http://www.gfd-dennou.org

RDoc による 数値モデルドキュメント作成 (解説編)

北海道大学/神戸大学
森川 靖大

第7回 地球流体電脳 davis/rubyワークショップ 2008 年 3 月 10 日 京大生存研遠隔講義室 HS109

はじめに

- ドキュメントの重要性
  - 第三者への提供 (プログラムの利用)
  - 開発や保守の効率化 (プログラムの改変)

- Fortran による数値モデルのドキュメント
  - 数理、離散化ドキュメント :: TeX
    - 数式の記述に最適
  - リファレンスマニュアル :: HTML
    - Web からの参照に最適、ハイパーリンクが便利

リファレンスマニュアルの作成

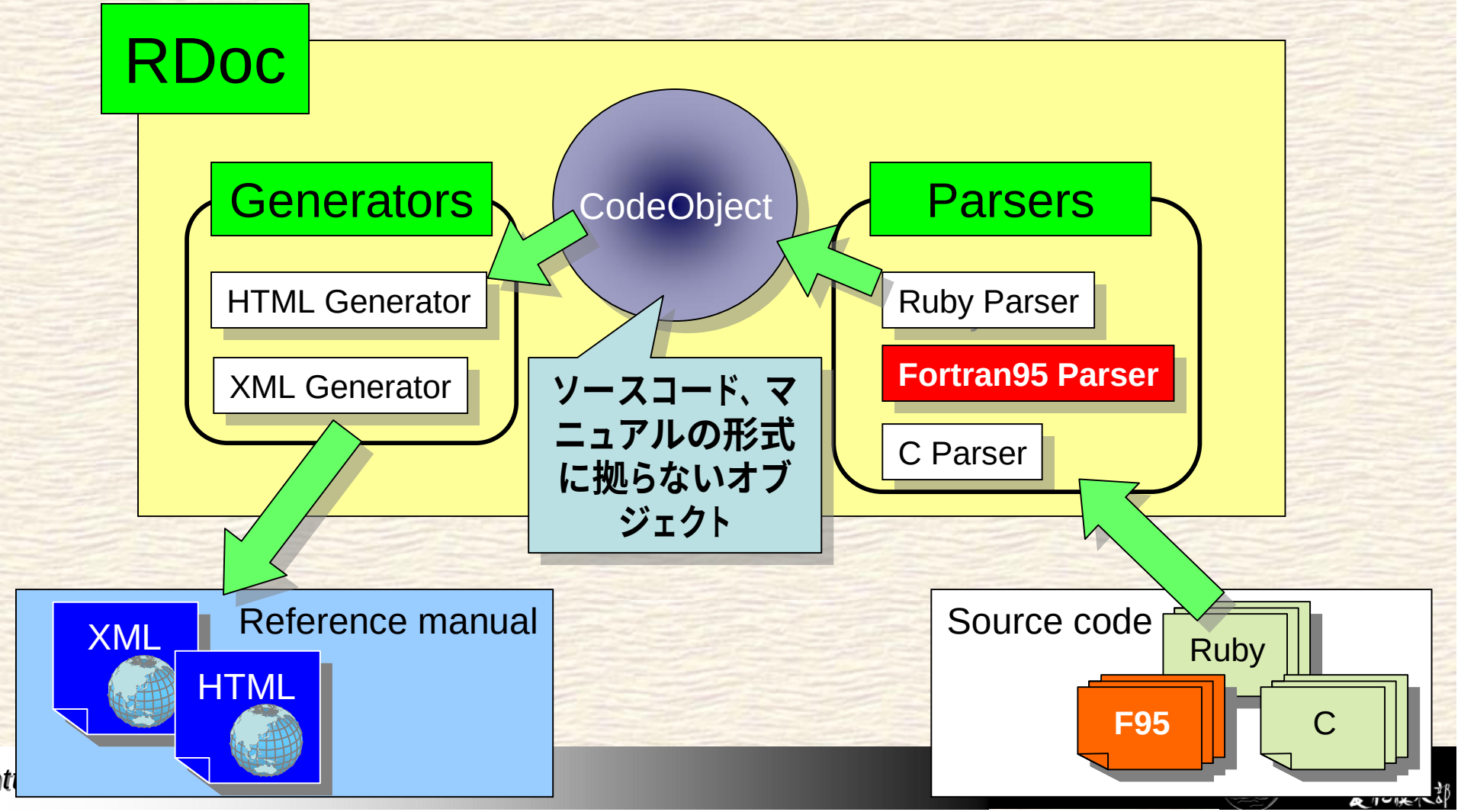
- 何が厄介かというと...
  - プログラムとマニュアルを別々に用意するのは面倒
  - プログラムで書いたものをもう一度書くのは面倒
- Ruby でのマニュアル生成
  - RDoc によるマニュアルの自動生成
- RDoc による数値モデルのマニュアル自動生成
  - RDoc の Fortran 90/95 の解析機能の強化

Ruby でのドキュメント生成

- リファレンスマニュアル作成には RDoc (Ruby Document System) を使用
- RDoc とは?
  - Ruby で書かれたソースコードから HTML、XML形式のマニュアルを自動生成する Ruby の標準ライブラリ (Dave Thomas 氏作成)
  - ソースコードを自動解析
  - ソースコードに記述されるコメントもマニュアルに反映
  - 自動生成される HTML 形式のマニュアルにハイパーリンクや見出し、箇条書きといった構造を付与することが可能 (そのために必要なタグも簡潔でソースコードを読む妨げにならない)
- RDoc により、リファレンスマニュアルの作成と維持管理が容易となる

RDoc の Fortran 90/95 への適用

- RDoc の作者の Dave Thomas 氏が Parser の一つとして Fortran 90/95 用のものを作成



Fortran95 Parser

- **Fortran90/95 の文法を解釈**
  - 別ファイル内のモジュール等へ自動的にリンク作成
  - タグを埋め込めば、目次、リスト、テーブルも表現可能

Fortran95 ソースコード

```
!= Module module_name_mod : Sample module
! Authors:: Yasuhiro MORIKAWA
!
! This module depends base_mod module
!
module sample_mod
  use base_mod
  implicit none
  private
  public :: sample_init, sample_end, Const

  real(8) :: Const = 3.14

  subroutine sample_init(inchar, outint)
    character(*)    , intent(in) :: inchar
    integer(INTKIND), intent(out):: outint
  end subroutine sample_init

  subroutine sample_end(err)
    logical, intent(inout) :: err
  end subroutine sample_end

end module sample_mod
```

```
!= Module ..    ①
! Authors:: ..
```

```
use base_mod    ②
      :
module module_..
      :
  subroutine mo..
      :
  end subrou..
end module modu..
```

① RDoc のタグは “=“ や “::” で表記

② module 文, use 文, subroutine 文を解釈。別ファイルのモジュールへのリンクを自動で作成

Fortran95 Parser

- **Fortran90/95 の文法を解釈**
  - 別ファイル内のモジュール等へ自動的にリンク作成
  - タグを埋め込めば、目次、リスト、テーブルも表現可能

Fortran95 ソースコード

```
!= Module module_name_mod : Sample module
! Authors:: Yasuhiro MORIKAWA
!
! This module depends base_mod module
!
module sample_mod
  use base_mod
  implicit none
  private
  public :: sample_init, sample_end, Const

  real(8) :: Const = 3.14

  subroutine sample_init(inchar, outint)
    character(*)    , intent(in) :: inchar
    integer(INTKIND), intent(out):: outint
  end subroutine sample_init

  subroutine sample_end(err)
    logical, intent(inout) :: err
  end subroutine sample_end

end module sample_mod
```

RDoc

HTML のリファレンスマニュアル

Files	Classes	Methods
sample.f90 base.f90	sample_mod base_mod	sample_init sample_end

Class sample_mod
In: sample.f90

Module sample_mod : Sample module
Authors: Yasuhiro MORIKAWA

This module depends **base_mod** module

Methods
sample_init sample_end

Included Modules
base_mod

Public instance methods

sample_init *(inchar, outint)*
[source]

sample_end *(err)*
[source]

ハイパーリンク

Fortran 90/95 解析機能の強化

- **Fortran 90/95 のリファレンスマニュアル自動生成ツールとして実用化**
  - **一通りの言語要素を解析**
  - Fortran **なので引数の型の情報もマニュアルに反映**
  - TeX **形式で記述したコメントを、コンパイルされた形式で表示可能とする (黒田拓氏の** Ruby **用** MathML **ライブラリを使用)**

一通りの言語要素と引数の解析

構造体の解析
要素、要素の型、解説の表示

一通りの言語要素と引数の解析

const()

Constant :

```
const = 3.14 : real(8), parameter
```

: 公開定数

sample_fun

Function :

定数の解析
型、初期値、解説の表示

[Source]

sample_init(*inchar, outint*)

Subroutine :

```
inchar :    character(*) , intent(in)
```

: 入力変数

```
outint :    integer(INTKIND), intent(out)
```

: 出力変数

初期化サブルーチン

[Source]

一通りの言語要素と引数の解析

const()

Constant :

```
const = 3.14 : real(8), parameter
```

: 公開定数

sample_func(*log*) result(res)

Function :

```
res :       logical
```

: 論理型の返り値

```
log :       logical, intent(in)
```

: 論理型入力変数

関数

[Source]

関数の解析
解説の表示

t(in)

tent(out)

初期化サブルーチン

[Source]

一通りの言語要素と引数の解析

const()

Constant :

```
const = 3.14 : real(8), parameter
             : 公開定数
```

sample_func(*log*) result(res)

Function :

```
res :     logical
          : 論理型の返り値

log :     logical, intent(in)
          : 論理型入力変数
```

関数

[Source]

sample_init(*inchar, outint*)

Subroutine :

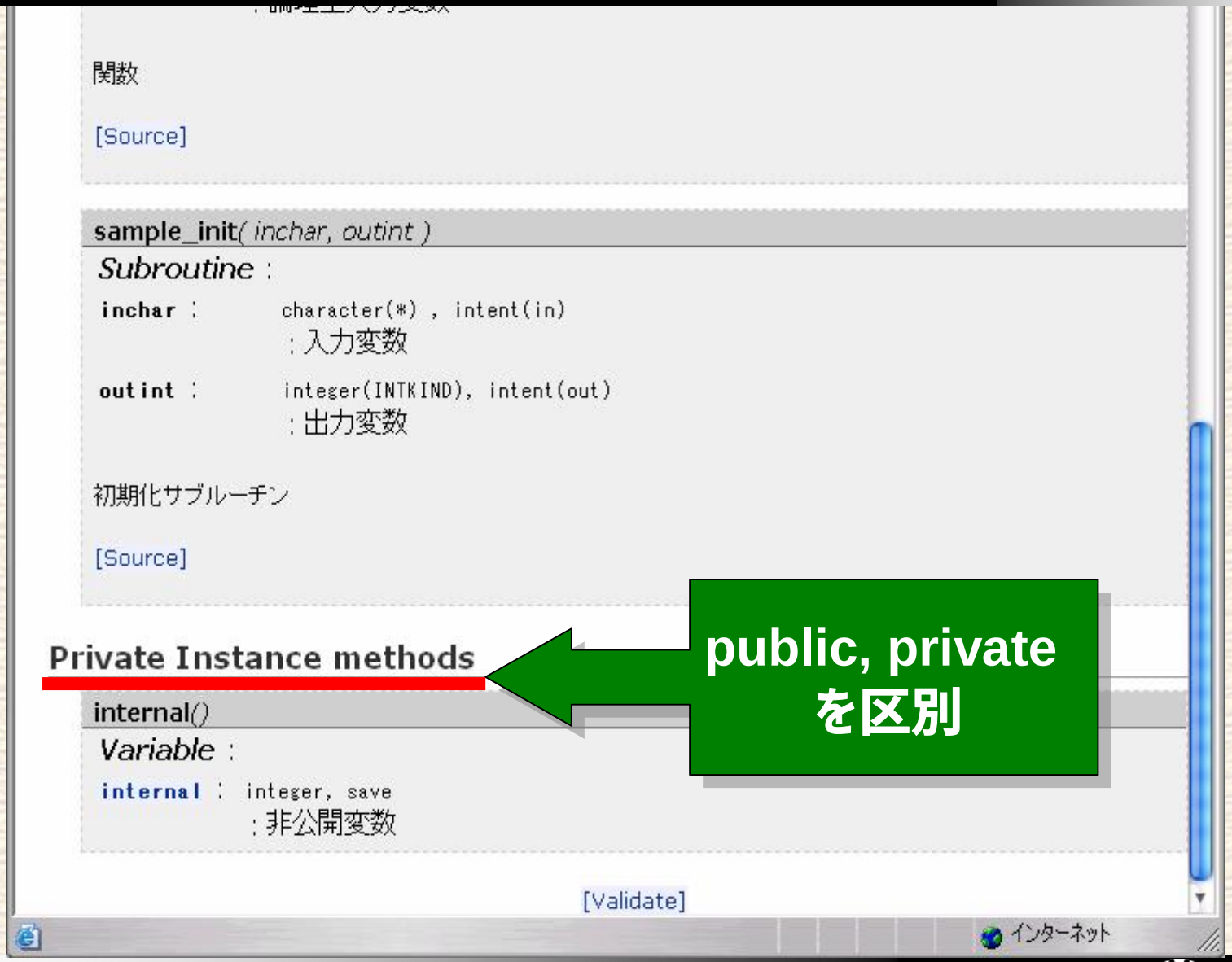
```
inchar :    character(*) , intent(in)
            : 入力変数

outint :    integer(INTKIND), intent(out)
            : 出力変数
```

初期化サブルーチン

[Source]

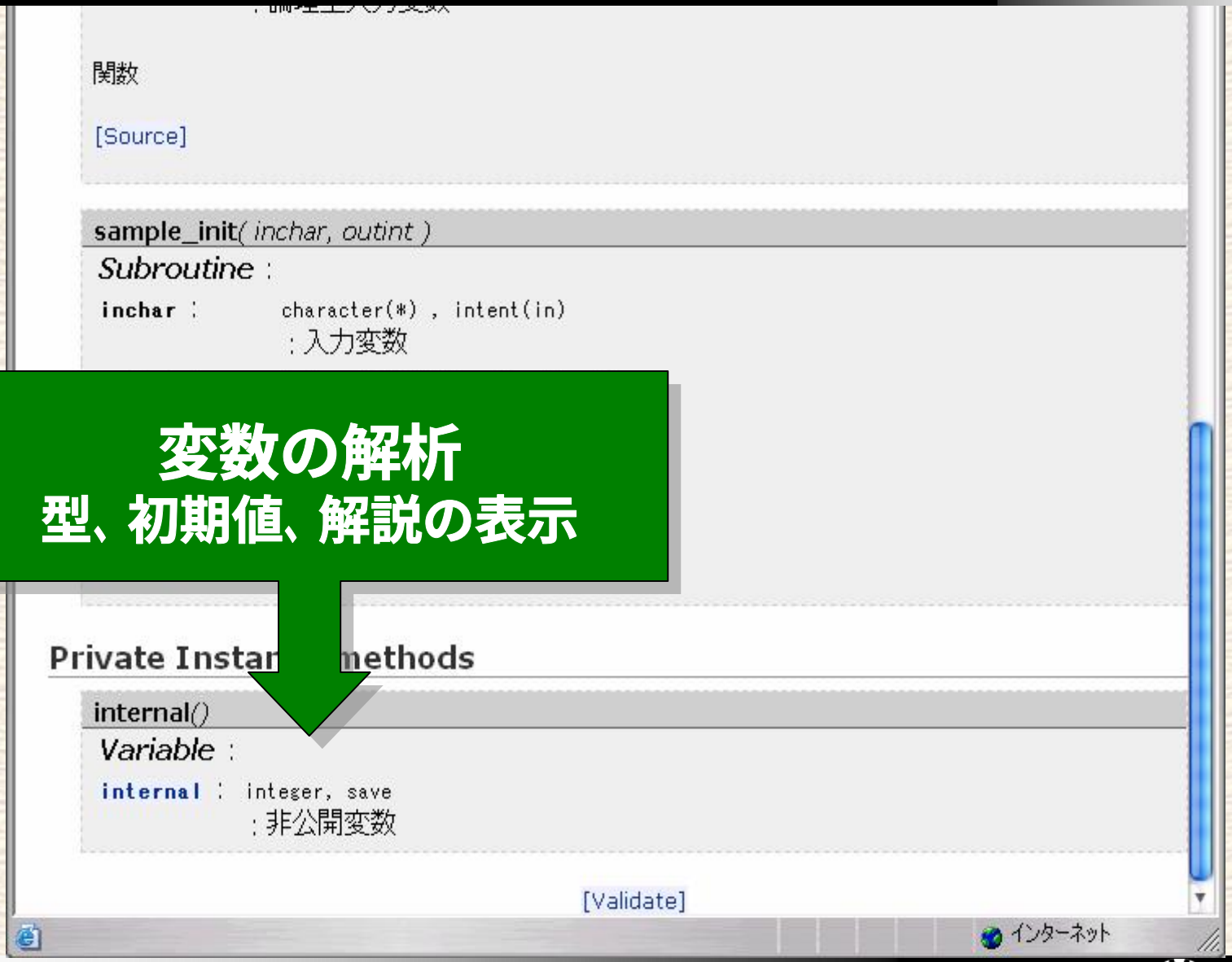
引数の解析
型、解説の表示

一通りの言語要素と引数の解析



public, private を区別

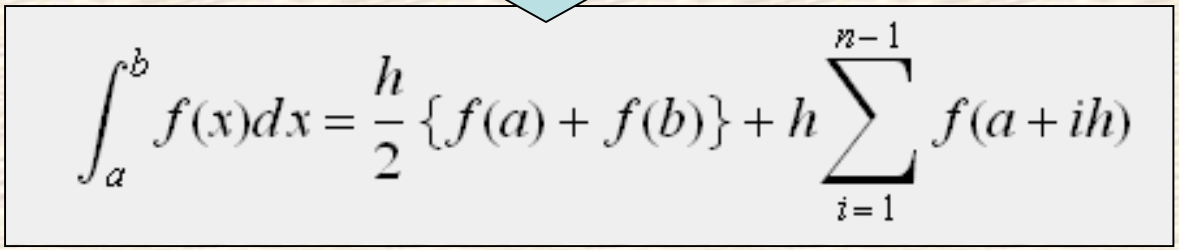
一通りの言語要素と引数の解析



TeX を数式として表示

- コメントに TeX 形式で記述した数式をMathML に変換

```
! \[
!  \int_{a}^{b} f(x) dx =
!     \frac{h}{2} \left\{ f(a) + f(b) \right\} + h\sum_{i=1}^{n-1} f(a+ih)
! \]
```



$$\int_a^b f(x)dx = \frac{h}{2}\{f(a)+f(b)\} + h\sum_{i=1}^{n-1} f(a+ih)$$

- Ruby 用 MathML ライブラリを使用
  - http://www.hinet.mydns.jp/?mathml.rb
- MathML 対応ブラウザが少なめなのが問題
  - Firefox、Internet explorer ぐらい…?

まとめ

- RDoc Fortran 90/95 解析機能強化版により、数値モデルのリファレンスマニュアル生成が簡単に
- 公開URL
  - http://www.gfd-dennou.org/library/dcmodel
    - パッチを当てたパッケージも配布してます
    - コメントの書法等の解説あります
- 使用例
  - http://www.gfd-dennou.org/library/gtool4
  - http://www.gfd-dennou.org/library/dcpam
  - http://www.gfd-dennou.org/library/deepconv

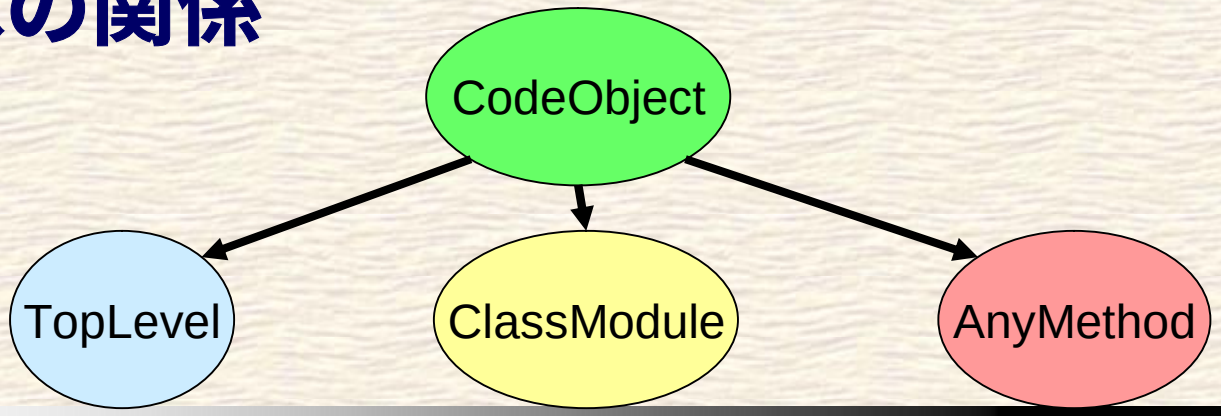
参考資料

- **数値モデリングプロジェクト dcmodel**
  - http://www.gfd-dennou.org/library/dcmodel/
- **森川靖大, 石渡正樹, 堀之内武, 小高正嗣, 林祥介, 2007: RDoc を用いた数値モデルのドキュメント生成. 天気, 54, 185--190.**
  - http://s-ws.net/tenki/cont/54_02/co.html
- **オブジェクト指向スクリプト言語 Ruby**
  - http://www.ruby-lang.org
- **Ruby 用 MathML ライブラリ**
  - http://www.hinet.mydns.jp/?mathml.rb
- **MathMLによる数学的文書の作成と公開のシステム（講演者: 黒田 拓氏）**
  - http://www.hinet.mydns.jp/presentation/hiki.cgi?MathML2006.03
- **Fortran からドキュメントを自動生成するツール**
  - f90tohtml
    - http://mensch.org/f90tohtml/
  - f90doc
    - http://theory.lcs.mit.edu/~edemaine/f90doc/

付録

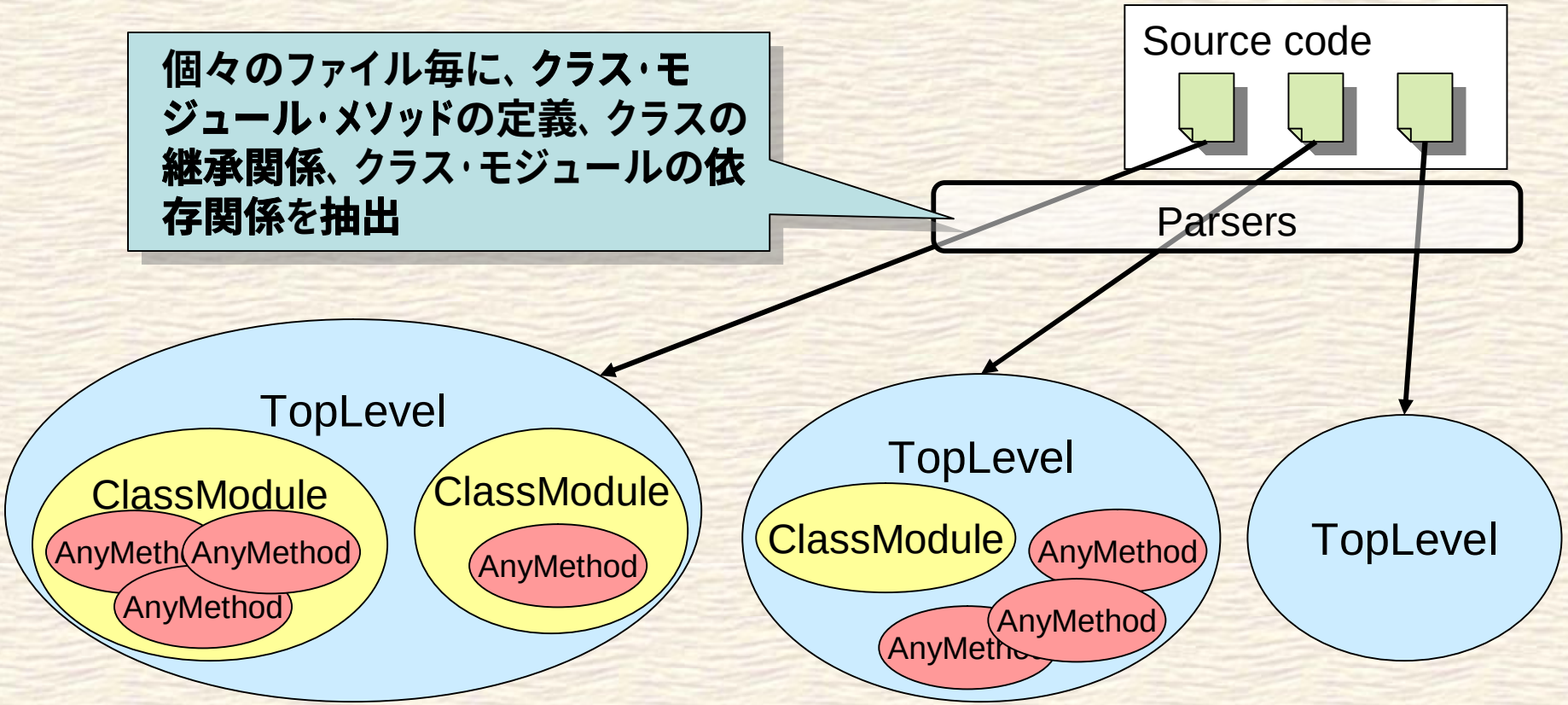
CodeObject

- ソースコード内の情報を階層的に保持するためのオブジェクト
  - CodeObject：共通する情報
  - TopLevel：ファイルの情報
  - ClassModule：クラス、モジュールの情報
  - AnyMethod：メソッドの情報
- 継承の関係



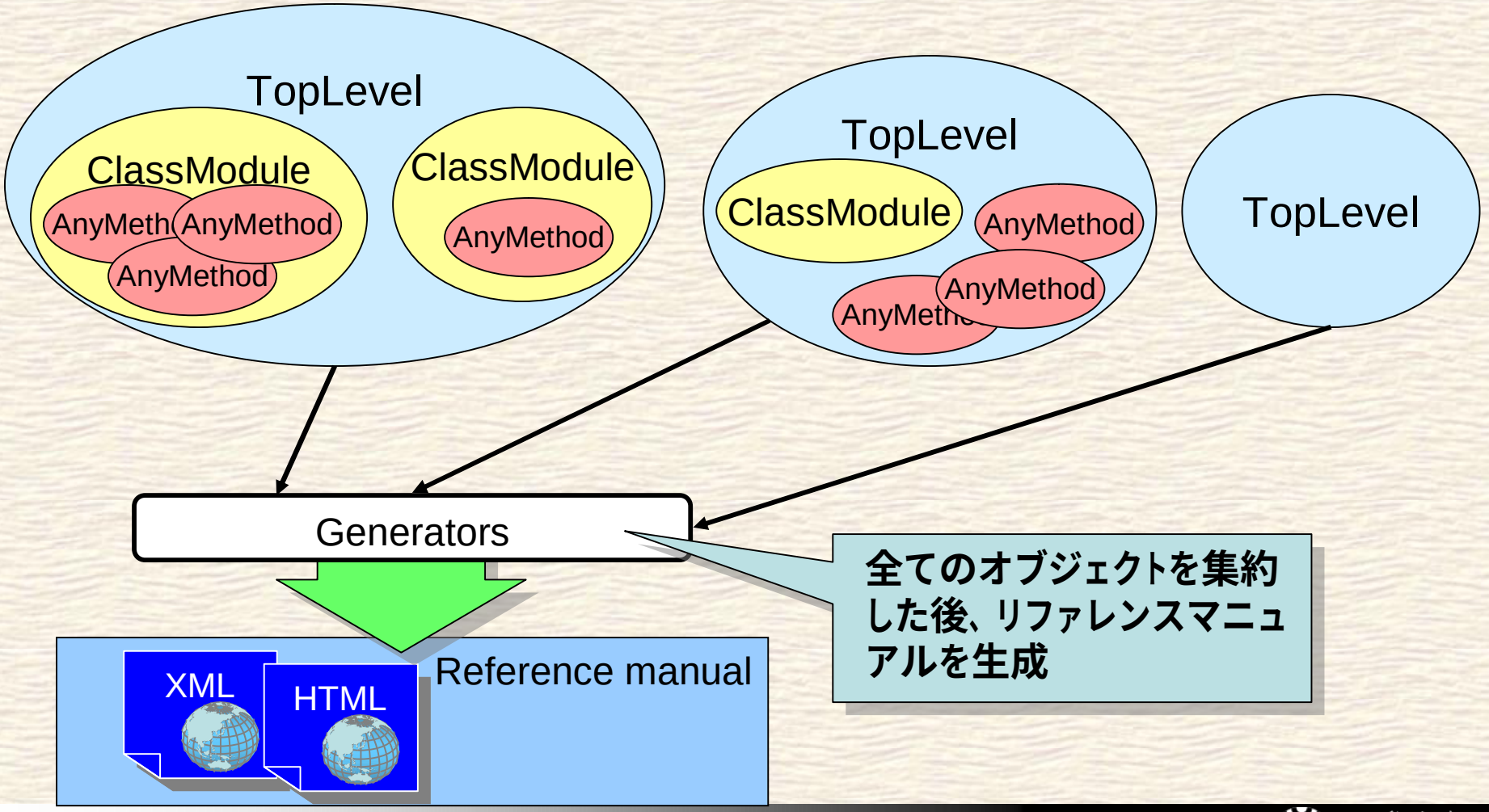
ソースコード ⇒ CodeObject

- ソースコード内の情報を TopLevel, ClassModule, AnyMethod へ



CodeObject ⇒ マニュアル

- HTML 等のドキュメントを生成



Fortran95 Parser (強化版)

■ 解析機能の強化の具体例

Fortran95 ソースコード

```
!= Module sample_mod : Sample module
! Authors:: Yasuhiro MORIKAWA
!
module sample_mod
  use base_mod
  !
  ! sample_mod の概要
  implicit none
  private
  public :: sample_init, sample_func, const, TYPE_A
  private:: internal

  type TYPE_A
    ! 構造体の解説
    integer :: counter    ! 構造体内部の変数の解説
  end type TYPE_A

  real(8), parameter :: const = 3.14   ! 公開定数
  integer, save      :: internal       ! 非公開変数

  subroutine sample_init(inchar, outint)
    ! 初期化サブルーチン
    character(*)     , intent(in) :: inchar  ! 入力変数
    integer(INTKIND), intent(out):: outint  ! 出力変数
  end subroutine sample_init

  function sample_func(log) result(res)
    ! 関数
    logical, intent(in) :: log  ! 論理型入力変数
    logical             :: res  ! 論理型の返り値
  end function sample_func
end module sample_mod
```

```
public:: samp...①
private:: inte...
```

```
type TYPE_A ...  ②
```

```
real(8), parame..③
Integer, save..
```

```
! 初期化...      ④
```

```
char..     ! 入力.⑤
integer..  ! 出力..
```

```
function samp...⑥
        :
end function ..
```

① 公開要素と非公開要素の区別

② 構造体

③ 公開定数、公開変数

④ サブルーチン、関数のコメント

⑤ 引数の型、コメント

⑥ 関数

Fortran95 Parser (強化版)

- **解析機能の強化**
  - 解析可能になった要素のリスト
    - 関数 (function 文)
    - サブルーチンや関数の引数の型
    - モジュールが公開する変数, 定数
    - 構造体 (type 文)
    - NAMELIST 文
    - 利用者定義演算子 (operator), 利用者定義代入 (assignment)
    - 上記要素のコメント文
    - 総称手続き (interface 文)
    - 公開要素と非公開要素との区別
    - 孫引きされている公開要素
  - 大文字小文字の違いを無視