DateRGF: From REXX to Java to Object REXX

Classes for Date and Time Calculation
By
Michael Warmuth, Vienna, Austria

REXX Symposium 2004 - Sindelfingen, Germany

Overview

- What are date and time calculations good for?
- History of DateRGF
- What way to go?
- A look at the code
- The classes
- Examples for use
- Outlook and time schedule
- Contact addresses

What are date and time calculations good for? (1)

What's the date of Easter Monday in 2005

```
call date.cmd
d = .Date~easter(.Date~new(2005,1,1))
say d~tostring
d~add(1)
say d~tostring
say .Date~getString(.DTC~DN, d~get(.DTC.DOW))
```

What are the dates of our Jour Fixe at every last Thursday in '04?

```
do m=1 to 12
   d=.Date~new(2005,m+1,1)
   d~add(-1)
   d~setGivenWeekday(-5,d)
   say d~tostring
end
```


What are date and time calculations good for? (2)

What's the time difference between two times?

```
t1=.Time~new(19,29,39)              /* 19:29:39 */
t2=.Time~new( 8, 8, 8)              /* 08:08:08 */
diff=t1~subtract(t2)
say diff                            /* 0.47327545 */
say .Time~valueOf(diff)~tostring    /* 11:21:31 */
say .Time~valueOf(-diff)~tostring   /* 12:38:29 */
```

History of DateRGF

- DateRGF written by Rony G. Flatscher:
  - It's written in REXX
  - First version of 1991-05-20
  - Continued development (at least) until 1996-04-30
- DateRGF for Java and Waba
  - It's written in Java (Waba) languages
  - First version of 2001-04-05
  - Continued development (at least) until 2001-04-05
- DateRGF for Object REXX
  - It's written in Object REXX
  - First version of (very near future)

Which way to go? (1)

- Possible starting points:
  - Classic REXX version
  - Version for Java
- The main goals:
  - Object oriented applying Object REXX's philosophy
  - Compatibility with the version for Java
- Development platform:
  - OS/2 Warp

Possible starting points - Which way to go? (2)

- Classic REXX version
  - Similar syntax
  - Procedural vs. object oriented philosophy
- Version for Java
  - Object oriented
  - Different syntax

==> Version for Java as starting point

Main goals - Which way to go (3)

- Object oriented applying Object REXX's philosophy
  - Java fields vs. OOREXX attribute methods
  - OOREXX class and instance methods
  - Java constructors vs. OOREXX new and init methods
- Compatibility with the version for Java
  - Solutions for overcoming different syntax
  - Redundancy by using instances as parameters
- Personal preferences
  - "Early returns" vs. using return variables

Development plattform - Which way to go (4)

- OS/2 Warp

A look at the code (1)

- Java fields simulated using SETMETHOD
- Which methods to implement as class methods?
- Class method NEW
- A cool solution?

Java fields simulated using SETMETHOD - A look at the code (2)

Java:

```
public final static int MILLIS_PER_SECOND = 1000;
```

OOREXX:

```
::method MILLIS_PER_SECOND attribute class
...
::method init class
   ...
   .self~MILLIS_PER_SECOND = 1000
   ...
```

I would like to have something like:

```
::method init class
   ...
   self~<some_method>('MILLIS_PER_SECOND',1000,'public')
```


Java fields simulated using SETMETHOD - A look at the code (3)

```
::method setmethod
   use arg methodname, value, public

   code = .array~new
   code[1] = 'expose' methodname
   code[2] = 'use arg' methodname
   putmethod = .method~new(methodname || '=',code)

   if translate(public) = 'PRIVATE' then do
      putmethod~setprivate
   end /* if */

   self~setmethod:super(methodname || '=',putmethod)

   /* Set the initial value */
   .message~new(self,methodname ||,
                     '=','individual',value)~send
```

Java fields simulated using SETMETHOD - A look at the code (4)

```
::class DTC public
   ::method method class
      use arg methodname
      return .methods[translate(methodname)]

   ::method init class
      self~setmethod('setmethod',,
                              self~method('setmethod'))

      self~setmethod('MILLIS_PER_SECOND',1000,'public')
```

Accessing the data:

```
say .DTC~MILLIS_PER_SECONDS                /* 1000 */
```


Java fields simulated using SETMETHOD - A look at the code (5)

The simple alternative:

```
::method init class
   .local['MILLIS_PER_SECOND']=1000
```

The better alternative:

```
::method init class
   .local['DTC.MILLIS_PER_SECOND']=1000
```

Accessing the data using the better alternative:

```
say .DTC.MILLIS_PER_SECONDS                /* 1000 */
```


Which methods to implement as class methods? - A look at the code (6)

- **Methods holding static data (constants), eg.:**
  - **Array for names of weekdays**
  - **Array for names of months**
- **Methods returning an new instance; eg. for Date:**
  - **(new)**
  - **easter**
  - **valueOf**

Class method NEW - A look at the code (7)

- **Class method NEW is used for:**
  - **Initialization of attributes using instance objects**
  - **(Creation of instances)**

A cool solution? - A look at the code (8)

The problem:

```
::class Date
   ::method aDate attribute class
   ::method init class
      self~aDate=self~new    /* Does not work */
```

A solution:

```
::method aDate attribute class
::method firstinstance attribute class
...
::method init class
   self~firstinstance = .true
   ...
::method new class
   if self~firstinstance then do
      self~firstinstance = .false
      self~aDate=self~new    /* This does work but... */
```


The classes (1)

- DTC (Date/Time Constants)
- Date (Date calculation)
- Time (Time calculation)
- DateTime (DateTime calculation)
- DateFormat (Formatting of date and time strings)
- DateTestRgf (Testing if dates or times are valid)

DTC - The classes (2)

- Storage container for a lot of integer numbers
- Addressed using class methods: .DTC~<METHOD>

Date - The classes (3)

- Creation of dates
- Subtracting, adding dates or days
- Getting string information, eg. weekday, month name
- Conversion to/from Julian day numbers
- Calculating with EPOCH dates (eg. Java, Palm)
- Getting next/last specific day of week
- much more

Time - The classes (4)

- Subtraction of times
- Conversion to/from numbers (fraction of day)
- Comparison of times

DateTime, DateFormat, TestDateRgf - The classes (5)

- DateTime:
  - Combination of date and time calculation
- DateFormat:
  - Formatting of date and time strings using patterns
- TestDateRgf:
  - Tests if a date or a time is correct

Some examples

- Calculating calendars
- Calculating schedules based on time differences
- Date and time calculation for eg. billing
- Cron like tool using the ALARM class
- ...

Outlook and time schedule

- What has to be done?
  - Porting the missing classes
  - Resolving the "issues"
  - Cleaning up the code
  - Porting the documentation
  - Providing some examples
- When will it be available?
  - Completion of coding until end of May
  - Testing and creating some examples until 15th of June
  - Documentation until end of June

Contact address

Michael Warmuth
michael@warmuth.at

http://www.warmuth.at/rexxla/datergf