist, time, owners, id)
     @name = name
     @album = album
     @time = time
     @artist = artist
     @owners = owners
    @id = id
 end

 def to_s
     puts "<< #{@name} >> by #{@artist} in their album
           #{@album} is owed by #{@owners} .\n"
 end

 def isa?
     instance_of?(Song)
 end

 def play_song
     no = rand(10)
     no.times {print "#{@name} do be do..."}
     puts "\n"
 end

end
```

print out

crappy predicate

ah..come on !

song.rb

```
class Album
 attr_accessor :name, :tracks, :length, :artist,:owners,:id
 def initialize(name, tracks, length, artist, owners)
     @name = name
     @tracks = tracks
     @length = length
     @artist = artist
     @owners = owners
     @id = name.object_id
 end

 def to_s
     puts "The album #{@name} by #{@artist}. \n"
 end

 def isa?
     instance_of?(Album)
 end

 def self.make_album(name,tracks, length, artist, owners)
     Album.new(name, tracks, length, artist, owners)
 end

 def self.build_all(albums = [])
    ….
 end

 def self.build_an_album_called(album_name)
     …
 end
```

print out

instance creator

class method

album.rb

Aside: **Uniq**

- uniq
- uniq!

```
>> foo = ["a","b","a1","c","b",
"b", "c", "a"]
=> ["a", "b", "a1", "c", "b",
"b", "c", "a"]

>> foo.uniq
=> ["a", "b", "a1", "c"]

>> foo
=> ["a", "b", "a1", "c", "b",
"b", "c", "a"]

>> foo.uniq!
=> ["a", "b", "a1", "c"]

>> foo
=> ["a", "b", "a1", "c"]
```

Aside: **Join**

```
>> foo = ["a","b","a1","c","b",
"b", "c", "a"]
=> ["a", "b", "a1", "c", "b",
"b", "c", "a"]

>> foo.join
=> "aba1cbbca"

>> foo
=> ["a", "b", "a1", "c", "b",
"b", "c", "a"]

>> foo.join(“ “)
=> “a b a1 c b b c a"
```

- join

Aside: **SPLIT**

```
>> foo = ["a","b","a1","c","b",
"b", "c", "a"]
=> ["a", "b", "a1", "c", "b",
"b", "c", "a"]

>> x = foo.join
=> "aba1cbbca"

>> x.split
=> ["aba1cbbca"]

>> x.split(/b/)
=> ["a", "a1c", "", “ca"]
>> x
=> "aba1cbbca"
```

Aside: **inject**

```
> foo
=> ["a", "b", "a1", "c"]
>> foo.inject {|a,b| a + b}
=> "aba1c"
>> [1,2,3,4,5].inject {|a,b| a + b}
=> 15
>> foo.inject {|a,b| [a,b]}
=> [[["a", "b"], "a1"], "c"]
```

- **inject** applies a block to an array, take pair-wise elements , evaluates them and then passes the value as the 1st part of the next pairwise comparison

Aside: ObjectSpace

```
ruby_engine
#<RubyVM::InstructionSequence:0x815934>
default_user_source_cache_dir
#<RubyVM::InstructionSequence:0x8159fc>
default_system_source_cache_dir
#<RubyVM::InstructionSequence:0x815a88>
default_bindir
#<RubyVM::InstructionSequence:0x815b64>
rescue in default_exec_format
#<RubyVM::InstructionSequence:0x815bdc>
default_exec_format
#<RubyVM::InstructionSequence:0x815c7c>
default_path
#<RubyVM::InstructionSequence:0x815d6c>
user_dir
#<RubyVM::InstructionSequence:0x815f38>
default_dir
#<RubyVM::InstructionSequence:0x815fb0>
default_sources
#<RubyVM::InstructionSequence:0x816028>
ensure_gem_subdirectories
#<RubyVM::InstructionSequence:0x81617c>
block in set_paths
#<RubyVM::InstructionSequence:0x8161f4>
set_paths
#<RubyVM::InstructionSequence:0x816280>
set_home
#<RubyVM::InstructionSequence:0x8162f8>
pre_uninstall
#<RubyVM::InstructionSequence:0x816370>
pre_install
#<RubyVM::InstructionSequence:0x8163e8>
post_uninstall
#<RubyVM::InstructionSequence:0x816460>
post_install
#<RubyVM::InstructionSequence:0x816550>
path
#<RubyVM::InstructionSequence:0x816604>
dir
#<RubyVM::InstructionSequence:0x8167a8>
gem
  end
Album
Album
@d
block in update_album
                '.%0*d' % [n, (sec_fraction / Rational(1,
10**n)).round]
valid_time?
              else
new_version
current_version
                ''
valid_nth_kday?
#<RubyVM::InstructionSequence:0x818904>
block in update_album
              if n < 1
dash
    strftime('T%T' +
gem_directory_name
valid_weeknum?
iso8601_timediv
  def iso8601_timediv(n) # :nodoc:
gems_directory
#<RubyVM::InstructionSequence:0x818d14>
block in update_album
push_all_highest_version_gems_on_load_path
valid_commercial?

valid_date?
  end
n
numbers
#<RubyVM::InstructionSequence:0x819174>
block in update_album
gem_version
valid_civil?
#<RubyVM::InstructionSequence:0x819318>
    super(str, fmt)
update_album
/opt/local/lib/ruby1.9/1.9.1/date.rb
/opt/local/lib/ruby1.9/1.9.1/date.rb
valid_ordinal?
required_version
date
#<RubyVM::InstructionSequence:0x819624>
make_album
date
  def self._strptime(str, fmt='%FT%T%z')
calculate_integers_for_gem_version
```

Aside: ObjectSpace

```
class Beno
      def hi
        puts "hi beno"
      end
end

p a = Beno.new
p b = Beno.new
p c = Beno.new
contents_a=  ObjectSpace.each_object(Beno)
p contents_a
contents_b =  ObjectSpace.each_object(Beno).to_a
p contents_b
```

returns enumerator

returns array

a_beno.rb

```
# NB. Ruby1.8 and ruby2.0 act differently; 1.8 throws an error

ruby  a_beno.rb
#<Beno:0x113b7c>
#<Beno:0x113b68>
#<Beno:0x113b54>
#<Enumerator: ObjectSpace:each_object(Beno)>
[#<Beno:0x113b54>, #<Beno:0x113b68>, #<Beno:0x113b7c>]
```

to_a converts a subclass of an array into an array

Part D:

the program-world objects & methods

itunes.rb
actor.rb
reader.rb
album.rb
error.rb
song.rb
utilities.rb
songs
owners

```
require 'csv'
require_relative 'actor','album','song','reader','utilities', 'error'

#songs_file = ARGV[0]                                  #for command line
#owners_file = ARGV[1]                                 #for command line
reader = Reader.new
songs_file = 'songs.csv'                               #in RubyMine
owners_file = 'owners.csv'                             #in RubyMine

puts "\nProcessing Songs from file: #{songs_file}"
$songs = reader.read_in_songs(songs_file)

puts "Processing Ownership from file: #{owners_file}"
$hash_owners = reader.read_in_ownership(owners_file)

puts "Building all owners..."
$actors = Actor.build_all()

puts "Updating songs with ownership details..."
$songs.each{|song| song.owners = $hash_owners[song.id]}

puts "Building All Albums..."
$albums = Album.build_all()

# markkean% ruby itunes.rb songs.csv owners.csv
```

instance of reader

Top-level

itunes.rb

ARGV[0]

ARGV[1]

Error

```
class MyErr
 attr_accessor :type, :holder, :method
 def initialize(type, holder, method)
     @type = type
     @holder = holder
     @method = method
 end

 def do_it
     if @type == "multiple_answer_error"
         then puts "Error: Item #{@holder} raised #{@type} in #{@method}"
     elsif @type == "not_found_error"
         then puts "Error: #{@holder} was #{@type} in #{@method}"
     else puts "Error: Have been given an unknown error type: #{@type}"
     end
 end

end
```

error.rb

Utilities

```
module Util
   def self.fetch(string_item, out = [])
       all = $songs + $actors + $albums
      found = all.select{|obj| string_item == obj.name}
       if found.size == 0
         then MyErr.new("not_found_error", string_item, "fetch").do_it
       elsif   found.size > 1
         then  MyErr.new("multiple_answer_error", string_item, "fetch").do_it
       elsif  found.size == 1
         then   found.first
       end
   end
end


class Array
  def clean_up()
    self.join(" ").split(" ").uniq   #this could be more elegant
  end
end
```

utilities.rb

- its not pretty but, hey, it works

Aside: **downcase**

```
>> "Cats and Jammers RULE !!!".downcase
=> "cats and jammers rule !!!"

>> "let me shout to you".upcase
=> "LET ME SHOUT TO YOU"

>> "ruth, let me capitalise you".capitalize
=> "Ruth, let me capitalise you"

>> "but we are all small already".downcase
=> "but we are all small already"

>> "AND WE ARE ALL LARGE".upcase
=> "AND WE ARE ALL LARGE"
```

Aside: **when**

```
def sheep_spotter(test)
  case
    when test == "sheep"
      puts "Yup, its a sheep alright."
    when test == "wolf"
      puts "No, this is a wolf."
    when test.instance_of?(String)
      puts "I really don't know what this is?"
    else "Are you trying to poison me!"
   end
end

p sheep_spotter(nil)
p sheep_spotter("sheep")
p sheep_spotter("wolf")
p sheep_spotter("elf with a ham")
```

when *test true do_next line*

may finish with **else**

a_wheno.rb

```
$ ruby a_wheno.rb
 "Are you trying to poison me !"
 Yup, its a sheep alright.
 nil
 No, this is a wolf.
 nil
 I really don't know what this is?
 nil
$
```

why the nils

csv.files I

row headers

with comments

```
"Songname","Artist","Album","Time","Id"
# THIS FILE CONTAINS ALL SONGS IN iTunes
# Aug 29 2014
# Plastic Beach by Gorillaz
"Superfast Jellyfish","Gorillaz","Plastic Beach",2.55,10
"On Melancholy Hill","Gorillaz","Plastic Beach",3.54,11
"Broken","Gorillaz","Plastic Beach”,3.17,12
# I Had the Blues and I Shook Them Loose by  Bombay Bicycle Club
"Always Like This","Bombay Bicycle Club","I Had the Blues and I Shook Them
Loose",4.06,13
# Horehound by The Dead Weather
"60 Feet Tall","The Dead Weather","Horehound",5.33,14
"I Cut Like a Buffalo","The Dead Weather","Horehound",3.28,15
"So Far From Your Weapon","The Dead Weather","Horehound",3.40,16
# Straight in No Kissing by  Dirty Epics
"Way Too Pretty","Dirty Epics","Straight in No Kissing",3.19,17
"Pony","Dirty Epics","Straight in No Kissing",3.14,18
"The Cure","Dirty Epics","Straight in No Kissing",2.96,19
# Crystal Castles by  Crystal Castles
"Untrust Us","Crystal Castles","Crystal Castles",3.06,20
"Air War","Crystal Castles","Crystal Castles",4.02,21
#
# Disco Crap  by  Multiple Artists
"Staying Alive","The GBs","Disco Crap",3.06,22
"Staying Alive","The GBs","Disco Crap",3.06,23
"Bling","SalboNedam","Disco Crap",4.02,24
```

a

songs.scv

csv.files II

- quick way of recording who owns what
- pick up id from songs.csv
- both are read in to build objects by reader.rb

row headers

with comments

```
"Songid","Libraries"
# File has Mapping from Song-Ids to Owners
# Aug 30 2014
##########################################
#           All SONGS
##########################################
10,"apple markk stinkypig"
11,"apple markk stinkypig"
12,"apple markk stinkypig"
13,"apple markk"
14,"apple markk"
15,"apple markk"
16,"apple markk"
17,"apple markk"
18,"apple markk stinkypig"
19,"apple"
20,"apple"
21,"apple"
22,"apple"
23,"apple"
24,"apple"
```

owners.csv

```
class Reader

 def read_in_songs(csv_file_name)
   songs = []
   CSV.foreach(csv_file_name, :headers => true) do |row|
      songname, artist, album, time, id = row[0],row[1], row[2], row[3], row[4]
      unless (songname =~ /#/)
          songs << Song.new(songname,album,artist,time.to_f,nil,id)
      end
    end
   songs
 end


 def read_in_ownership(csv_file_name, temp_hash = Hash.new)
   CSV.foreach(csv_file_name, :headers => true) do |row|
       song_id, owner_data = row[0], row[1]
       unless (song_id =~ /#/)
              temp_hash[song_id] = owner_data
           end
    end
    temp_hash
 end
end
```

reader.rb

```
# READER
# Copyright Mark Keane, All Rights Reserved, 2014

class Reader
 #read in the songs
 def read_in_songs(csv_file_name)
   songs = []
   CSV.foreach(csv_file_name, :headers => true) do |row|
      songname, artist, album, time, id = row[0],row[1], row[2], row[3], row[4
      unless (songname =~ /#/)
          songs << Song.new(songname,album,artist,time.to_f, nil, id)
      end
    end
   songs
end

#read in the owners.csv file
def read_in_ownership(csv_file_name, temp_hash = Hash.new)
  CSV.foreach(csv_file_name, :headers => true) do |row|
      song_id, owner_data = row[0], row[1]
      unless (song_id =~ /#/)
            temp_hash[song_id] = owner_data
          end
   end
   temp_hash
end
```

recall csv i/o

create song inst

return song array

will give us hash

reg exp

return hash map

reader.rb

Part E:

some issues arising...

Issues I: I/O Interface

- in reading in from files, most things are strings (and may have control characters); e.g., “2”
- So, we need to be careful to convert “2” to 2; using e.g., to_i or to_f
- chomp takes care of control characters at the end
- need to be careful that things are the type we want, should do it as they enter the program

Issues II: Objects or Names

- similarly, we are often juggling objects and the names of those objects; Util.fetch(“pony”) => <#song pony> or storing arrays of objs or arrays of ids
- this becomes a major problem within the program; as we have to check entering each method whether we have a string-name or an obj
- best, to avoid this and use objects from the top level
- then, we are doing true OOP, letting the methods do the type-checking

Issues III: Finding Objects

- we have used Global variables: **$songs $albums...**
- We could have used **@@songs** *class variable* but both probably should be avoided
- but, we will do it to see how such variables work

Phew !