1 RDoc による自動ドキュメント生成

1.1 RDoc とは?

Ruby Documentation System (RDoc) とは, Ruby で書かれたソースコードからドキュメントを自動生成する, Ruby 本体に付属する標準ライブラリの 1 つです.

RDoc は Ruby ソースコードを解析し, クラス, モジュール, メソッドの定義を抜き出し, include や require に関して解釈します. そしてこれらの内容とその直前に書かれたコメントを併合し, HTML ドキュメントを出力します. 詳しくは 1.8 1,2 を参照ください.

1.2 概要

以下では, まず RDoc のインストールを行います. そして Ruby で簡単なクラスライブラリを作成し, RDoc を用いてそのプログラムからドキュメントを生成してみます.

前提として, Ruby 本体のインストールは行っておいてください.

1.3 インストール

Ruby をソースコードからインストールした場合

RDoc は, Ruby のバージョン 1.8.4 では既に標準ライブラリとして Ruby 本体に同梱されているはずです. Ruby のホームページ (1.8 3) からソースコードをダウンロードしてコンパイルした場合には RDoc も既にインストールされています.

バイナリパッケージを利用する場合

例えば Fedora, Vine, Debian などでは, ruby というパッケージとは別に rdoc というパッケージが用意されているので別途インストールを行う必要があります. (その他の場合は未調査です. すいません).

- Fedora Core の場合

```
# yum install rdoc
```

- Vine, Debian の場合

```
# apt-get install rdoc
```

1.3.1 インストールの確認

空のディレクトリを作成し, そのディレクトリ内に移動してください. そのディレクトリ内で, *rdoc* というコマンドが実行できることを確認してください.

```
$ mkdir rdoc_test
$ cd rdoc_test
$ rdoc
```

以下のようなメッセージが表示され, doc というディレクトリが作成されていれば OK です.

```
Generating HTML...

Files:   0
Classes: 0
Modules: 0
Methods: 0
Elapsed: 0.262s
```

1.4 サンプル Ruby スクリプト作成

まず, GPhys を用いて簡単なクラスライブラリを作成しましょう. (これよりも前のチュートリアルで既に GPhys は利用可能な状態だと仮定しています). 以下のプログラムを作成してください. <URL:sample/gphys_cont1.rb>

```
class GPhys_cont
  require "numru/ggraph" ; include NumRu

  FILENAME = 'T.jan.nc'

  attr_reader   :var

  def initialize(var='T')
    @var  = var
  end
  def cont
    gphys = GPhys::IO.open(FILENAME, @var)
    DCL.gropn(1) ; DCL.sgpset('lcntl', false) ; DCL.uzfact(0.7)
    GGraph.contour( gphys)
    DCL.grcls
```

```
  end
end

if __FILE__ == $0
  gphys = GPhys_cont.new
  gphys.cont
end
```

if __FILE__ == $0 以降の部分は, このプログラムを実行した際のメイン文に当たります. このプログラムは実際には GPhys チュートリアル – 6. とりあえず可視化のように動作します. 以下のデータファイルをダウンロードした後, 上記の Ruby スクリプトを ruby で実行してみてください.

- ダウンロード: NetCDF ファイル T.jan.nc

```
$ ruby gphys_cont1.rb
```

もう 1 つ, このクラスを継承したクラスライブラリを作成してみましょう. 以下のプログラムを作成してください. <URL:sample/gphys_tone1.rb>

```
require "gphys_cont1"
class GPhys_tone < GPhys_cont

  attr_accessor :draw_tone

  def initialize
    super
    @draw_tone = true
  end
  def tone(itr=1)
    gphys = GPhys::IO.open(FILENAME, @var)
    DCL.gropn(1) ; DCL.sgpset('lcntl', false) ; DCL.uzfact(0.7)
    GGraph.set_fig( 'itr'=>(itr == nil) ? 1 : itr.to_i)
    GGraph.tone( gphys ) if @draw_tone
    GGraph.contour( gphys, !@draw_tone )
    DCL.grcls
    return true
  end
end

if __FILE__ == $0
```

```
  gphys = GPhys_tone.new
  gphys.tone
end
```

このプログラムは, gphys_cont1.rb の色塗り版です.

```
$ ruby gphys_tone1.rb
```

1.5 RDoc によるドキュメント生成

1.5.1 まず rdoc を使ってみる

では次に, RDoc を用いてこのクラスライブラリのリファレンスマニュアルを自動生成してみましょう. gphys_cont1.rb, gphys_tone1.rb が置いてあるディレクトリで以下のコマンドを実行してください.

```
$ rdoc gphys_cont1.rb gphys_tone1.rb --main GPhys_cont
```

引数 "gphys_cont1.rb gphys_tone1.rb"
: 対象となるファイルです. ここでは明記していますが, ファイルを明記しない場合はカレントディレクトリ以下に存在する全ての "rb" という拡張子を持つファイルを検索します.

引数 "–main GPhys_cont"
: メインページを指定します. ここではクラス GPhys_cont をメインページに指定します.

このコマンドにより, doc というディレクトリが作成され, その中に RDoc によって作成されたドキュメントが出力されたはずです. ブラウザで doc/index.html を見てみましょう. 以下のようなページが表示されるはずです.

- GPhys_cont (コメント無し)

RDoc により作成されたドキュメント (コメント無し)

上段の 3 分割されたフレームにファイル, クラスおよびモジュール, メソッドのリストが表示されています. 下の部分には GPhys_cont クラスの内容が表示されています. 下のフレームのメソッド名の部分をクリックすると, ソースコードが表示されます.

1.5.2 コメントを書き込んでみる

ソースコードにコメントを埋め込むことで, ドキュメントにより多くの情報を付加してみましょう. gphys_cont1.rb にコメントを追加した以下のファイルを作成しましょう.

<URL:sample/gphys_cont2.rb>

```
#
# GPhys を利用して等値線図を描画するクラスライブラリ
#
class GPhys_cont
  require "numru/ggraph" ; include NumRu

  # ファイル名（固定）
  FILENAME = 'T.jan.nc'

  # 描画する変数
  attr_reader   :var

  #
  # 初期化処理用のメソッド．引数 _var_ には描画する変数を
  # 与えます．
  #
  def initialize(var='T')
    @var  = var
  end

  #
  # ※ 空行を入れると，それより上の部分は無視されます．
  #

  #
  # 等値線図の描画を実行します．
  #
  def cont
    gphys = GPhys::IO.open(FILENAME, @var)
    DCL.gropn(1) ; DCL.sgpset('lcntl', false) ; DCL.uzfact(0.7)
    GGraph.contour( gphys)
    DCL.grcls
  end
end

if __FILE__ == $0
  gphys = GPhys_cont.new
  gphys.cont
end
```

クラスやメソッド定義の直前に書かれているコメントが各々のドキュメントとして解釈されます. なお, 空行をいれた段階でそれよりも上部の部分 (上のソースコードで言うと「※ 空行を入れると...」の部分) はドキュメントとして解釈されません.

では, 再度 rdoc コマンドを実行してみましょう.

```
$ rdoc gphys_cont2.rb gphys_tone1.rb --main GPhys_cont --charset euc-jp
```

引数 "–charset euc-jp"

ソースコード内にマルチバイトの文字 (日本語など) が含まれる場合は必ずこのオプションを指定してください. ソースコード内のマルチバイト文字の文字コードに合わせ, "euc-jp", "shift_jis", "iso-2022-jp" のいづれかを指定します.

今度は, 以下のようなページが生成されます.

- GPhys_cont (コメントあり)

RDoc により作成されたドキュメント (コメントあり)

ソースコード内のクラスやメソッドの上部に書かれたコメントがドキュメントに反映されているのが分かります.

1.6 RDoc の便利な機能を使ってみる

1.6.1 コメント部の修飾

RDoc のコメント部はかなり自然に書くことができますが, いろいろな修飾も可能になっています.

gphys_tone1.rb にコメントを追加した以下のファイルを作成しましょう.

<URL:sample/gphys_tone2.rb>

```
require "gphys_cont2"
#
#= GPhys を利用して色塗り図を描画するクラスライブラリ
#
#Authors::   森川 靖大
#Version::   1.2 2006-03-08 morikawa
#Copyright:: Copyright (C) GFD Dennou Club, 2006. All rights reserved.
#License::   Ruby ライセンスに準拠
#
#-- (#-- から #++ までの部分を RDoc は解釈しません.)
#"=", "==", ""===" は見出しを表します.
```

```
#
#= 見出しレベル 1
#== 見出しレベル 2
#=== 見出しレベル 3
#++
#
#このクラスのスーパークラスは GPhys_cont です.
#new メソッドで初期化を行い，tone メソッドで描画を行います.
#
#--
# モジュール名やメソッド名はそのままモジュールやメソッドへのリンクに
# 変換されます.
#++
#
#=== 参考資料
#
#* http://ruby.gfd-dennou.org
#  1. GPhys[http://www.gfd-dennou.org/library/ruby/products/gphys/]
#  2. {2006 年 電脳ruby セミナー・電脳davis/ruby ワークショップ}[http://www.gfd-dennou.org/libr
#
#--
#==リストの表示に関して
#
#リストは以下のような記号が付いたパラグラフです.
#
# - '*' もしくは '-' で普通のリスト
# - 数字+ピリオドで番号付きリスト
# - アルファベット+ピリオドでアルファベットリスト
#
#
#== リンクに関して
#
# http:, mailto:, ftp:, www. で始まるテキストはウェブへのリンクだ
と
# 判別されます.
#
# label[url] の形式でもハイパーリンクが張れます. この場合は lavel が
表
# 示され，url がリンク先となります. label が複数の単語を含んでいる
```

```
場合
# (日本語の場合はこっちを使ってください), 中括弧を使い, <em>{multi word
# label}[</em>url<em>]</em>としてください.
#++
#
#=== 開発履歴
#
#* 1.2 2006-03-08
#  * 堀之内さんのコメントを下に, 作者やライセンス, 開発履歴の
#    欄を足してみる.
#
#* 1.1 2006-03-07
#  * とりあえず作成してみる.
#
class GPhys_tone < GPhys_cont

  # 図に色塗りを行うかどうかのフラグ.
  # このフラグを false や nil にした場合, GPhys_cont#cont と
  # 同様に動作します.
  #
  attr_accessor :draw_tone

  #
  #=== 初期化処理用メソッド
  #
  #GPhys_cont#new を参照してください.
  #
  #--
  # 別のモジュール内のメソッドへリンクする場合は
  # "<i>モジュール名</i>#<i>メソッド名</i>" と指定します
  #++
  #
  def initialize
    super
    @draw_tone = true
  end

  #=== 描画メソッド
  #
```

```
  #色塗り図を描画するメソッド．等値線図のみを描画したい場合は
  #GPhys_cont#cont を利用してください．
  #
  #_itr_ :: 描画する際の地図投影法を指定します．数値を与えてくださ
い．
  #          デフォルトは 1 になっています．番号と投影法の関係に関し
ては
  #          http://www.gfd-dennou.org/library/dcl/dcl-f90/doc/term/2d.htm
  #          を参照ください
  #
  #返り値:: 常に true が返ります．
  #
  def tone(itr=1)
    gphys = GPhys::IO.open(FILENAME, @var)
    DCL.gropn(1) ; DCL.sgpset('lcntl', false) ; DCL.uzfact(0.7)
    GGraph.set_fig( 'itr'=>(itr == nil) ? 1 : itr.to_i)
    GGraph.tone( gphys ) if @draw_tone
    GGraph.contour( gphys, !@draw_tone )
    DCL.grcls
    return true
  end
end

if __FILE__ == $0
  gphys = GPhys_tone.new
  gphys.tone
end
```

では, 再度 rdoc コマンドを実行してみましょう.

```
$ rdoc gphys_cont2.rb gphys_tone2.rb --main GPhys_tone --charset euc-jp
```

今度は, 以下のようなページが生成されます.

- GPhys_tone

RDoc により作成されたドキュメント (いろいろ修飾)

モジュールやメソッドなどに自動的にリンクがはられ, 見出し, リスト表示が行われていることがわかります.

修飾のための書式に関して, ソースコードに解説が付記してあるので参照してください. より詳しい情報は, 1.8 1, 2 の "MarkUp" の部分を参照してください. ソースコード内で解説が書き込んである '#--' 〜'#++' の部分に関しては RDoc は無視するため, ドキュメントに反映されません.

1.6.2 便利なオプション

rdoc コマンドのオプションのうち, 上記で説明しなかった便利なものをいくつか紹介します. より詳しい情報は, 1.8 1, 2 の ”Usage” または ”使い方” の部分を参照してください.

–all, -a
: private 属性のメソッドもドキュメントに表示します. 開発者向けのドキュメントとして便利かもしれません.

–diagram, -d
: クラスの継承関係などを画像化して表示します. Dot が必要になります. (Fedora, Vine, Debian ならば graphviz パッケージのインストールで利用可能になります).

–inline-source, -S
: ソースコードの表示をポップアップではなく, ページ内で表示するようにします.

–op, -o dir
: dir ディレクトリにドキュメントを出力します.

–title, -t text
: text を HTML のタイトルに設定します.

以下のコマンドで作成したドキュメントを載せておきます. (コマンドプロンプトや DOS 窓を利用している方がコピペしやすいように, 改行しないものも載せておきます)

```
$ rdoc gphys_cont2.rb gphys_tone2.rb --main GPhys_tone \
      --charset euc-jp --inline-source --diagram \
      --title "GPhys_tone and GPhys_cont Documentation"
```

```
$ rdoc gphys_cont2.rb gphys_tone2.rb --main GPhys_tone --charset euc-jp --inline-source
```

- GPhys_tone and GPhys_cont Documentation

1.7 RDoc ドキュメントのサンプル

RDoc を用いて作成されたドキュメントをいくつか紹介します.

- Ruby Standard Library Documentation
- Ruby on Rails

1.8 参考資料

1. rdoc: Ruby Standard Library Documentation
2. 大林一平さんによる上記ページの日本語訳
3. オブジェクト指向スクリプト言語 Ruby の本家サイト