PENDEKAR RUBY ON RAILS

NITZA ALFINAS

Acknowledgements

Pertama saya berterimakasih bla... bla... bla...

Errata

Feedback

Translations

Jika ada yang berminat untuk meng-alih bahasakan buku ini ke bahasa selain Bahasa Indonesia, silahkan kirimkan permintaan ke itsme@nitzaalfinas.com

---

If you are interested to translate this book to a language other than Indonesian, please send a request to itsme@nitzaalfinas.com

Table of Contents

Acknowledgements....2
Errata....3
Feedback....4
Translations....5
Table of Contents....6
1. Pendahuluan....10
  1.1. Apa itu Ruby on Rails?....10
    1.1.1. Ruby....10
    1.1.2. Rails....11
  1.2. Sejarah Rails....11
  1.3. Kenapa Menggunakan Ruby on Rails....12
  1.4. Aplikasi/Website Apa Saja yang Menggunakan Rails....13
  1.5. Instalasi Ruby on Rails....14
    1.5.1. Instalasi Ruby on Rails pada Ubuntu 14.04....14
    1.5.2. Instalasi Ruby on Rails pada Windows....15
    1.5.3. Instalasi Ruby on Rails pada Mac....20
2. Ruby....22
  2.1. Kenapa Harus Belajar Ruby Terlebih Dahulu....22
  2.2. Fitur Ruby....22
  2.3. Ruby Editor....23
  2.5. Interactive Ruby Shell....24
  2.6. Ruby Syntax....26
  2.7. Reserved Words....27
  2.8. Memberikan Comment pada Ruby....28
  2.9. Tipe Data....28
    2.9.1. String....28
    2.9.2. Numbers....29
    2.9.3. Symbol....30
    2.9.4. Arrays....30
    2.9.5. Hashes....31
  2.10. Variables....32
    2.10.1. Global....32
    2.10.2. Instance....33
    2.10.3. Local....34
    2.10.4. Class Variables....35
    2.10.5. Constants....35
  2.11. Operators....36
    2.11.1. Operator Aritmatik....36
    2.11.2. Operator Perbandingan....37
    2.11.3. Operator Pemberian Nilai....37
    2.11.4. Operator Pemberian Nilai Paralel....38
    2.11.5. Operator Logika....38
    2.11.6. Operator Tenary....38

2.11.7. Oprator Range....39
2.11.8. Operator defined?....39
2.11.9. Operator “.” dan “::”....40
2.12. Blok dan Perulangan....40
2.13. Methods....42
2.14. Class dan Object....43
2.14.1. Class....43
2.14.2. Objek....44
3. Rails....46
3.1. Rails Solusi Startup Sampai dengan Enterprise....46
3.2. Membuat Project Rails....47
3.3. Struktur File/Folder....48
3.4. MVC Pattern....51
3.4.1. Models....53
3.4.1.1. Active Record....53
3.4.1.2. Active Redord sebagai ORM Framework....53
3.4.1.3. Aturan dan Konfigurasi pada Active Record....54
3.4.1.4. Active Record Query Interface....56
3.4.2. Views....59
3.4.2.1. Layouts....59
3.4.2.2. Render secara Default....59
3.4.2.3. Menggunakan Render....61
3.4.2.3.1. Render Nothing....61
3.4.2.3.2. Render Action View....61
3.4.3. Controllers....62
3.4.3.1. Route....62
3.4.3.1.1. Menghubungkan URL dengan Controller dan Method....62
3.4.3.1.2. Resource Routing....62
3.4.3.1.3. Membuat Multiple Resource....63
3.4.3.1.4. Singular Resources....64
3.4.3.2. Aturan Pemberian Nama Controller....65
3.4.3.3. Methods dan Actions....65
3.5. Koneksi dengan Database....66
4. Membuat Aplikasi dalam 1 Menit....69
4.1. Buat Project....69
4.2. Jalankan Server....69
4.3. Buka Browser....70
4.4. Membuat CRUD dengan Perintah Scaffold....70
4.5. Migrate....72
4.6. Halaman Aplikasi....73
4.7. Mengganti URL Default dari Scaffold....74
4.8. Mengganti Tampilan Default....76
5. Membuat CRUD Manual....78
5.1. Alur Kerja....78
5.2. Model....79
5.3. Migrate....79

5.4. Route....80
5.5. Controller....80
5.6. Action Index dan View....80
5.7. Action Form Insert dan View....82
5.8. Partial View Form....83
5.9. Action Insert Data....84
5.10. Action Form Update dan View....85
5.11. Action Hapus....86
6. Membuat Blog....88
6.1. Gems....88
6.2. Menjalankan Web Server....90
6.3. Mengatur Database....91
6.4. Membuat Article Scaffold....92
6.5. Menjalankan Migrate....92
6.6. Mengganti URL....92
6.7. Memasukkan TwitterBootstrap....92
6.8. Mengganti Layout Default....94
6.9. Menyesuaikan Tampilan dengan TwitterBootstrap....96
6.9.1. Index....96
6.9.2. Form....97
6.10. Autentikasi Pengguna Menggunakan Devise....98
6.11. Membatasi Akses Pengunjung ke Halaman Admin....102
6.12. Membuat Halaman Admin untuk Users....103
6.13. Relasi Antar Tabel....105
6.14. Keluar dari Halaman Autentikasi....108
6.15. Membuat Halaman Landing....108
6.16. Membuat Partial View untuk Memudahkan Mengelola Kode....109
6.17. Pagination....113
7. Snippets....115
7.1. Commandline....115
7.1.1. Membuat Project Baru....115
7.1.2. Menjalankan Server....116
7.1.3. Generate dan Destroy....116
7.1.3.1. Generate Scaffold....116
7.1.3.2. Destroy Scaffold....117
7.1.3.3. Generate Controller....118
7.1.3.4. Destroy Controller....119
7.1.3.5. Generate Model....119
7.1.3.6. Rollback dan Destroy Model....120
7.2. Menambahkan Custom Action pada RESTful....121
7.3. Form....122
7.4. Form Ajax....124
7.5. Validasi Menggunakan ActiveRecord....125
7.5.1. presence....125
7.5.2. validates_associated....126
7.5.3. length....126

7.5.4. uniqueness....127
7.6. Upload File....127
7.7. Mengirim Email Menggunakan Action Mailer....130
7.8. Basic Authentication dengan Bcrypt....132
7.9. Mengganti URL Default pada Devise....135
7.11. Basic Pagination....136
7.12. Membuat aplikasi satu halaman....140
7.13. Pengenalan singkat Sass....140
7.13.1. Variables....140
7.13.2. Nesting....141
7.13.3. Partials....142
7.13.4. Import....142
7.13.5. Mixins....143
7.13.6. Extend....143
7.13.7. Operators....144
7.14. Pengenalan singkat CoffeeScript....145
7.14.1. Functions....145
7.14.2. Objects dan Arrays....146
7.14.3. Conditional....146
7.14.4. Array Slicing dan Splicing Menggunakan Range....146
7.15. Bekerja dengan Tabel Virtual pada Database....147
7.16. Melewatkan Token melalui Ajax dengan Cara Manual....150
7.17. Menjalankan Aplikasi Ruby, Bash/Shell Script, Java, PHP dan Lain-Lain....151
8. Server Produksi....153
9. Troubleshooting....156
9.1. WEBrick Web Server Tidak Berjalan....156
9.2. MySQL Gagal di Bundle pada Ubuntu....157
9.3. Error Ketika Penambahan Gem....157
9.4. Error 'Bundle Exec'....158

1. Pendahuluan

1.1. Apa itu Ruby on Rails?

Ruby on Rails tidak dapat dipisahkan dengan Ruby sehingga untuk mengetahui Ruby on Rails kita terlebih dahulu harus bahas tentang Ruby.

1.1.1. Ruby

Ruby adalah sebuah bahasa pemrograman yang dibuat oleh Yukihiro "Matz" Matsumoto.

Ruby adalah bahasa program yang singkat, *syntax* yang rapi, *syntax* yang tidak memerlukan banyak tanda baca yang berlebihan.

Yang sangat menarik dan kenapa kita juga menggunakan Ruby adalah pernyataan Matz's sebagai berikut;

"**Saya berbicara dengan teman tentang kemungkinan bahasa script menggunakan object-oriented. Saya menguasai Perl, tetapi saya tidak menyukainya karena terlihat seperti bahasa mainan. Object-oriented sepertinya sangat menjanjikan. Saya juga menguasai Python, tetapi saya juga tidak menyukainya karena saya fikir Python bukanlah bahasa object-oriented yang murni. Object-oriented pada Python terasa seperti fitur tambahan. Sebagai orang yang sangat suka programming dan object-oriented dalam 15 tahun ini, saya sangat ingin bahasa script object-oriented yang murni, mudah digunakan. Saya sudah mencari-carinya, saya tidak menemukannya sehingga saya membuatnya sendiri.**"[1]

Alasan tersebut juga merupakan salah satu alasan yang cukup kuat kenapa kita menggunakan Ruby sebagai bahasa pemrograman.

---

1 Dikutip dan diterjemahkan dari http://thenewstack.io/ruby-a-programmers-best-friend/

1.1.2. Rails

Rails dikembangkan oleh David Heinemeier Hansson. Ruby on Rails adalah sebuah *framework* aplikasi berbasis web yang ditulis dengan bahasa Ruby. Ruby on Rails secara singkat disebut dengan Rails yang merupakan sebuah perpanjangan *library* dari bahasa Ruby.

Secara umum Ruby on Rails merupakan kumpulan kode-kode yang ditambahkan pada program Ruby. Secara teknis, kode-kode tersebut disebut sebagai paket yang lebih spesifik disebut RubyGems[2] yang ditambahkan melalui sebuah perintah pada *shell*.

Ruby on Rails juga dikenal dengan *full stack framework* yang menekankan pola pengembangan software, pengaturan konfigurasi yang sederhana (*Convention over configuration*/COC), jangan ulangi kode (*Don't Repeat Your Self*/DRY), *Active Record* dan *model-view-controller* (MVC).

1.2. Sejarah Rails

David Heinemeier Hansson mengembangkan Rails pada Basecamp, sebuah manajemen proyek yang sekarang telah menjadi sebuah perusahaan pembuat aplikasi web. Hansson pertama kali merilis Rails pada bulan Juli 2004 sebagai *open source*, tetapi Hasson tidak membagikan *code*nya walaupun bersifat *open* sampai bulan Februari tahun 2005. Pada bulan Agustus 2006, *framework* ini membuat pencapaian yang bagus ketika Apple mengumumkan bahwa mereka akan menyertakan Ruby on Rails dengan Mac OS X v10.5 "Leopard", yang dirilis pada bulan Oktober 2007.

Rails versi 2.3 dirilis pada tanggal 15 Maret 2009 dengan pengembangan utama pada *template*, *engine*, *rack* dan *model form*. *Template* tersebut memungkinkan pengembang untuk menghasilkan aplikasi dengan konfigurasi yang sederhana dan *custom* Gems. *Engine* atau versi rilis aplikasi tersebut memberikan kemudahan bagi *developer* membuat aplikasi dengan *code* yang dapat digunakan kembali dengan menggunakan bantuan *route*, *view* dan *model*.

Pada tanggal 23 Desember 2008, Merb, *framework* untuk aplikasi web lainnya diluncurkan, dan Ruby on Rails mengumumkan akan bekerja sama dengan Merb untuk mengembangkan *framework* menjadi

2 RubyGems adalah sebuah paket manager untuk bahasa program Ruby untuk didistribusikan pada program Ruby dan Ruby *library* (secara satuan hanya disebut sebagai gem). RubyGems juga merupakan sebuah *tools* yang digunakan untuk mengelola dan mendistribusikan gems.

lebih baik dengan rilis awal Rails 3. Pada Rails 3 inilah pertama kali diperkenalkan istilay DRY agar *developer* tidak menulis *code* yang berulang dan ini menjadikan Rails 3 sangat efektif dalam pengembangan *software*. Merb dan Rails selanjutnya bergabung sebagai bagian dari Rails 3.0.

Rails 3.1 dirilis pada 31 Agustus 2011, menampilkan *Migrasi database Reversible*, *Asset Pipeline*, *Streaming*, jQuery sebagai *library* JavaScript *default* dan juga memperkenalkan CoffeeScript dan Sass ke dalam *framework stack*.

Rails 3.2 dirilis pada tanggal 20 Januari 2012 yang memungkinkan *developer* men*develop* aplikasi dengan cepat dengan *engine routing* yang juga dikenal sebagai *Journey engine*, *Automatic Query Explain* dan *Tagged Logging*. Rails 3.2.x adalah versi terakhir yang mendukung Ruby 1.8.7 dan Rails 3.2.12 mendukung Ruby 2.0.

Rails 4.0 dirilis pada 25 Juni 2013 dengan memperkenalkan *Rusian Doll Caching*, *Turbolinks*, *Live Streaming* serta *Active Resource*, *Active Record Observer* dan komponen lainnya yang dapat dimasukkan kedalam Gems.

Rails 4.1 dirilis pada tanggal 8 April 2014 dengan memperkenalkan *Spring*, *Variants*, *Enum*, *Mailer preview*, dan **secrets.yml**.

Rails 4.2 dirilis pada tanggal 19 Desember 2014 dengan memperkenalkan *Active Job*, *asynchronous emails*, *Adequate Record*, *Web Console* dan *foreign keys*.

1.3. Kenapa Menggunakan Ruby on Rails

Banyak developer Rails sebelumnya menggunakan bahasa PHP, Java dan .NET dan sangat enjoy ketika berpidah kepada Rails. Selain alasan kenapa memilih Ruby seperti yang dikatakan Martz pada sub-bab diatas, alasan lainnya kenapa menggunakan framework Ruby on Rails adalah;

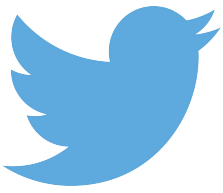
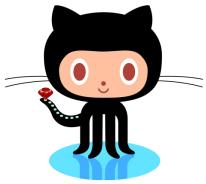
- Membuat aplikasi menggunakan Ruby on Rails sangat cepat bila dibandingkan dengan *framework* lainnya. Kecepatan ini didapat dari perpaduan Ruby yang merupakan bahasa objek murni dengan Rails sebagai *framework* dengan konsep MVC yang bagus.
- Aturan-aturan dalam Ruby on Rails memungkinkan *developer* dapat saling mengerti ketika

bekerja dalam sebuah tim maupun pergantian anggota tim untuk project yang sama.

- Kode yang singkat dan terstruktur dengan baik untuk dan aplikasi yang *reliable*.
- Ruby adalah bahasa *object-oriented* yang dinamis. Dengan perpaduan *framework* Rails, ini dapat komunikasi aplikasi dengan *database*, membuat *sitem template*, menangani *layout*, menangani ajax dan implementasi *plugin* yang sangat banyak dan mudah. Dengan kata lain, Rails mengeliminasi bagian yang membosankan pada software *development*.
- Ruby on Rails berkembang sangat pesat dan ini sangat memudahkan mengakomodasi perubahan kearah yang lebih sempurna.
- Kode yang ditulis dalam bahasa Ruby sangat mudah dibaca dan dapat didokumentasikan dengan mudah. Konvensi atau aturan penulisan *code* sangat jelas sehingga hal ini meningkatkan produktivitas walaupun terjadi perubahan struktur *developer*.
- Ruby on Rails mempunyai lingkungan *test* dan *framework* pengujian tersendiri.
- Library pada Ruby on Rails sebagian besar adalah *open source* sehingga biaya lisensi dapat ditekan.
- Selain itu, hal yang mempengaruhi kecepatan dalam pengembangan dengan Ruby on Rails adalah komunitas yang besar dan *library* yang terus bertambah setiap harinya.

1.4. Aplikasi/Website Apa Saja yang Menggunakan Rails

Berikut adalah beberapa aplikasi/website yang menggunakan Rails;

Twitter	Github	Bloomberg	SlideShare	SoundCloud
				

AirBNB 	Heroku 	Shopify 	IndieGogo 	Zendesk 

Selain website diatas, menurut wapplizer.com, terdapat 35.758 website yang dipastikan menggunakan Ruby on Rails dan sebanyak 8.061.564 terindikasi menggunakan Ruby on Rails.

1.5. Instalasi Ruby on Rails

Seperti telah kita ketahui pada sub bab diatas, Ruby adalah bahasa *script* dan Rails adalah *framework* untuk Ruby.

Proses instalasi Ruby dan Rails dapat saja dipisahkan pada Bab Ruby dan Bab Rails tetapi ini sangat tidak efektif dan membuang waktu. Untuk itu pada sub-bab ini akan dibahas cara menginstall Ruby on Rails pada Ubuntu 14.04, Windows dan Mac.

1.5.1. Instalasi Ruby on Rails pada Ubuntu 14.04

Cara yang paling cepat dalam menginstal Ruby on Rails adalah dengan menggunakan Ruby Version Manager (RVM).

1. Paket RVM dikelola oleh Michal Papis. Untuk memulai menginstall RVM, terlebih dahulu kita harus meng*install* mpapis *public key* dengan *command* berikut;

2. Ruby

2.1. Kenapa Harus Belajar Ruby Terlebih Dahulu

Pada *framework* Ruby on Rails, bahasa yang digunakan adalah bahasa Ruby sehingga sebelum menggunakan *framework* Ruby on Rails terlebih dahulu kita harus mengenal Ruby.

Berdasarkan pengalaman saya sebagai *developer*, menguasai hal dasar akan memberikan pemahaman yang sangat baik untuk proses belajar selanjutnya. Sehingga Bab ini harus Anda kuasai terlebih dahulu sebelum beranjak pada Bab berikutnya.

2.2. Fitur Ruby

Ruby adalah bahasa *object oriented* murni. Ruby mempunyai fitur yang hampir sama dengan Smalltalk, Perl dan Python.

Berikut adalah fitur-fitur utama yang dimiliki oleh Ruby;

1. Ruby adalah *open source*,
2. Ruby adalah bahasa script yang memerlukan interpreter,
3. Ruby adalah bahasa *object-oriented*,
4. Ruby adalah bahasa pada sisi server atau lebih dikenal dengan *server side programming*,
5. Ruby dapat digunakan untuk menulis *Common Gateway Interface* (CGI),
6. Ruby dapat disisipkan kedalam *Hypertext Markup Langage* (HTML),
7. Ruby mempunyai *syntax* yang ringkas dan mudah dimengerti,
8. Ruby mempunyai *syntax* yang hampir sama dengan bahasa program lain seperti C++ dan Perl,

9. Ruby dapat digunakan untuk aplikasi berskala besar atau kecil. Aplikasi berskala besar yang dibuat dengan Ruby dapat dikelola dengan mudah karena kode sudah terorganisir menjadi *class* dan *object*.
10. Ruby dapat digunakan untuk mengerjakan aplikasi internet atau intranet,
11. Ruby dapat diinstal pada POSIX dan Windows,
12. Ruby mendukung GUI tools; Tcl/Tk, GTK, dan OpenGL,
13. Ruby dapat dengan mudah dikoneksikan dengan DB2, MySQL, Oracle, Sybase dan SQLite,
14. Ruby mempunyai banyak fungsi *built-in* yang dapat langsung digunakan pada *script* Ruby.

2.3. Ruby Editor

Ruby adalah bahasa pemrograman yang populer dan popularitasnya terus menanjak sampai saat ini. Banyak *Integrated Development Environment* (IDE) yang mendukung penulisan kode Ruby pada berbagai macam sistem operasi.

Berikut adalah beberapa IDE untuk menulis kode Ruby berdasarkan sistem operasi;

Pada Linux:

- Aptana Studio
- Emacs with Ruby mode and Rsense
- Geany
- Gedit
- Vim with vim-ruby plugin and Rsense
- RubyMine
- SciTe

- NetBeans
- Sublime Text

Pada Windows:

1. Notepad++
2. E-TextEditor
3. Ruby In Steel
4. Netbeans

Pada Mac OS X:

1. TextMate
2. TextWrangler

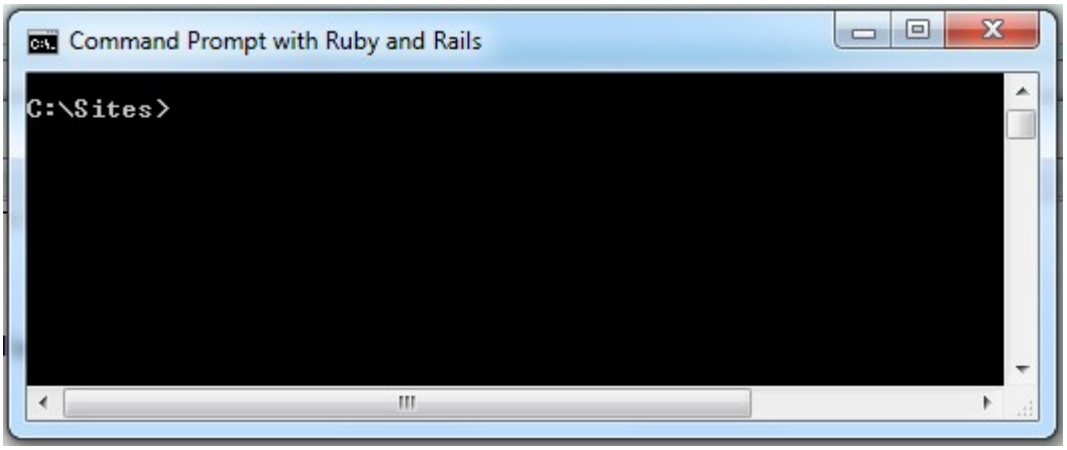
2.5. Interactive Ruby Shell

Interactive Ruby Shell atau biasa disebut dengan irb adalah sebuah *Read Eval Print Loop* (REPL) untuk pada *object oriented script* Ruby. Dengan menggunakan *shell* irb, kita dapat melihat hasil secara cepat.

Shell ini tersedia langsung ketika menginstal Ruby sehingga kita tidak memerlukan instalasi tambahan.

Untuk menjalankan irb pada sistem operasi unix, cukup dengan terminal. Dan untuk menjalankan pada sistem operasi Windows lebih disarankan menggunakan *Command Prompt with Ruby on Rails*. *Command prompt* ini dapat dipanggil lewat menu Windows.

Command prompt pada sistem operasi Window terlihat seperti berikut;



Untuk penulisan lebih lanjut, saya akan lebih banyak menulis kata *terminal* untuk merujuk *command prompt* pada Windows. Sehingga jika saya menulis *terminal*, bagi yang menggunakan sistem operasi Windows silahkan menjalankan *Command Prompt with Ruby on Rails*.

Untuk melihat versi irb yang digunakan, ketik perintah berikut pada terminal;

```
nitza@ubuntu-RV:~$ irb -v
irb 0.9.6(09/06/30)
```

Untuk menjalankan *shell* irb, cukup dengan mengetikkan perintah irb.

Sekarang marilah kita buat sebuah aplikasi sederhana untuk lebih mengenal Ruby.

```
nitza@ubuntu-RV:~$ irb
irb(main):001:0> puts "hello world"
hello world
=> nil
irb(main):002:0> exit
nitza@ubuntu-RV:~$
```

Untuk keluar dari irb *shell*, kita bisa mengetikkan perintah `exit` seperti diatas.

2.6. Ruby Syntax

Pada Bab ini kita akan membuat *script* yang disimpan pada sebuah file dan dijalankan melalui irb. File Ruby mempunyai eksteksi .rb sehingga file-file yang akan kita buat akan diberi ekstensi .rb.

Sebelum kita mulai membuat Ruby *script*, terlebih dahulu kita harus melihat konfigurasi Ruby yang ada pada mesin kita dengan mengetik perintah berikut;

```
nitza@ubuntu-RV:~/RoRBook$ irb
irb(main):001:0> RbConfig.ruby
=> "/home/nitza/.rbenv/versions/2.1.4/bin/ruby"
irb(main):002:0>
```

Untuk sistem operasi Windows terlihat seperti berikut;

```
C:\Sites>irb
irb(main):001:0>RbConfig.ruby
=> "C:/RailsInstaller/Ruby2.1.0/bin/ruby.exe"
irb(main):002:0>
```

Command yang diketik tersebut adalah untuk mengetahui lokasi interpreter pada sistem operasi yang kita pakai. Dan sekarang kita tau lokasi interpreter Ruby untuk sistem operasi Unix yaitu pada **/home/nitza/.rbenv/versions/2.1.4/bin/ruby** dan untuk sistem operasi Windows adalah **C:/RailsInstaller/Ruby2.1.0/bin/ruby.exe**.

Lokasi tersebut berguna bagi aplikasi yang akan kita buat untuk diterjemahkan oleh interpreter yang tepat.

Sekarang kita akan melanjutkan membuat sebuah *script* sederhana yang diberi nama **01_app.rb**. Berikut adalah *script* yang ada didalamnya;

```
#!/home/nitza/.rbenv/versions/2.1.4/bin/ruby
puts "Hello world"
```

Untuk sistem operasi Windows tidak diharuskan menyertakan *interpreter path* pada baris pertama, tetapi jika disertakan aplikasi tidak akan menjalankannya. Jadi, untuk sistem operasi Windows

01_app.rb hanya berisi satu baris seperti berikut;[3]

```
puts "Hello world"
```

Untuk menjalankan aplikasi tersebut, lakukan perintah berikut;

```
nitza@ubuntu-RV:~/RoRBook$ ruby 01_app.rb
Hello world
nitza@ubuntu-RV:~/RoRBook$
```

Pada sistem operasi Windows terlihat seperti berikut;

```
C:\Sites>ruby 01_app.rb
Hello world

C:\Sites>
```

2.7. Reserved Words

Reserved words adalah kata-kata yang tidak dapat digunakan sebagai *identifier*; variabel, *function* atau label.

Berikut adalah kata-kata yang tidak dapat digunakan sebagai *identifier*;

BEGIN	END	alias	and	begin
break	case	class	def	defined?
Do	else	elsif	end	ensure
false	for	if	in	next
nill	not	or	redo	rescue
retry	return	self	then	true

3 Pada sistem operasi Windows, letakkanlah file **01_app.rb** didalam folder C:\Sites sehingga memudahkan untuk menjalankan perintah `ruby` tanpa diikuti dengan *file path*

3. Rails

Bab berikut lebih banyak berisi tentang dokumentasi Rails dan cara menggunakannya.

Dalam mempelajari aplikasi, *developer* mempunyai cara yang berbeda-beda. Ada yang lebih senang membuat aplikasi sederhana terlebih dahulu baru membaca dokumentasi untuk memberikan penjelasan lebih dari apa yang telah dibuat.

Buku ini juga telah melalui riset dengan diuji coba pada beberapa orang *developer*. Hasilnya sebagian besar *developer* menyatakan 'kurang nyambung' ketika membaca dokumentasi sehingga dia membuat aplikasi sederhana pada Bab 4 dan 5 terlebih dahulu dan kembali lagi pada Bab ini untuk mempelajari lebih jauh.

Jika Anda adalah *developer* yang suka cara membuat aplikasi sederhana terlebih dahulu, silahkan lewati Bab ini, teruskan pada Bab 4, Bab 5 dan kembali lagi untuk membaca Bab ini.

3.1. Rails Solusi Startup Sampai dengan Enterprise

Rails sangat populer dikalangan *startup* karena sangat cepat dalam *development* dan butuh dana pengembangan yang sedikit. Sedangkan untuk *Enterprise*, Rails mampu diskala menjadi aplikasi yang besar dan komplek karena *code*nya mudah dibaca dan mudah dikelola. Berikut adalah solusi yang ditawarkan Rails secara lebih rinci;

1. *Agility/Flexibility*

   Rails dengan *syntax* Ruby sangat ringkas dan mudah digunakan. Rails sangat fleksibel untuk dikembangkan dengan *operating system* apa saja, *tools* apa saja dan *library* yang dapat dibagi pakai walaupun dengan aplikasi lainnya. Cara ini tentu saja dapat menghemat waktu, tenaga dan uang.

2. Proses pengembangan yang sangat cepat

Dengan Rails, semua diatur pada *library* yang dapat dipanggil dan digunakan secara langsung sehingga proses *development* menjadi sangat cepat, terstruktur dan dapat di skalasi. Dengan hanya menggunakan Rails, *script* penunjang untuk aplikasi web seperti HTML, CSS dan Javascript telah digabungkan menjadi satu. Disinilah peran *full stack framework* untuk mempercepat proses pengembangan aplikasi.

3. Ringan dan cepat

   Pada aplikasi Rails, sebelum HTTP *response* diberikan kepada pengunjung web, terlebih dahulu *resource* tersebut di*compile*.

4. Keamanan

   Rails mempunyai *built-in security* dan solusi yang elegan untuk mengamankan *password* dan data.

5. Scalability

   Dalam *enterprise*, *scalability* sangat penting seiring dengan pertumbuhan bisnis. Ada beberapa pilihan *scalability* pada Rails yaitu horizontal dan vertikal. Skala horizontal seperti penambahan perangkat server, sementara skala vertikal yaitu dengan penambahan memori, disk atau penggabungan kedua-duanya. Sementara pada sisi software, *scalability* dapat dilakukan dengan cukup mudah karena peran bahasa berorientasi objek, *code* yang mudah dibaca, terstruktur dan rapi karena telah diatur sedemikian rupa melalui *convention*.

3.2. Membuat Project Rails

Setelah kita menginstal Rails pada komputer *local*, selanjutnya kita akan coba membuat sebuah *project* baru.

Project baru pada Rails dapat dibuat dengan perintah berikut;

```
$ rails new RoRCMS
```

Hasil dari perintah diatas terlihat seperti berikut;

```
nitza@ubuntu-RV:~/RoRBook$ rails new RoRCMS
      create
      create  README.rdoc
      create  Rakefile
      create  config.ru
      create  .gitignore
      create  Gemfile
      create  app
      [...]
Using sdoc 0.3.1
Installing spring 1.3.3
Using sqlite3 1.3.10
Using turbolinks 2.5.3
Installing uglifier 2.7.1
Installing web-console 2.1.2
Your bundle is complete!
Use `bundle show [gemname]` to see where a bundled gem is installed.
         run  bundle exec spring binstub --all
* bin/rake: spring inserted
* bin/rails: spring inserted
nitza@ubuntu-RV:~/RoRBook$
```

Baris setelah perintah tersebut memberikan kita *clue* apa yang dilakukan Rails.

Bagian yang diberi *high light* adalah tempat dimana kita bisa menyertakan *dependency library* pada Rails yaitu **Gemfile** dimana didalamnya terdapat library yang kita butuhkan untuk pengembangan aplikasi berbasis web seperti; HTML, CSS, JavaScript dan *library* lainnya. Dengan hanya satu baris perintah `bundle install`, kita langsung dapat fokus pada pengembangan aplikasi.

3.3. Struktur File/Folder

Dengan menjalankan perintah `rails new`, kita akan mendapatkan sebuah aplikasi lengkap dengan struktur file dan folder-nya.

Setiap aplikasi Rails akan mempunyai strutkrur file/folder yang sama.

4. Membuat Aplikasi dalam 1 Menit

Beberapa *developer partner* kerja saya sering mengeluh karena menjalankan/membuat aplikasi melalui *commandline*. Tetapi ketika mereka mengenal *commandline* lebih jauh, *commandline* sangat membantu produktivitas mereka dan mereka mulai suka dengan *commandline*.

Membuat aplikasi menggunakan Rails akan terasa lebih mudah menggukan bantuan *commandline*. Hampir semua *commandline* digunakan untuk *generate* aplikasi, mencari *dependecy* aplikasi, menempatkan *library* dan perintah otomatisasi lainnya hanya dengan satu baris perintah.

Aplikasi ini dapat dibuat dalam satu menit dengan mengikuti poin-poin penting terutama pada bagian *commandline* yang telah dituliskan pada pembahasan tanpa memperhatikan keterangan-keterangan tambahan yang ada.

Setelah beberapa kali membuat aplikasi, saya memastikan Anda juga tertarik dan langsung ingat *commandline* yang digunakan sehingga ini mempercepat dan memudahkan proses *developemnt*.

Silahkan ikuti langkah-langkah berikut untuk membuat aplikasi hanya dalam waktu 1 menit.

4.1. Buat Project

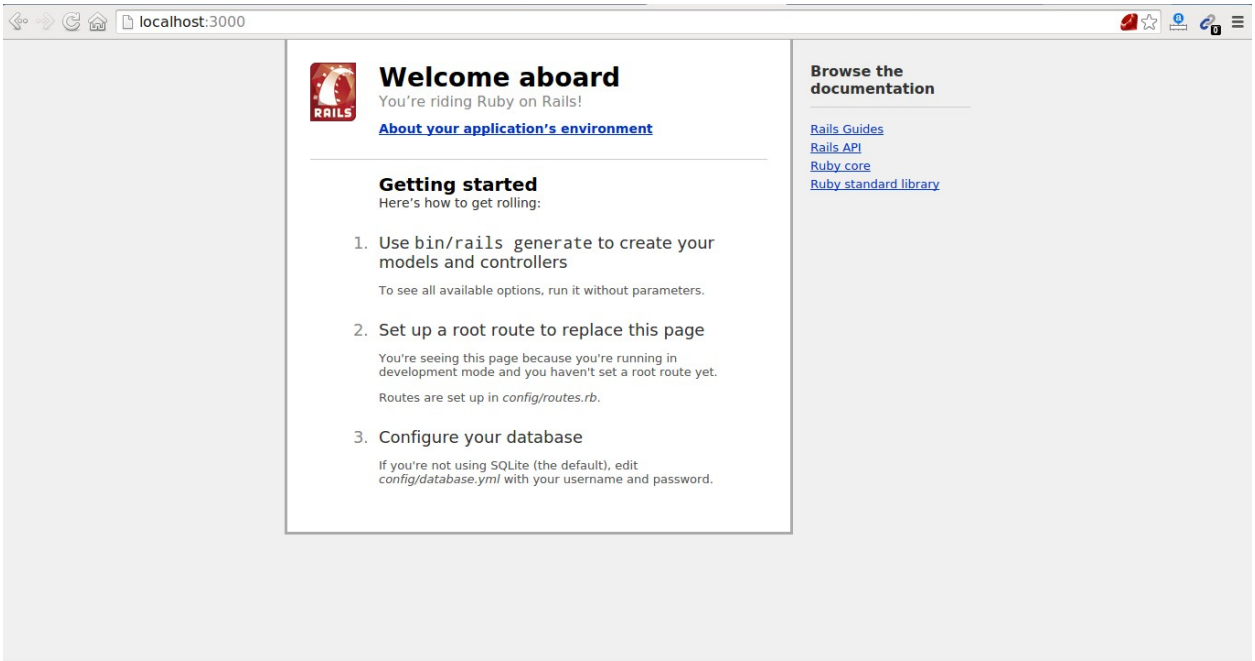
```
$ rails new scatwit
```

4.2. Jalankan Server

```
$ rails s
```

4.3. Buka Browser

Bukalah browser dan ketik URL: http://localhost:3000 dan terlihat gambar seperti berikut;



4.4. Membuat CRUD dengan Perintah Scaffold

5. Membuat CRUD Manual

Secara umum, aplikasi yang menggunakan *database* tidak lebih dari operasi memasukkan data, membaca data, mengupdate data dan menghapus data. Istilah tersebut sangat populer dengan Create Read Update and Delete (CRUD). Sehingga dikatakan jika seorang *developer* sudah menguasai CRUD, maka sebagian proses *development* sudah dikuasai.

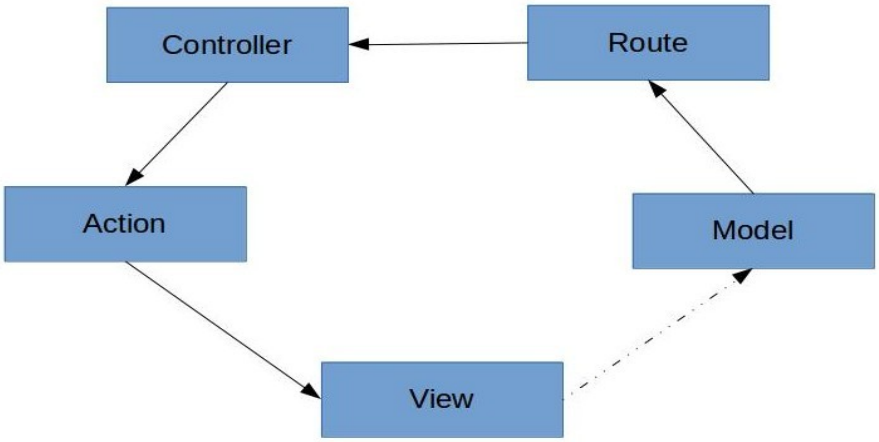
Mengetahui dasar pemrograman sama baiknya dengan kita mengetahui pembuatan aplikasi secara otomatis melalui *commandline*. Ini memberikan kita pemahaman lebih dan jika terjadi *troubleshooting* dalam *development*, maka kita tau cara mengatasinya.

Disini kita akan membahas tentang membuat CRUD secara manual untuk lebih mengetahui bagaimana sebenarnya alur kerja CRUD.

5.1. Alur Kerja

Untuk mempermudah pengerjaan CRUD secara manual, berikut adalah alur kerja yang mudah untuk diikuti. Alur ini tidak mengikat, tetapi bagus untuk dipraktekkan.

Berikut adalah gambar alur kerja CRUD pada Rails.



5.2. Model

Untuk membuat *model*, ketik perintah berikut;

```
$ rails g model Car name:string brand:string description:text
```

Perintah tersebut akan membuat file *migrate*, *model* dan *test*.

Jika kita lupa memasukkan beberapa *field*, kita masih dapat menambahkannya secara manual. Misalnya kita lupa menambahkan harga, untuk menambahkannya bukalah file **db/migrate/**********_create_cars.rb** dan tambahkan baris yang di *high light*;

```
class CreateCars < ActiveRecord::Migration
  def change
    create_table :cars do |t|
      t.string :name
      t.string :brand
      t.text :description
      t.integer :harga

      t.timestamps null: false
    end
  end
end
```

7. Snippets

Untuk meningkatkan produktivitas kerja, kita akan membahas *snippets* yang penting pada proses *development* Rails.

7.1. Commandline

Beberapa *commandline* yang penting dalam *development* aplikasi Rails adalah sebagai berikut;

- `rails new nama_project`
- `rails generate`
- `rails dbconsole`
- `rails server`
- `rails console`
- `rake`

Commandline tersebut secara detail akan dibahas pada sub bab berikut;

7.1.1. Membuat Project Baru

Untuk membuat *project* baru, format penulisannya adalah `rails new nama_project`.

Contoh penggunaannya seperti berikut;

```
$ rails new blog
```

7.1.2. Menjalankan Server

Rails *stack* mengikutsertakan web *server* WEBrick didalamnya sehingga proses *development* akan menjadi lebih cepat tanpa harus mengkonfigurasi *server*.

Untuk menjalankan *server*, ketik perintah berikut;

```
$ rails s
```

7.1.3. Generate dan Destroy

Untuk membuat segala sesuatu pada *Rails project*, umumnya menggunakan perintah `generate` atau secara singkat bisa langsung dituliskan dengan huruf `g` saja.

7.2. Menambahkan Custom Action pada RESTful

Terkadang *action* yang telah dibuat melalui perintah `scaffold` tidak sesuai keinginan kita. Ada kalanya kita kurangi atau ingin kita tambahkan beberapa *route* dan *action* agar sesuai dengan aplikasi yang kita buat.

Jika pada sub-bab terdahulu kita hanya mengganti *path* URL, maka pada sub-bab kali ini akan dibahas cara menambahkan *custom action* atau menambahkan *path* URL melalui `route` yang dibuat dengan perintah `scaffold`.

Sebagai contoh, kita mempunya RESTful *articles* yang telah dihasilkan melalui perintah `scaffold` dimana secara *default* hanya akan menghasilkan 8 *route* seperti berikut;

Helper	HTTP Verb	Path	Controller#Action
articles_path	GET	/articles(.:format)	articles#index
	POST	/articles(.:format)	articles#create
new_article_path	GET	/articles/new(.:format)	articles#new
edit_article_path	GET	/articles/:id/edit(.:format)	articles#edit
article_path	GET	/articles/:id(.:format)	articles#show
	PATCH	/articles/:id(.:format)	articles#update
	PUT	/articles/:id(.:format)	articles#update
	DELETE	/articles/:id(.:format)	articles#destroy

Dan dalam contoh ini, kita ingin menambahkan *path* **/articles/by_user** dan **/articles/by_category/category_id**.

Untuk menambahkan *path* ini, bukalah file **config/routes.rb** dan tambahkan baris yang diberi *high*

light;

```
Rails.application.routes.draw do
  resources :articles do
    collection do
      get 'by_user', to: 'articles#by_user'
      get 'by_category/:id', to: 'articles#by_category'
    end
  end
end
```

Dengan menambahkan beberap route tersebut, maka langkah selanjutnya adalah membuat *action* `by_user` dan `by_category` pada **app/views/articles_controller.rb** dan juga menambahkan *view* yang berhubungan dengan *action* tersebut pada folder **app/views**.

9. Troubleshooting

Proses *problem solving* untuk *term* sistem informasi mengacu kepada istilah *troubleshooting*. Dalam troubleshooting tentunya terlebih dahulu ada trouble atau error pada sistem atau aplikasi baik itu pada proses *development* atau pada proses *production*.

Error atau trouble pada proses *development* atau produksi adalah hal yang wajar. *Error* tersebut dapat dilihat melalui *commandline* langsung atau dapat dilihat melalui file log.

Troubleshooting pun pada setiap mesin dan lingkungan sistem operasi berbeda-beda. Maka dari itu troubleshooting secara umum yang ditemui pada proses development dan production Rails akan dibahas pada Bab ini.

9.1. WEBrick Web Server Tidak Berjalan

Jika kita menjalankan server dengan perintah `rails s` dan menemukan *error* seperti berikut;

```
nitza@nitza-Lenovo-G405s:~/blog$ rails s
/var/lib/gems/1.9.1/gems/execjs-2.5.2/lib/execjs/runtimes.rb:48:in `autodetect':
Could not find a JavaScript runtime. See https://github.com/rails/execjs for a
list of available runtimes. (ExecJS::RuntimeUnavailable)
[...]
```

Untuk memperbaikinya, bukalah Gemfile dan isikan baris berikut;

```
gem 'therubyracer', :platforms => :ruby
```

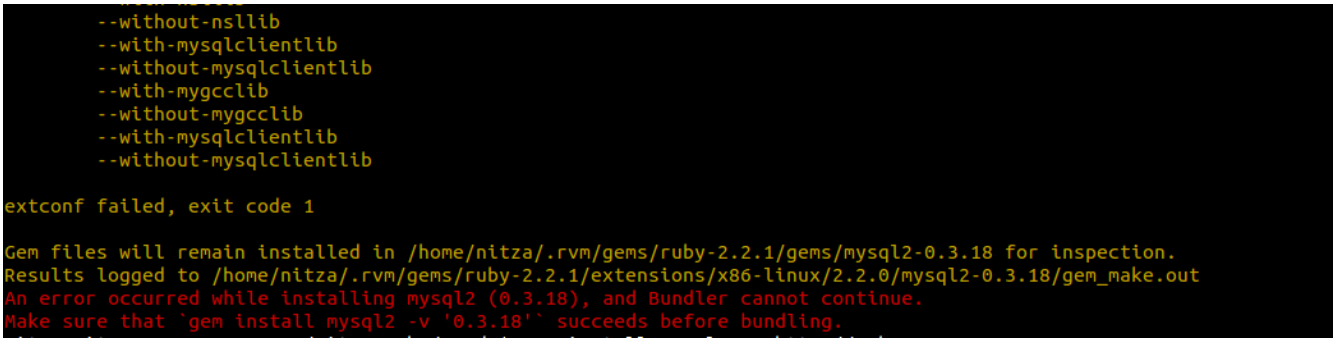
Setelah menambahkan, jalankan perintah;

```
$ bundle update
$ bundle install
```

Dan terkhir jalankan server kembali dengan perintah `rails s`.

9.2. MySQL Gagal di Bundle pada Ubuntu

Kasus seperti gambar berikut ini cukup banyak terjadi;



Kasus tersebut terjadi karena library untuk MySQL *client* belum ter*install*.

Untuk dapat menginstall MySQL *client*, ikuti langkah berikut;

```
$ sudo apt-get -f install

$ sudo apt-get update

$ sudo apt-get install libmysqlclient-dev
```

Setelah selesai meng*install*, jalankan kembali perintah `bundle install`.

9.3. Error Ketika Penambahan Gem

Jika terjadi error ketika menambahkan sebuah gem, lakukan langkah-langkah berikut;

1. Matikan server dengan kombinasi tombol ctrl + c

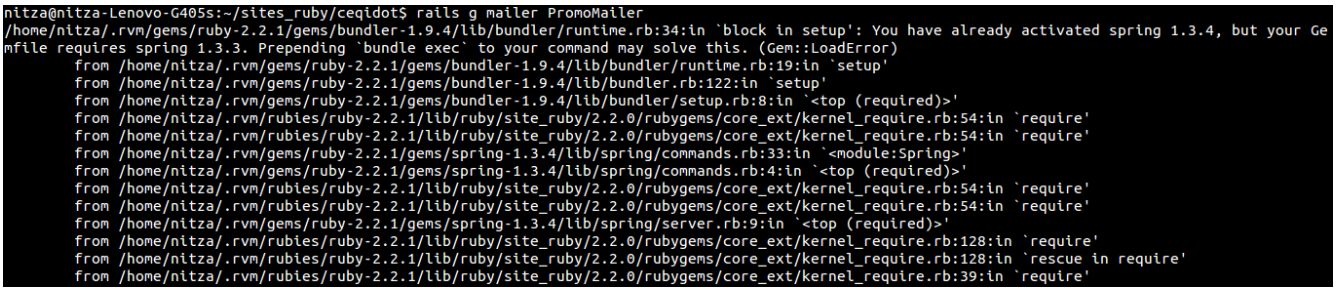
2. Jalankan `bundle install`

3. Jalankan server dengan perintah `rails s`

4. Jika masih terjadi error, ulangi langkah diatas tetapi dengan mengganti perintah pada poin dua yaitu `bundle update`

9.4. Error 'Bundle Exec'

Ketika *development*, terkadang muncul *error* seperti gambar berikut;



Untuk mengatasi *error* seperti terlihat diatas, jalankan `bundle update`;

```
$ bundle update
```