NUUG.topic = Ruby.new

Johannes Brodwall

Takk til Dave Thomas for bruk av enkelte slides

Hva er Ruby?

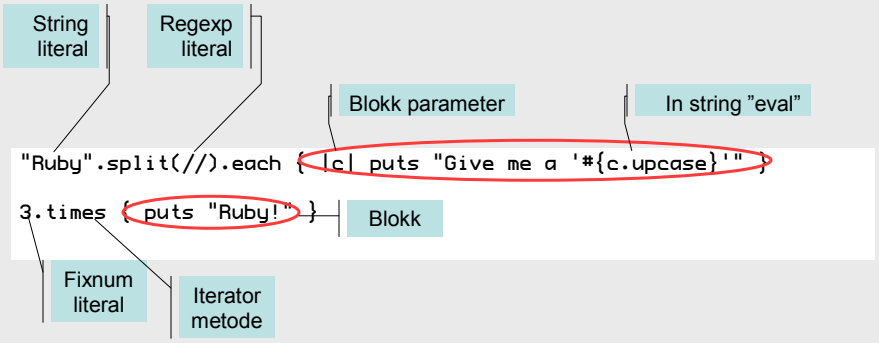
Et programmeringsspråk som er:

- Dynamisk typet
  - Dynamiske variable
  - Modifiserbare klasser
- Objekt-orientert
- Script-basert
- Transparant

- Morsomt

johannes@brodwall.no

La oss få bena våte!



Ruby som et bedre Perl

- ”ruby –e”, ”ruby –p”, ”ruby –n”
- Et enkelt ruby-filter
- Ruby-grep

```
#!/usr/bin/ruby -p
gsub ”perl”, ”ruby”
```

```
#!/usr/bin/ruby -n
BEGIN {
$regex=Regexp.new ARGV.shift
}

print if $regex.match($_)
```

Hvorfor jeg like Ruby 1: Perl-ismer

```
print while gets

while gets do
    gsub "perl", "ruby"
end

$_ =~ /[Rr]uby/ and print "#$`(((#$&)))#$'"
```

Konklusjon! `ruby -pe 'gsub "perl", "ruby"'`

Ruby som en bedre Java

```
public class HelloWorld {

   public static void main(String[] args) {
     System.out.println("hello world");
   }

}
```

```
puts "Hello world!"
```

Ruby som en bedre Java

```
public class MyComponent {
   private Color color;
   private int   xpos;
   private int   ypos;

   public Color getColor() {
     return this.color;
   }

   public void setColor(Color color){
     this.color = color;
   }

   public int getXPos() {
     return this.xpos;
   }

   public void setXPos(int xpos) {
     this.xpos = xpos;
   }

   public int getYPos() {

ZZZZZZzzzzzzzz.......
```

Ruby som en bedre Java

```
public class MyComponent {
   private Color color;
   private int   xpos;
   private int   ypos;

   public Color getColor() {
     return this.color;
   }

   public void setColor(Color color){
     this.color = color;
   }

   public int getXPos() {
     return this.xpos;
   }

   public void setXPos(int xpos) {
     this.xpos = xpos;
   }

   public int getYPos() {

ZZZZZZzzzzzzzz.......
```

```
class MyComponent
  attr_accessor :color
  attr_accessor :xpos
  attr_accessor :ypos

  def ypos=(new_pos)
    raise "Yo!" if new_pos > xpos
    @ypos = new_pos
  end

end


c = MyComponent.new
c.xpos = 4
c.ypos = 5   # will raise exception
c.ypos = c.xpos - 1
c.xpos += 2
```

Hva liker jeg ved Ruby?

- Enkelt å få interpreteren glad (i motsetning til Perl)
- Leser som pseudokode
- Super strenghåndtering
- Lambda all the way down
- Ren OO med godt metatype system

Strenger, Lister, hash

- `"hello", 'hello "this is quoted" hello'`
- `/literal reg[Ee]xp?/` (ingen escaping nødvendig)
- `a = [1, 2, 3, 4]`
  - `a[0] => 1`
  - `a[1..3] => [2,3,4]`
  - `a[-2..-1] => [3,4]`
  - `a[1,2] => [1, 2]`
- `h = { 'a'=>1, 1=>Color.new, [1,2] => "a" }`
  - `h['a'] => 1`
  - `h["dgnslgn"] => nil`

Løkker, alternativer

- `for i in 1..3 do ... end`
- `(1..3).each { |i| ... }`
- `i=0; until i==3 do i += 1; ... ; end`

- `if ( ... ) { print }`
- `print if ...`
- `... and print`
- `print unless ...`

Hvorfor jeg liker Ruby 2: Irb, irb, irb

```
irb(main):001:0> m = Hash.new
=> {}
irb(main):002:0> m = { 'a' => 'b', 'r' => [1,2,3] }
=> {"a"=>"b", "r"=>[1, 2, 3]}
irb(main):003:0> m['a']
=> "b"
irb(main):004:0> m['b']
=> nil
irb(main):005:0> m['r']
=> [1, 2, 3]
irb(main):006:0> m['c'] = 15
=> 15
irb(main):007:0> m
=> {"a"=>"b", "c"=>15, "r"=>[1, 2, 3]}
irb(main):008:0> m['a'] = "hello world"
=> "hello world"
irb(main):009:0> m
=> {"a"=>"hello world", "c"=>15, "r"=>[1, 2, 3]}
irb(main):010:0> m.delete("r")
=> [1, 2, 3]
irb(main):011:0> m
=> {"a"=>"hello world", "c"=>15}
irb(main):012:0>
```

Hvorfor jeg liker Ruby 3: Blokker

- Blokker er objekter
  - Fullstendige medlemmer av språket
- Kan evalueres nå, eller senere
- Bruk:
  - Iterasjon
  - Callbacks
  - Ressurshåntering

- Eksempler fra Dave Thomas's Ruby presentasjon på rOOTs 2002 (used with permission)

Blocks and Iterators

- Blocks and iterators are pervasive in Ruby

```
3.times { puts "Ho!" }

hash.each { |key, value|
 puts "#{key} -> #{value}"
}


IO.foreach("/etc/passwd") do |line|
    process(line)
end
```

Blocks and Iterators

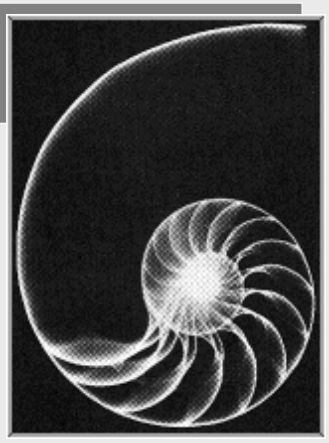
- Method calls block using "yield"

```
def example
  yield "one"
  yield "two"
end

example { |txt|  puts txt }

=> one
   two
```


Iterators



- Method that calls a block *n* times
- Passes the block zero or more parameters

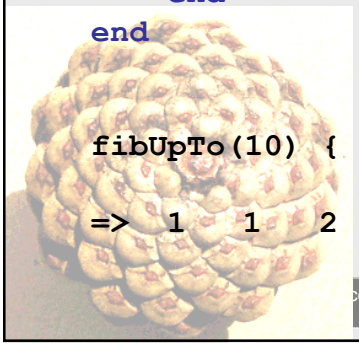
```
fibUpTo(10) { |n|  print n }
=>
    1   1   2   3   5   8
```

Iterators

```
def fibUpTo(max)
    n1 = n2 = 1
    while n1 <= max
        yield n1
        n1, n2 = n2, n1+n2
    end
end

fibUpTo(10) { |n|  puts n }

=>  1   1   2   3   5   8
```



Blocks Everywhere

```
DBI.connect("DBI:Pg:my_db") do |db|
  db.transaction do
      db.execute("SELECT ...") do |stmt|
          stmt.each do |row|
              # Process Row
          end
      end
  end
end
```

Classes and Accessors

```
class Song
  def initialize(aTitle)
    @title = aTitle
  end
  attr_reader   :title
  attr_accessor :artist
end
```

```
def artist
  @artist
end

def artist=(val)
  @artist = val
end
```

```
aSong = Song.new("As Time Goes By")
aSong.title                    # => "As Time..."
aSong.artist = "Sam"
```


Hvorfor jeg liker Ruby 4: Metakonstruksjoner

```
def Module.once(id)
  module_eval <<-"end_eval"
   alias_method :__#{id.to_i}__, #{id.inspect}

   def #{id.id2name}(*args, &block)
      @#{id.to_i} = __#{id.to_i}__(*args, &block)
      def self.#{id.id2name}(*args, &block)
         @#{id.to_i}
      end
      @#{id.to_i}
    end
  end_eval
end

def columns
  # expensive DB-lookup
end

once :columns
```

Hide old "columns" for later

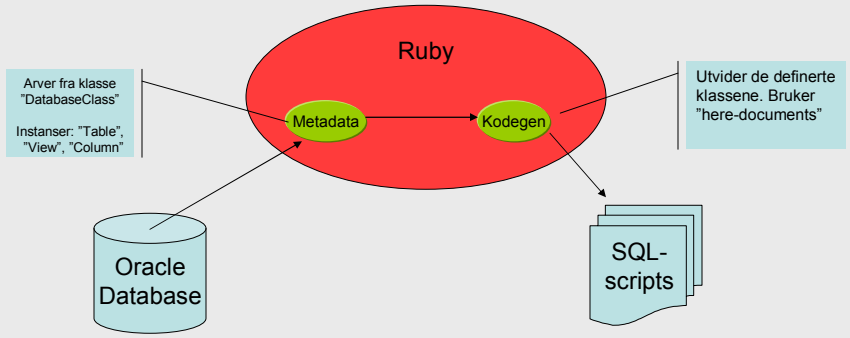
Redefine "columns"

Set @columns to result of original "columns" call

Return member @columns

Redefine "columns" again(!)

Utvidet eksempel: Kodegenerator



Eksempel databaseklasse

```
class Schema < DatabaseObject
    obj_table_name "ALL_USERS"
    make_key :owner

    collection :tables, :Table
    collection :views, :View
    collection :triggers, :Trigge
    collection :procedures, :Procedure
    collection_where :sequences,
      :Sequence, [ "SEQUENCE_OWNER"
end
```

Table to map to

Creates attribute, used for SQL generation

Creates lazy-loaded attribute that reads from the database using the definitions in class Table

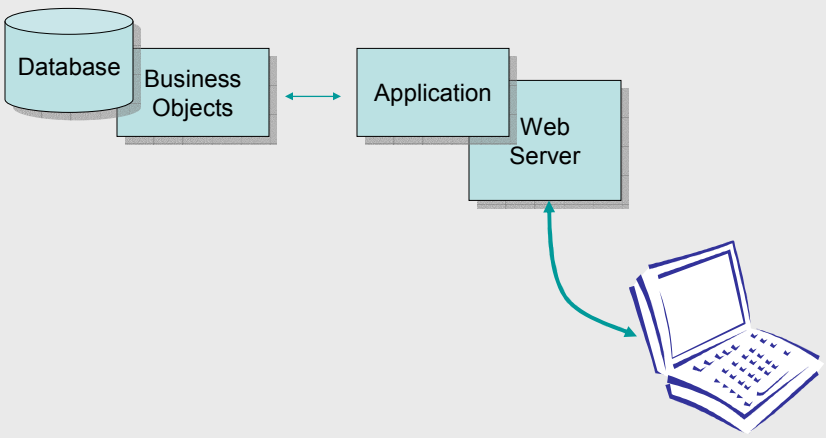
Lazy-loaded collection attribute using SQL

Needed because ALL_USER has column "OWNER", "ALL_SEQUENCES" has column "SEQUENCE_OWNER"

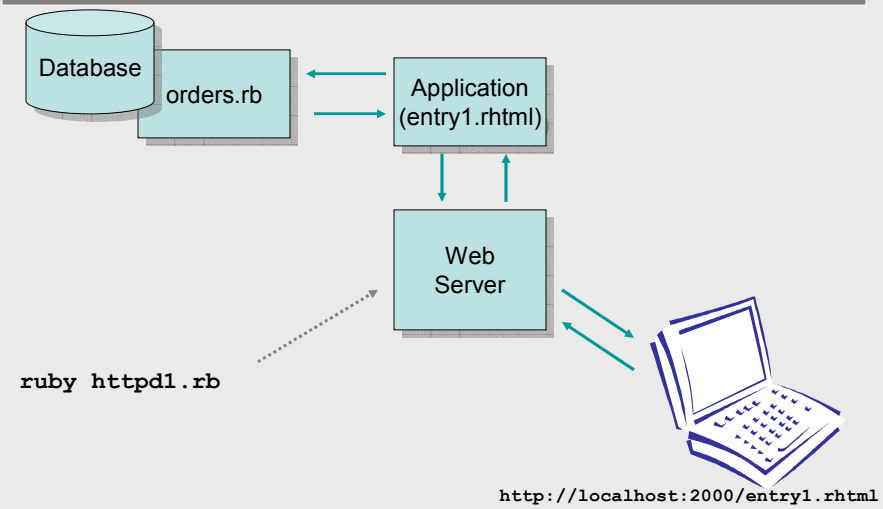
Utvidet eksempel 2: 3-lags app

- Database: MySQL
- Forretningsobjekt
- Dynamisk HTML-side: eruby eller amrita
- Ruby HTTPD: WEBRick

1½ Tier Architecture



Run the Application



Hvorfor jeg liker Ruby 6: Helt opp, helt ned

- Starter like enkelt som Perl
- Kan bli mer avansert enn C++, Java, C#

Innføring av Ruby

- Uten å komme i politiske situasjoner
  - Kodegenerering
  - Domenespråk
    - Build-verktøy – Rake
    - Mailfilter – Gurgitate-mail
  - Testing
  - Tekstredigering a-la Perl
- Dersom du har full kontroll
  - Lim-logikk
  - Web-applikasjoner
  - Hele programmet!

Links

- http://www.ruby-lang.com
- http://www.ruby.no
- http://www.ruby.no/prosjekt/RubyNuby/
- http://www.ruby.no/prosjekt/ProgrammeringIRuby/
- http://www.rubycentral.com
- http://onestepback.org/articles/usingruby/

Bonusmateriell

- Noen viktige punkter om Ruby
- Detaljer kodegenerering
- Ruby versus Python, Ruby og andre språk
- Make i Ruby: Rake
- Procmail i Ruby: gurgitate-mail

Dette må oxo med

- Ruby kan extendes med C
- Perl's regular expression support
- Mod_ruby
- $SAFE
- druby
- Ruby har et aktivt brukemiljø – men ikke nok brukere i Norge!

Eksempel databaseklasse

```
class Table < DatabaseObject
    obj_table_name "ALL_TABLES"
    make_key :owner, :table_name

    collection_sql :primary_keys, :Column,
           "SELECT owner, table_name, column_name FROM ALL_CONS_COLUMNS " +
           "where owner = ? AND table_name = ? AND NOT POSITION IS NULL"
    collection :columns, :Column
    collection :constraints, :Constraint
    collection_where :triggers, :Trigger, ["table_owner", "table_name" ]

    def column_values
        columns.values.sort
    end

    def column_names
        column_values.collect { |column| column.column_name }
    end

    def pk_constraint
        constraints.values.find { |cons| cons.constraint_type == "P" }
    end
end
```

Eksempel kodegenklasse

```
class Table
    def createTable(table_name = self.table_name)
        "CREATE TABLE " + table_name + " (\n" +
            column_values.collect { |column| column.column_def }.join(",\n") +
            "\n);\n\n" +
            constraints.values.collect { |cons| cons.alter_constraint_def(table_name) + ";" }.join("\n")
    end
end


class Column
    def <=>(other)
        self.column_id.to_i <=> other.column_id.to_i
    end

    def column_def(include_nullable = true)
        column_name + " \t" + full_type + (nullable == "N" && include_nullable ? " \tNOT NULL" : "")
    end

    def full_type # long, boring method returning e.g. "NUMERIC(10,4)"
    end
end

class Constraint
    def alter_constraint_def(table_name = self.table_name, ref_postfix = "")
        return "ALTER TABLE " + table_name + " ADD " + constraint_def(ref_postfix)
    end
    def constraint_def
      # long, boring code returning e.g. "FOREIGN KEY (key) REFERENCES other_tab (key) ON DELETE CASCADE"
    end
end
```


Alternative kodegeneringsmåter

- Bruk here-dokumenter
- Bruk eruby (”Embedded Ruby” – JSP-aktig)

Ruby versus Python

Ruby

- 100% Ren OO
- Sann GC
- Blokker (versus Python lambda)

Filosofisk: Mest mulig frihet til programmereren (som Perl)

Python

- Konsistent formattering og bruk (one way to do it)
- Mer modent bibliotek
- Mer (engelsk) doko

Filosofisk: Konsistent og fornuftig

Ruby og andre språk

- Perl: Ruby er ”neste skritt”. Lik filosofi.
- SmallTalk: Ruby er en mer script-orientert SmallTalk.
- Ruby: SmallTalk’s semantikk i perl’s språkdrakt

Bonus: Rake – Ruby Make

```
# Example Rakefile -*- ruby -*-

task :default => [:main]

file "main.o" => ["main.c"] do |t|
  src = t.name.sub(/\.o$/, '.c')
  Sys.run "gcc #{src} -c -o #{t.name}"
end

OBJFILES = ["a.o", "b.o", "main.o"]
task :obj => OBJFILES

file "main" => OBJFILES do |t|
  Sys.run "gcc -o #{t.name} main.o a.o b.o"
end

task :clean do
  Sys.rm '*~', '*.o'
end

task :run => ["main"] do
  Sys.run "./main"
end
```


Bonus: Gurgitate-mail

```
if from =~ /ebay.com/ then save("=ebay"); return; end

if headers.matches(["To","Cc"],"webmaster@") then
   save("=webmaster")
   return
end

friendsfile=homedir+"/.friends"
if FileTest.readable?(friendsfile) then
  IO.foreach(friendsfile) do |friend|
    if from =~ friend.chomp then
      log "Mail from friend "+friend.chomp
      save("=friend")
      return
    end
  end
end
```