WEB+DBシステム（入門編）

第7回(2016年5月26日)

ruby言語

今日のテーマ

- Ruby言語の基礎を学ぶ。

- Object指向言語とは、どんな言語かを学ぶ。
- また、実際にruby言語を走らせてみる。

Rubyって何？

Rubyの原石!? →

http://ja.wikipedia.org/wiki/%E3%83%AB%E3%83%93%E3%83%BC

Ruby / Rails / Gem

- Ruby on Railsでは三本立ての構造
  - Ruby – 言語
  - Gem – Rubyを取り巻く様々なツール・環境の部品
    - rake – Ruby版のmakeツール
    - Javaではantが相当する。
  - Rails – 開発がスムーズに進むようにしてくれている支援ツール

Ruby言語とは

- 「スクリプト言語」とは？
  - コンパイラとインタプリタ
  - 中間言語(Java)
- 完全な「オブジェクト指向」
  - クラスとインスタンス
  - メッセージパッシング
  - Event Driven

プログラム言語の種類

- C, C++, C#, Java, Javascript
- Perl, PHP, Ruby, Python,
- LISP, Prolog, Fortran, COBOL

- などなど、数多くの言語が導入されている。
  - ⇒ YACC/LEXなどで、自分専用の言語も作れる
- なぜこんなにたくさんの言語がある？
  - 用途に応じて特化した: 「人工知能向け言語」
  - 環境に合わせて進化した: C⇒C++／C#

プログラムの実行動作

- プログラムは、CPUで実行される。
  - CPUの種類によって、機械語が異なる。
  - 最終的には、プログラムは機械語に翻訳され、「実行」される。
- 最初に「機械語」に翻訳して準備しておく場合
  - 「コンパイラ」によって、翻訳する。
  - エラーがあるとコンパイルできないが、実行は高速
- １行ずつ機械語にしながら実行する場合
  - 「インタープリタ」によって翻訳する。
  - エラーがあっても、その行までは実行できるが、遅い

コンパイルとアセンブリ言語

- 人間が読むことのできるコンピュータ言語を「高級言語」と呼ぶこともある。
  - コンピュータの中にあるCPUごとに定められた「機械語」に翻訳する前に、「アセンブリ言語」に変換される。
  - アセンブリ言語と機械語は、一対一
- 機械語になるとCPUが制約される。

```
P3       START  PROCEDU ;条件分岐
DATADIV  DS     0
KIJUN    DC     17
NENREI   DC     16
HANTEI   DC     #FFFF
PROCEDU  LD     GR2,=0
         LD     GR1,KIJUN
         CPA    GR1,NENREI
         JZE    ATARI
         LD     GR2,=1
ATARI    ST     GR2,HANTEI
         RET
         END
```

画像引用元：http://www.vector.co.jp/soft/win95/edu/se453553.html?ds

JavaとRubyの共通点

- Javaのソースコードは「中間コード」に変換される。
  - 「中間コード」を実行するのは、Java VM
  - VM:Virtual Machine（仮想マシン）
  - 中間コードに変換する作業をJavaコンパイラが担っている。
- Java ScriptはScript言語
  - RubyもJava Scriptと同じように、スクリプトとして実行される。

スクリプト言語

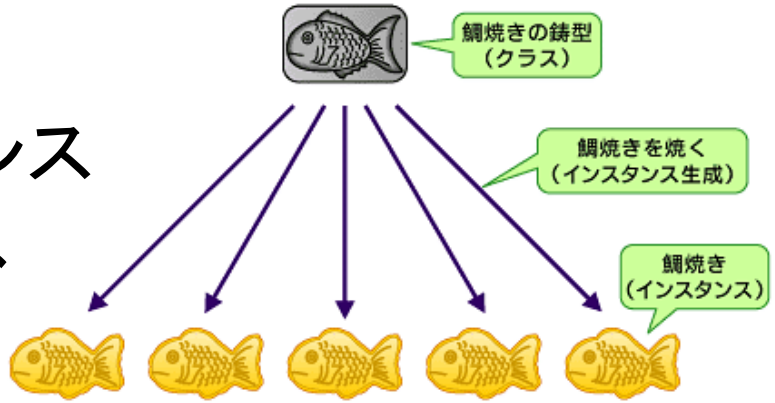
- コマンド入力を「まとめて行う」ために、スクリプト（台本）が記述されるようになった。
  - Windowsのバッチファイル（Batch Jobのための記述ファイル）も、使用目的は同じ
- このスクリプトが、HTML（WEB画面）の中に組み込まれて、簡単な「プログラム処理」ができるようになった。（Java script)
- こうした環境で使われる言語がscript（スクリプト）言語として、batchやShell Scriptよりも有名になり、スクリプト言語の主流になった。

オブジェクトとは何か？

- 「物」・・・ではなく、「対象」と考えて下さい。
- 処理プログラムを中心に考えるのではなく、処理の「対象物」であるデータを中心に考える、ということ
- 予め書かれているプログラムの流れの中に「データ」をはめ込んでいくのではなく、「データ」自体をそれぞれ「オブジェクト」として定義して、それぞれに、「特徴」や、「手続き」を持たせておく。

クラスとインスタンス

- クラスとは – 型
  - オブジェクト指向で、一つの「型」とか、ある集合の抽象的な「特徴」などを記述したもの
  - 例：「人」クラス
- インスタンスとは – 型からできた実体
  - クラスが実体化したもの
  - 例：「山田太郎」インスタンス
    （「人」が実体化すると、特定個人になる。）



クラスとインスタンス（2）

図1　クラスとインスタンス。一般的な“犬”が「クラス」で，“タロー”“ポチ”“マル”などの具体的な犬が「インスタンス」に相当すると説明されることがある

http://itpro.nikkeibp.co.jp/article/COLUMN/20060921/248617/

「完全な」オブジェクト指向

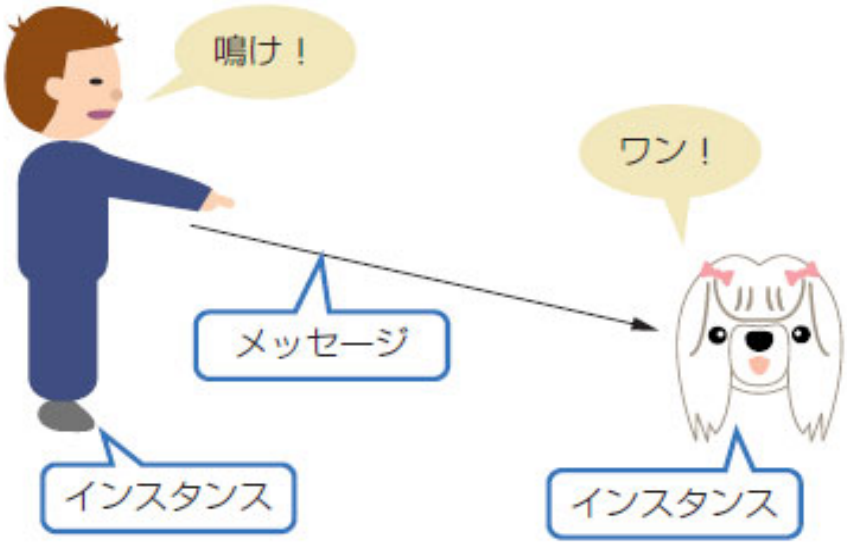
- Javaは、Object指向言語
  - Classや、Instanceが定義されている。。
    - クラスで概念的な設計を行う。
    - Instanceでクラスに実体を持たせる
  - その一方で、intやboolean, floatなどの「型」がある。
    - 従来の「言語」の特徴を踏襲
- Rubyには変数の「型」がない
  - 全て、Classとして定義されている。
  - 「型」の継承、定義の拡張が容易にできる。
  - その気になれば、自分で「変数型」が作れる。

Railsの強烈な「コンセプト」

- CoC （Convention over Configuration）
  - Javaをはじめとする多くの言語環境が、XMLを設定用に用いて、設定が違っているとうまく動作しなかったしている
  - 「規約」をきちんとして「設定」は不要をにしたい
    ルールに従わないと、全く動作させられない！
- DRY
  - Don't Repeat Yourself
  - 同じ記述をあちこちで繰り返すな、ということ

メッセージパッシング

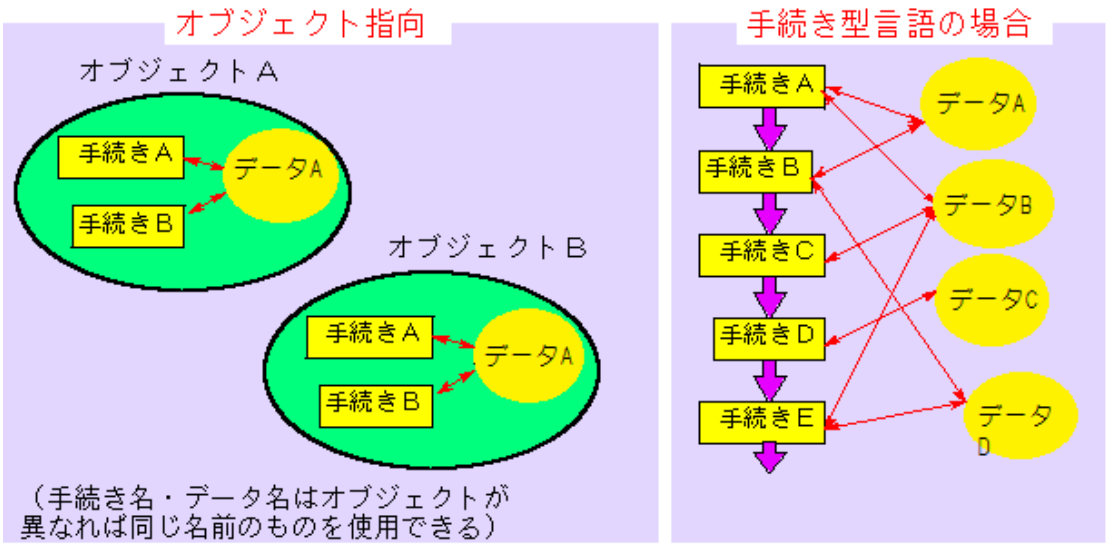
- オブジェクト指向では、インスタンス間のメッセージのやり取りで、プログラムが進行する。
  - 飼い主が飼い犬に対して“鳴け”とメッセージを送ると“ワン”と応える(「実行」)



カプセル化とメソッド

- 一つのオブジェクトを一つの塊（クラス）として
  カプセル化したものをオブジェクトと呼ぶ。

  オブジェクト = データ（プロパティ） + 手続き
- この手続き部分がメソッド



イベントドリヴン

- Event Driven (イベント駆動）
  - ここで、Eventというのは、「キーが押された」とか、「作業が終わった」とか、特定の「事象の発生」（何かが起きた）ということを表す。

「何かが起きたら」それに対応して「こうする」

- という形で、プログラムを書いていく。
  - このEventの例には、

    ユーザがボタンをクリックした、データを受信し
  - などがある。

Ruby文法の参照

Rubyの文法は、Railsなどの環境やコマンドと異なり、あまり変化していません。

一度覚えればそのまま使える「言語」です。

最新の仕様が反映されるので、WEBからの参照か、または、WEBページのcloneをローカルに作成するのが便利です。

文書の入手/Online Document

日本Rubyの会から

http://www.ruby-lang.org/ja/documentation/

ルビーアソシエーションのtutorialサイトから

http://www.ruby.or.jp/ja/tech/development/ruby/tutorial/

今日の演習

2通りのrubyのコマンド実行を試して下さい。

- irb（ interactive Ruby）
- rubyのファイルからの実行

irb

- ruby console Windowから立ち上げる

  irb

  - と入力する。

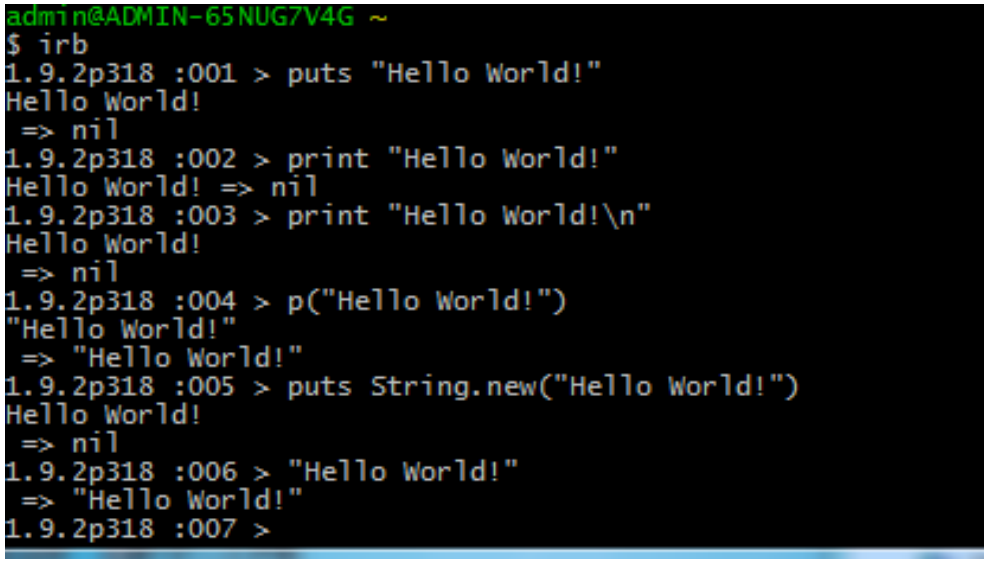
- 対話型なので、入力するたびに実行される。

演習1

- irbで以下のコマンドを入力する。
  - puts “Hello World!”
  - print “Hello World!”
  - print “Hello World!\n”
  - p(“Hello World!”)
  - puts String.new(“Hello World!”)
  - “Hello World!”

演習1の結果

- どれも、”Hello World!”の文字列を表示する。
- putsは改行が最後に入るが、printは改行しない。
- \n(￥n)を最後に加えるとprintでも改行する。
- String.new(“Hello World!”)では、Stringクラスの新しいインスタンスを生成する。
- p(オブジェクト) で、オブジェクトを表示。
- 最後のパターンが重要
  - 文字列を書くだけで表示
- 出力先はどちら？
  - コンソール？
  - プログラムで処理？



以下のコマンドを実行する。

- irb(main):001:0> require "active_support/all"
- => true
- irb(main):002:0> puts "ox".pluralize
- oxen
- => nil
- irb(main):003:0> puts "data".singularize
- datum
- => nil
- irb(main):004:0> puts "category".pluralize
- categories
- => nil
- irb(main):005:0>

演習2の結果について

- railsでは、規約で単数形、複数形の扱いが細かく決まっている。
- このため、ruby自身が単数形、複数形を英語のルールに従い、変換できる。
- このためのライブラリは、active_supportに入っている。
- 試しに、以下の単語で複数形を調べて下さい。

  foot, elf(妖精), life, child, mouse, thesis(卒業論文)

演習３

- 次のページのrubyプログラムを、授業支援システムからダウンロードして、「適当な」フォルダに置いて下さい。
  - （「適当な」は、自己管理でお願いします。）
  - ファイル名は、sample1.rb　です。
  - このファイルは、「改行」はWindows形式ですが文字コードはUTF-8です。Windows上では文字化けします。

sample1.rb

```
#! ruby -Ku
# -*- coding: utf-8 -*-

# Studentクラスを作る
class Student
    # Studentクラスのインスタンスを初期化する
    def initialize( name, age )
      @name = name
      @age = age
    end

    # name属性、age属性のアクセサ
    attr_accessor :name, :age

    # Studentクラスのインスタンスの文字列表現を返す
    def to_s
      "#@name, #@age"
    end
end

# Studentクラスのインスタンスを作成し、shinという名前をつける
shin = Student.new( '久保秋　真', 45 )

# ゲッターを使ってインスタンスの名前と年齢を表示する
puts "氏名：#{shin.name}, 年齢：#{shin.age}歳"

# セッターを使ってshinの名前と年齢を変更する
shin.name = 'singh, Tiger Jeet'
shin.age = 445          # 彼はなんと400と45歳!

# ゲッターを使ってshinの名前と年齢を表示する
puts shin.name
puts shin.age

puts "氏名：#{shin.name}, 年齢：#{shin.age}歳"
```

久保秋真著「作りながら学ぶRuby入門」, ソフトバンククリエイティブ, 2009,
ISBN-13: 978-4797352603, ¥2,625-

演習3の実行

- コンソール端末で、

  ruby ファイル名

- と入力して、実行して下さい。
- これの結果について解説します。

```
[root@cisnote rails3work]# ruby sample1.rb
氏名：久保秋　真，年齢：45歳
singh, Tiger Jeet
445
氏名：singh, Tiger Jeet，年齢：445歳
[root@cisnote rails3work]#
```

```
[root@cisnote rails3work]# ruby sample2.rb
true
Shinsaku Takasugi
みつかりません
[root@cisnote rails3work]#
```

rubyのコメント

- コメントの入れ方
  - #で始まる行
  - =begin行　から　=end行　の間
  - __END__　行の後　（アンダースコア_は二つ）

Magic comment

コーディングの文字コードを指定するため、
以下の先頭の2行を入れてあります。

```
#! ruby –Ku
# -*- coding: utf-8 -*-
```

1行目　：　uはutf-8, sならshift-JISで、1.8用

2行目　：　magic comment (1.9用）

http://www.kkaneko.com/rinkou/ruby/rubyintro.html
このページ内で、「スクリプトの文字コードの設定」で検索

変数

- rubyには「型」はなく、あらかじめ定義されたClassのインスタンスとして「変数」がある。
  - Ruby-lang.orgのサイトでも「変数」と言っていますので、「変数」で通します。

変数の命名規則

- ローカル変数　　name
- グローバル変数　　$name
- インスタンス変数　　@name
- クラス変数　　@@name
- 定数　　NAME

変数のスコープ

Ruby言語に限らず、すべてのプログラム言語で
変数のスコープを意識して使って下さい。

特定のクラスに所属するのか、特定のファイル
内でのみ参照されるのか、Globalなのか。

Globalスコープの場合、

スレッドローカルな変数か、
スレッドをまたいで値を保持するか・・・。

使い間違えると、悲惨な結果になります。

シンボル

: で始まる名前

- プログラム中のどこでも、同一のオブジェクトを名前で表せるようにしたもの。
- ハッシュのキー値などに使われる。

- ハッシュとは連想配列
  - 次回、掘り下げます。
- :シンボル.to_sで、シンボル名自体を表示できる。

sample2.rb

```
#! ruby -Ku
# -*- coding: utf-8 -*-

# 新しいハッシュを作る
friends = {
    :shin => "Shin Kuboaki",
    :shinichirou => "Shinichirou Ooba",
    :shingo => "Shingo Katori"
}

# ハッシュに要素を追加する
friends[ :shinsaku ] = "Shinsaku Takasugi"

# ハッシュの要素を検索する（見つかるはず）
puts friends.include?( :shinsaku )
puts friends[ :shinsaku ]
print :shinsaku.to_s, “: ”,friends[:shinsaku]

# 追加した要素を削除する
friends.delete( :shinsaku )

# ハッシュの要素を検索する（見つからないはず）
if friends.include?( :shinsaku ) then
    puts friends[ :shinsaku ]
else
    puts "みつかりません"
end
```

久保秋真著「作りながら学ぶRuby入門」, ソフトバンククリエイティブ, 2009,
ISBN-13: 978-4797352603, ¥2,625-
一部変更

Hashクラスのメソッド

クラスにどんなメソッドがあるか？

http://docs.ruby-lang.org/ja/2.3.0/class/Hash.html

Hash, Arrayなど基本的なクラス（主だったもの）だけを調べてみる。

Integer, Bignum, Fixnum, Float, String, File

など

今日の演習課題（レポート課題2）

- Sample1.rbと、sample2.rbの両方をそれぞれダウンロードして、走らせて下さい。
  - （文字コードや、改行コードには注意し、必要なら修正して下さい。）
- Hashを使うか、Classを使うか、いずれかの方法で、食堂のメニュー
  - curryは、300円です。
  - misoNoodleは、380円です。
- などと、3品目（以上）を表示させるプログラムを書いて下さい。
- 期限は、１週間後です。

演習課題の報告

1. 課題の内容:「Rubyのプログラム」を明記。
2. 「どんな表示を出すつもりでプログラムしたか。」
3. 「ソースコード」(コードの簡単な説明)
4. 課題ファイルの「実行画面:コンソール」のスクリーンショット

以上を報告して下さい。

Hashを用いるかClassを用いるか、どちらかの方法で解いた場合、報告が適切なら、レポートの評価はB評価とします。

両方のソースコードを試した場合でA評価、さらに、実際にプログラムをしてみて考察があった場合、S評価とすることがあります。

今日の授業を休んだ人は・・・

- Object指向言語とはどんな言語か、レポートとしてまとめてください。この部分の報告がなかったなら、「出席」扱いには切り換えません。

- 授業課題が行われていた場合には、授業課題は通常通り評価します。
- 提出用のレポートは、欠席課題と通常課題を一つにまとめても構いません。(absenceは書かなくても、出席に切り替えます。)