

<http://jp.rubyist.net/RubyKaigi2008/?SubSession> より抜粋

Ruby構文による構造化データ記述 (前田和昭)

Rubyを構造化データを記述するためのDSL(Domain Specific Language)として使い、JRubyを紹介して、Javaから利用可能にする試みを進めている。研究用プロトタイプではなく、実務アプリケーションでの本格利用が目標である。本発表では、DSLとしてRubyをどのように使い、Javaプログラムとどのように連携させるかについて説明する。

仕事仲間にRubyに詳しい人がいなくて、厳しいコメントをくれる人がいません。叱咤激励・厳しいコメントなど何でも感謝します。よろしくお願いします。

前田和昭 - 日本Rubyの会

Rubyの入門書を読み始めてから。まだ数ヶ月しかたっていない初心者です。ソースコードの解析を中心に仕事をやっています。最近、XMLを使ったデータ記述に飽きてしまい、XMLを見るのがイヤになってきたので。Rubyでデータを記述することを実験中です。年齢が40歳になっても50歳になっても、プログラマとして仕事したいですね。

Structured Data Representation Using Ruby Syntax

Kazuaki Maeda

kmaeda@gmail.com

Ruby構文による構造化データ記述

Structured Data Representation Using Ruby Syntax

前田和昭()
Kazuaki Maeda
日本Rubyの会
kmaeda@gmail.com

日本Ruby会議 2008

2008年6月22日

3

はじめに

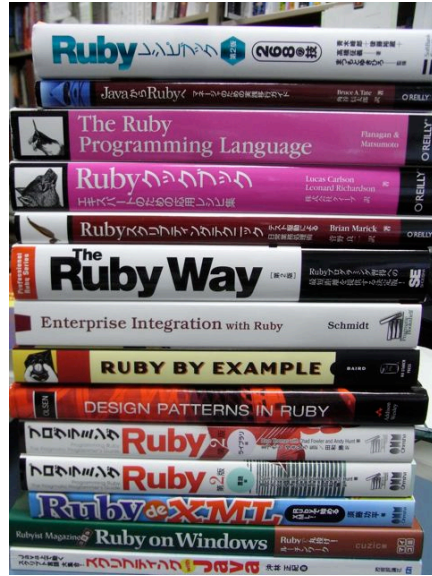
1. Big Nameではありません
 いろんな「前田さん」がいますが、私は無名です
2. 最近「日本Rubyの会」に入会しました
 私のビジネスとは分離した話です
 kaz@acm.orgで登録して困っています
3. Railsを知りません
 Webプログラミングしません
4. 数ヶ月前から学び始めた初級レベルです
5. 周りにRuby知っている人が全くいません
 -> いろいろ教えてください！

日本Ruby会議 2008

2008年6月22日

4

定番?の積読



日本Ruby会議 2008

2008年6月22日

5

Java本

小森裕介さん著

なぜあなたはJavaでオブジェクト指向開発ができないのか, 技術評論社(のWebページから抜粋)

ダウンロード

本書に掲載したサンプルプログラムをダウンロードできます。プログラミングの学習に、どうぞご利用ください。

◦ JavaSeminar.zip

● Java以外の言語で作成されたサンプルファイルのダウンロード

以下のWebサイトにおいて、本書サンプルプログラム (JavaSeminar.zipと同等) の各言語移植版が公開されています。各言語におけるオブジェクト指向の習得はもちろん、各ソース間の比較学習/研究等、幅広くご利用ください。

- C#版 (小野修司様による移植)
- C++版 (田中秀生様による移植)
- Ruby版 (吉田裕美様による移植)

※なお、ここで配布するプログラムの使用により生じたいかなる損害に対しても、技術評論社および著者はいっさいの責任および損害に対する責務を負いません。あくまでも自己責任のもとでのご使用をお願いいたします。あらかじめご承知おきください。

日本Ruby会議 2008

2008年6月22日

6

新しいもの好きです

新しいものに寄生すること多いです

前田Kazuaki

-> 前田K

前田K
まいたけ？

<Wikipediaによれば>

マイタケは世界中の暖温帯から温帯北部にかけて分布し、ナラ類、カシ類、シイ類といったブナ科樹木の大木の根株で心材に寄生して白色腐朽を引き起こす木材腐朽菌である。

Ruby構文による構造化データ記述

Structured Data Representation Using Ruby Syntax

前田和昭(前田K)
Kazuaki Maeda
日本Rubyの会
kmaeda@gmail.com

日本Ruby会議 2008

2008年6月22日

9

前田Kのいろいろ

1. いまだにプログラム作成(20K~50K LOC/年)
 - + コンパイラのフロントエンド
VHDL, Java, SQL, C#, Visual Basic, etc.
 - + ソフトウェア開発ツール, リバースツール
2. プログラミング言語歴
 - + C, (Smalltalk), C++, (Java), C#, そして, Java, Ruby
 - + Smalltalkを勉強したが挫折
 - + Javaを勉強したが挫折, 現在再挑戦中
 - + ただいまRubyを勉強中! (=Perl + Smalltalk?)

日本Ruby会議 2008

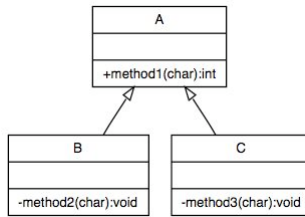
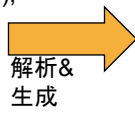
2008年6月22日

10

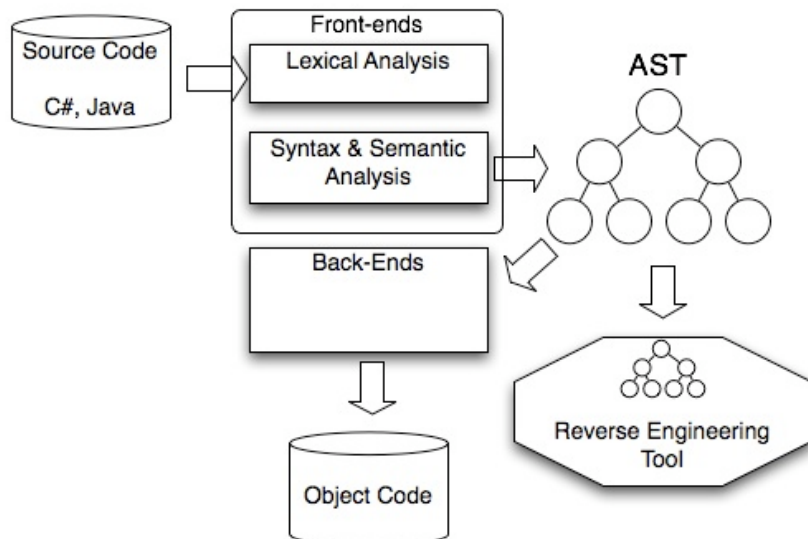
リバースツール

- + C#ソースコードからダイアグラム生成
- + Mac OS Xで動作中
 - × Mono C#, C#コンパイラ (& Jay) 改造 -> Monoに寄生

```
namespace N{  
  public class A {  
    public int method1(char a){  
      Console.WriteLine("hello world");  
    }  
  }  
  public class B: A {  
    private void method2(char b){ }  
  }  
  public class C: A{  
    private void method3(char c){ }  
  }  
}
```



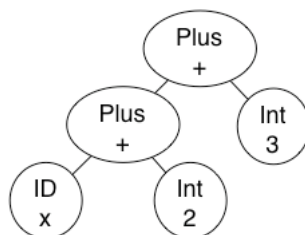
リバースツール



構造化データ

コンパイラフロントエンドはASTを作成
(Abstract Syntax Tree, 抽象構文木)

$x + 2 + 3$



構造化データ

Ruby構文による構造化データ記述

Ribbon

- + Ruby構文による新しいデータ記述
- + Ruby Instructions Becoming Basic Object Notation

特徴

- + とりあえず, Javaを対象
- + データをファイルに読み書きするときの記述
 - × Serialization/Deserialization, Marshalling/Unmarshalling

XML & Ribbon

```
<config name="my behavior">  
  <before action="programming"/>  
  <buy object="coffee">  
    <at store="starbucks"/>  
</config>
```

```
config "my behavior" do  
  before "programming"  
  buy "coffee"  
  at "starbucks"
```

end

日本Ruby会議 2008

2008年6月22日

15

XML

日本Ruby会議 2008 2008年6月22日

16

XMLの例

```
<config name="my behavior">
  <before action="programming"/>
  <buy object="coffee">
    <at store="starbucks"/>
  </buy>
</config>
```

属性中心

```
<config name="my behavior">
  <before>programming</before>
  <buy>coffee</buy>
  <at>starbucks</at>
</config>
```

要素中心

日本Ruby会議 2008

2008年6月22日

17

struts-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<form-beans>
  <form-bean name="CoffeeForm" type="com.starbucks.struts.action.CoffeeForm" >
    <form-property name="number" type="java.lang.Integer" />
  </form-bean>
</form-beans>
<action-mappings>
  <action name="CoffeeForm" path="/coffee"
    type="action.catalog.CoffeeShowAction">
    <forward name="show" path="catalog" />
  </action>
</action-mappings>
```

日本Ruby会議 2008

2008年6月22日

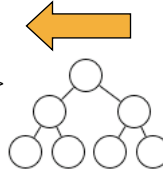
18

JavaML (WWW Conf., 2000, Amsterdam)

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE java-source-program SYSTEM "java-ml.dtd">
3
4 <java-source-program name="FirstApplet.java">
5 <import module="java.applet.*"/>
6 <import module="java.awt.*"/>
7 <class name="FirstApplet" visibility="public">
8 <superclass class="Applet"/>
9 <method name="paint" visibility="public" id="meth-15">
10 <type name="void" primitive="true"/>
11 <formal-arguments>
12 <formal-argument name="g" id="frmarg-13">
13 <type name="Graphics"/></formal-argument>
14 </formal-arguments>
15 <block>
16 <send message="drawString">
17 <target><var-ref name="g" idref="frmarg-13"/></target>
18 <arguments>
19 <literal-string value="FirstApplet"/>
20 <literal-number kind="integer" value="25"/>
21 <literal-number kind="integer" value="50"/>
22 </arguments>
23 </send>
24 </block>
25 </method>
26 </class>
27 </java-source-program>

```



```

import java.applet.*;
import java.awt.*;

public class FirstApplet
    extends Applet {
    public void paint(Graphics g) {
        g.drawString("FirstApplet", 25, 50);
    }
}

```

XML

XMLは、幅広く使われるようになった

でも、

嬉しいこと、嫌なことがある

XMLの嬉しいこと

プラットフォーム独立

- + プログラミング言語
- + オペレーティングシステム
- + コンピュータ(ベンダー)

プレーンテキストなので可読

標準であり, 仕様が豊富

- + DOM, SAX, XPath, XSLT, SOAP など

ライブラリが豊富

日本Ruby会議 2008

2008年6月22日

21

XMLの嫌なこと

Seven problems of highly ineffective XML

1. タグが見つらい Many ugly tags
2. 人間が読めるフォーマット? でも読みたくない
3. 属性中心, 要素中心, どちらが良いのか?
4. 仕様が多すぎて, どう「しよう」もない
XML, XML Schema, XSL, XSL-T, XML Query, WS-*
5. そんなに多くの仕方を正しく実装できるのか?
XML Schema, XML Query, WS-*
6. Name spaceは便利だけど, 複雑になる
7. DOM APIがあっても, 結局自分でAPIをかぶせる
DOM APIとアプリケーションAPIのミスマッチ

日本Ruby会議 2008

2008年6月22日

The Seven Habits of Highly Effective People

Ribbonの簡単な紹介

Ribbon =
Ruby Instructions Becoming
Basic Object Notation

日本Ruby会議 2008 2008年6月22日

23

Ribbonを考えたきっかけ

1. JRuby 1.0のリリース
2. Software Engineering Radio (www.se-radio.net)
Episode 52: DSL Development in Ruby
3. Dynamically Typed Languagesの特集号
Building Domain-Specific Languages for Model-Driven
Development
IEEE Software, pp.48-55, Sep/Oct, 2007
Abstractによれば,
 - × DSLをDynamic Languageに組み込めば, 読みやすく
て, DSL開発時間を短くできる
 - × 処理系を書かなくてよい

内部DSLで, Rubyをホスト言語とする

日本Ruby会議 2008

2008年6月22日

24

その記事から抜粋

```
class Relational < MetamodelKeyword
  metaclass 'Table' do
    attriute 'name', String
    .....
  end
  metaclass 'Column' do
    attribute 'name', String
    attribute 'type', String
    attribute 'primary', Boolean
  end
end
```

```
class MetamodelKeywords
  def self.metaclass(name)
    @classes << Ecore::Eclass.new(name)
    yield
  end
  def self.attribute(name,type)
    att = Ecore::EAttribute.ew
    att.name = name
    @classes.last.attributes << att
  end
end
```

日本Ruby会議 2008 2008年6月22日

25

Ribbon

構造化データの記述法

設計指針

- + より簡潔に, より読みやすく
- + Ruby構文を採用
- + 性能は, とりあえず後で検討

一行に, 一つの「要素」

各要素は, 一つの名前と一つの値を持つ

例:

```
config "my behavior"
```

名前->config 値->"my behavior"

日本Ruby会議 2008

2008年6月22日

26

Ribbon

4つの特徴

1. 子要素
2. 基本データ型
3. リスト
4. 構造をまたいだリンク

1: 子要素

- Rubyのブロックで子要素を表現

```
config "my behavior" do
  before "programming"
  buy "coffee"
  at "starbucks"
end
```

2: 基本データ型

- 4つの基本データ型

```

config "my behavior" do
  pay "today" do
    price 340 ← int
    tax 0.05 ← real
    togo true ← bool
    date "April 1, 2008" ← string
  end
  .....
end

```

3: リスト

- 同じ名前の要素で表現

```

config "my behavior" do
  .....
  buy "coffee" do
    order "grande" ← order 要素1
    order "hot" ← order 要素2
    order "double-shot" ← order 要素3
  end
  .....
end


```

4: 構造をまたいだリンク

ユニークな識別子でリンクを表現

```
location "starbucks" do
  at "Seattle, Washington" do
    uuid :dbe0c2e6_5a63_4159_a76f_6e44
  end
end

config "my behavior" do
  before "programming"
  buy "coffee"
  at :dbe0c2e6_5a63_4159_a76f_6e44
end
```



Ribbon

4つの特徴

1. 子要素
2. 基本データ型
3. リスト
4. 構造をまたいだリンク

構造化データの定義

```
config "my behavior" do
  pay "today" do
    price 340
    tax 0.05
    togo true
  end
  at "starbucks"
end
```

定義もRuby構文を使用

```
:config.has :pay, :at
:pay.has :price, :tax, :togo
:price.is_type :int
:tax.is_type :real
:togo.is_type :bool
:at.is_type :string
```

他には,

- is_seq_of でリストの定義
- is_a で継承が利用可能

日本Ruby会議 2008

2008年6月22日

33

ツール側：定義からの生成

生成ツール ribgen (Rubyで作成)

コマンドラインから実行して,

- + 構造化データの定義
- + 言語に依存した定義

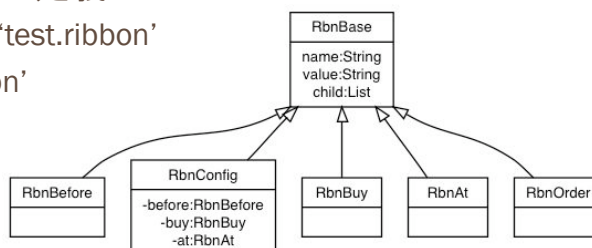
```
java_package 'test.ribbon'
java_prefix 'Rbn'
generate_java
```

を読んで,

Javaクラス

Rubyによる設定ファイル

を生成



日本Ruby会議 2008

2008年6月22日

34

Mono C#のソースコードから (1)

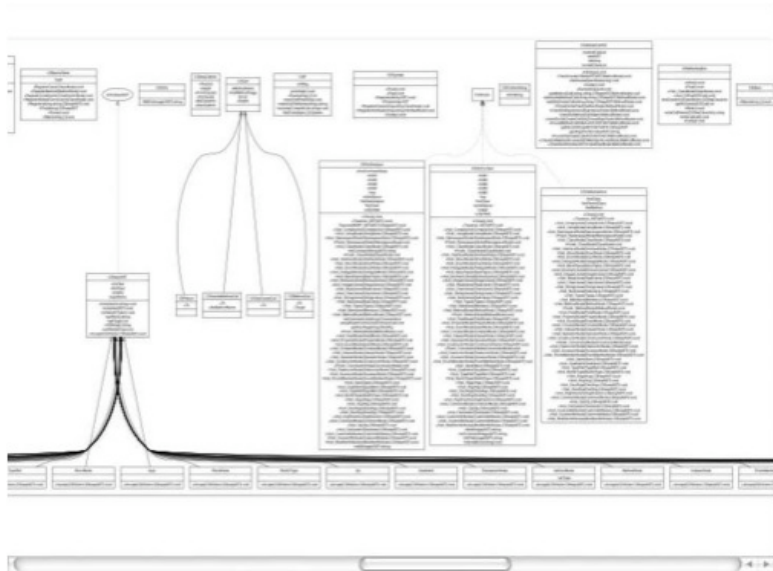


日本Ruby会議 2008

2008年6月22日

35

Mono C#のソースコードから (2)



日本Ruby会議 2008

2008年6月22日

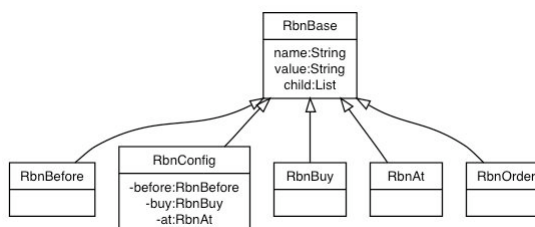
36

アプリケーション側：Reader-Writerの例

```

RbnBase obj;
RbnConfig c;
RbnBefore b;
RbnAt a;
RbnAPI rbnapi = new RbnApi();
obj = rbnapi.readFile("test-in.rb");
c = (RbnConfig)obj;
b = c.getBefore();
b.setValue("朝食");
rbnapi.writeFile(c,"test-out.rb");

```



<- Javaです

アプリケーション側：Reader-Writerの例

- test-in.rb


```

config "my behavior" do
  before "programming"
  buy "coffee"
  at "starbucks"
end

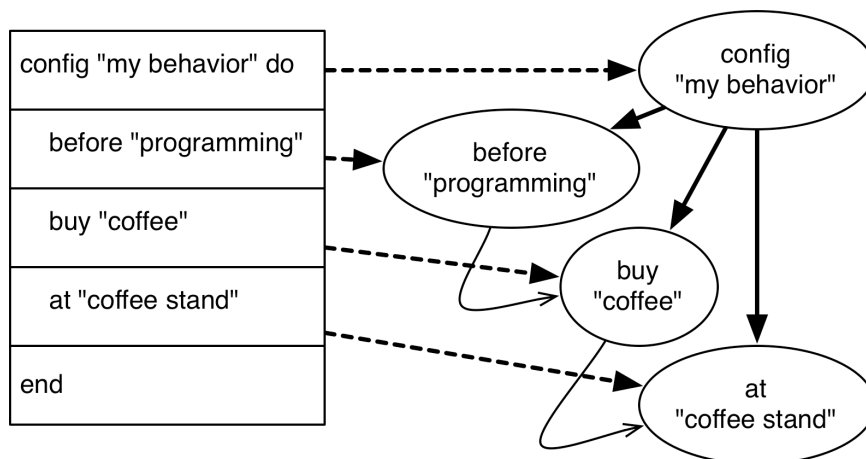
```
- test-out.rb


```

config "my behavior" do
  before "朝食"
  buy "coffee"
  at "starbucks"
end

```

Ribbonデータを読むときは...



日本Ruby会議 2008

2008年6月22日

39

実装

開発環境

- + Mac OS X 10.5.3
- + JRuby 1.1.2 (1.0.xだと日本語が通らない?)
- + Java 1.6.0_05 (以前は, Java 1.5.0_13)
- + NetBeans 6.1

Ribbonの現ターゲット

- + Java5 パーザ
- + エディタ

日本Ruby会議 2008

2008年6月22日

40

JRubyを使ってみて...経験

RubyとJavaを混在させる

- + NetBeans 6.1を使っていると,
 - × Rubyの編集, 実行, デバッグが便利
 - × Javaの編集, 実行, デバッグが便利
 - × Ruby+Javaでどうやってデバッグするのか不明

- + Java<->Rubyを実行させると,
 - × 例外が起きると, さっぱり分からなくなる

- + RubyからJavaオブジェクトを一杯newすると,
 - なんだか遅い -> そのうち実験予定

日本Ruby会議 2008

2008年6月22日

41

NetBeans 6.1で, Ribbonを使う

```

1 | #
2 | # To change this template
3 | # and open the template i
4 |
5 |
6 | config "my behavior" do
7 |   before "programming"
8 |   buy "coffee" do
9 |     order "grande"
10 |    order "hot"
11 |    order "double-shot"
12 |   end
13 |   at "starbucks"
14 |   pay "today" do
15 |     price 340
16 |     tax 0.05
17 |     goto true
18 |   end
19 | end

```

```

1 | #
2 | # To change this template, choose
3 | # and open the template in the ed
4 |
5 | require 'ribanally.rb'
6 | require 'ribsynth.rb'
7 |
8 | class MyConfig < MetaRibbon
9 |
10 |   :config.has :pay, :at
11 |   :pay.has :price, :tax, :togo
12 |   :price.is_type :int
13 |   :tax.is_type :real
14 |   :togo.is_type :bool
15 |   :at.is_type :string
16 |
17 |
18 |   java_package 'aaa.bbb'
19 |   java_prefix 'Rbn'
20 |   generate_java
21 |   generate_ribsetup 'rbn-init.rb'
22 | end
23 |

```

日本Ruby会議 2008 2008年6月22日

42

まとめ

Ribbon

- + Ruby構文を使ったデータ記法

プログラム生成ツール: ribgen

- + 入力: 構造化データの定義
- + 出力: Javaプログラム, Rubyプログラム(初期設定)

面白いこと

- + Ruby構文を使うので, IDEをそのまま利用可能
- + 実行するので, Iterator & Visitor で便利かも?

Thank you !

Any questions or comments ?

前田K

Kazuaki Maeda

kmaeda@gmail.com
