

Flexible SAS®

Daniel Olguin, First Coast Service Options, Jacksonville, FL

ABSTRACT

In 2007 Adobe® Inc. open-sourced it's Flex® SDK. The Flex SDK gives developers the ability to develop RIA applications for the Flash environment. Combining the rich user interface capabilities at which Flash excels, and the data access and statistical capabilities of SAS®, can give the end user great power in a pleasing experience. The interface capabilities of Flex include highly interactive business graphics and dashboards. This paper will briefly explore the combination of Flex and SAS with a small demonstration application mimicking some of the basic functionality of SAS Enterprise Guide®.

Keywords: Enterprise Guide, open-source, Ruby, JRuby, Flex

INTRODUCTION

There are those in business that believe that business is itself a zero-sum game. In order for them to have more, someone else must have less. Most proprietary software is built with that assumption. There is another model which believes that if you create the right environment, an ecosystem flourishes providing more for everyone. The open-source movement is based on this model and the success of frameworks such as Eclipse™ and programming languages such as Ruby attest to the efficacy of the model.

IBM® has successfully mixed open-source with proprietary software to redefine itself. Its success shows that a company can not only survive but also flourish while still contributing to the greater good.

It's my belief that SAS, and more importantly SAS customers, can benefit from a thriving open-source community. My papers at SESUG are motivated by this belief. This paper will highlight one such example, Enterprise Gateway, an open-source substitute for SAS Enterprise Guide.

FLEX

The Flex SDK is Adobe's open-source contribution to the development community. The SDK includes a full complement of gui widgets as well as classes for event notification, state transitions, and OO development. The applications produced run in the Flash browser add-in or the new Adobe AIR® RIA platform. The primary language is ActionScript which is an implementation of ECMAScript, aka Javascript. Adobe sells two versions of its Flex IDE which run on the Eclipse framework which are reasonably priced and offer a pleasing development environment.

As in any well supported open-source offering, a rich ecosystem has evolved around the product. The Web is full of examples on personal and company blogs of Flex code from simple widgets to full applications. Finding information you need for your projects is a google search away.

The limiting factor in the browser based version is the standard sand-box nature of web applications. Restrictions on access to the desktop, motivated looking for another solution. Adobes AIR platform was released too late to be a contender for this SESUG event. Maybe next year!

JRUBY

Although Flex was used to prototype the ideas associated with an open-source alternative to Enterprise Guide, JRuby was the primary tool for implementing a more full-featured implementation. JRuby is an open-source implementation of the Ruby programming language which runs on top of the Java® JVM. As such, JRuby has access to the entire world of open-source Java projects. Running on the JVM means that JRuby's world covers most OSes on most hardware platforms from mobile devices to mainframes. SAS has a long history of integrating Java into its products including its BI server platform. JRuby simplifies the development of applications and makes an excellent glue-code for piecing together disparate projects.

It's hoped that JRuby's accessibility, as well as its pure Java underpinnings, will make it easier for adoption. Another motivation was the desire to provide affordable (read free) choices. Choice was a primary driver (note the splash screen on the next page). The power to choose what, how, where and when, are never far from my mind when dealing with proprietary software and its restrictions, limitations, cost, and releases which are slow and too often behind schedule.

ARCHITECTURE

The architecture of the Enterprise Gateway is composed of three parts:

- 1) The front-end which can be web based as in the Flex prototype, or client based as in the JRuby version.
- 2) The middle-tier which is a JRuby application providing distributed communication as well as access to SAS
- 3) The base SAS product
 - a. It's assumed you have licensed, at a minimum, Base SAS.



Figure 1. A screen shot of the Enterprise Gateway splash screen

The JRuby version of the application is released under the Creative Commons® share-alike license in the hopes that it will stimulate ideas and future open-source works involving integration with SAS software. Note that the splash screen pictured above is itself a reworking of the official Ruby logo (used here to visually represent powered by Ruby) which is itself released under the Creative Commons share-alike license!

Enterprise Gateway has been tested with JRuby 1.0 through 1.1.2 , Java 5 (aka Java 1.5), and SAS 9.1.3 (both the standard version and the SAS Learning Edition 4.1).

The Front-end

The front-end mimics in general the layout of SAS Enterprise Guide which follows a standard framework layout also seen in Eclipse. It includes a menu, toolbar, and several views surrounding the central workspace which is the traditional home of editors used to create and modify application resources.

On the left is the Project view in standard tree outline used for navigation of the project. A project contains zero or more process flows which contains zero or more tasks and/or resources. Tasks in Enterprise guide always involve some SAS execution but in our open-source version a task can be anything executable. One of the results of open-source thinking is a broadening of horizons.

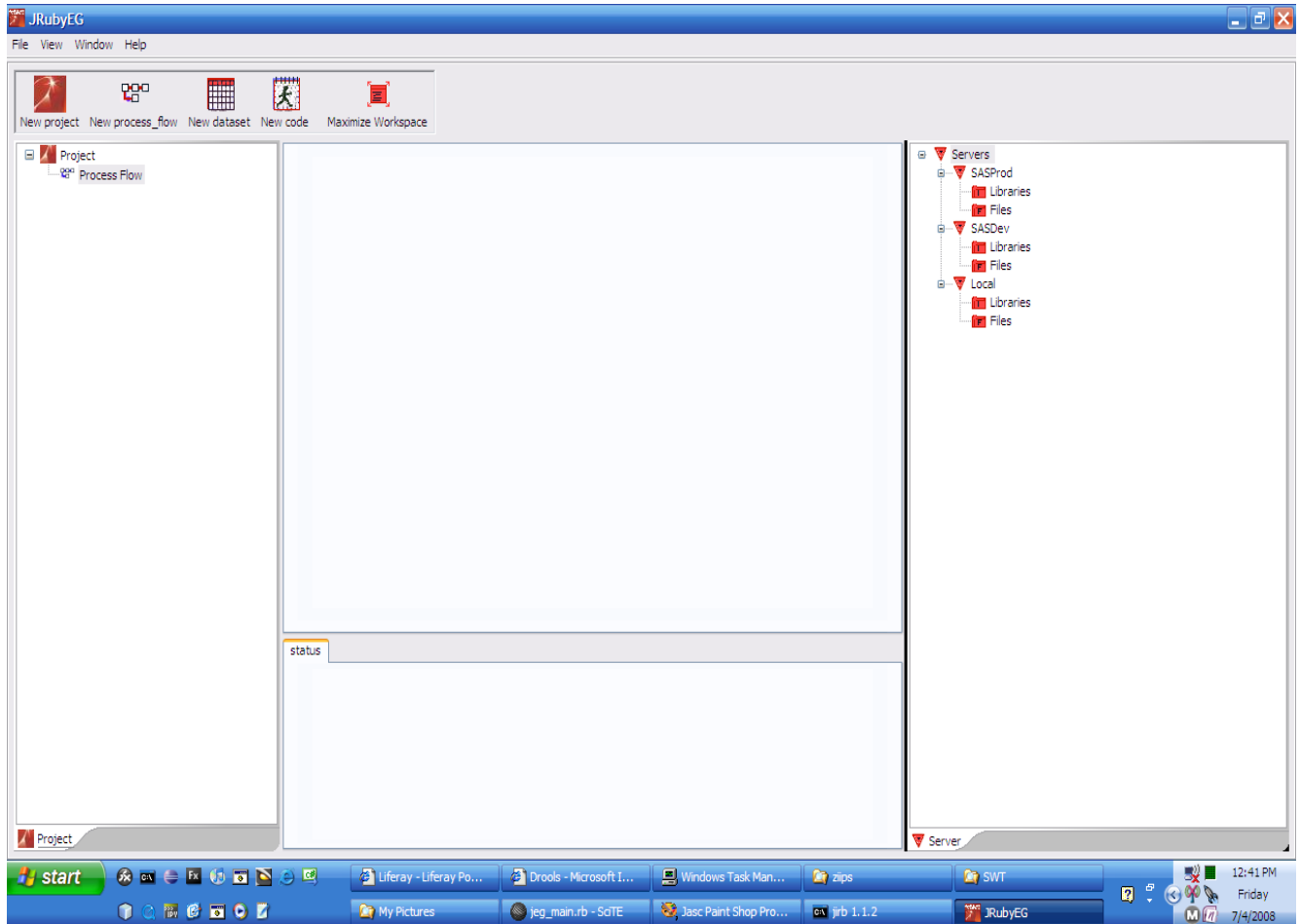


Figure 2. A screen shot of the Enterprise Gateway application

The view on the far-right is the Server view which shows the SAS Servers available the libraries associated with the server and the files. In general the far-right view area is for supporting Context information which includes servers but may also include things like properties.

The view at the bottom is for notification and miscellaneous interaction. One such miscellaneous item is an interactive ruby session allowing the user to type ruby code and get an immediate response. This follows the standard ruby practice of providing an IRB (interactive ruby) tool which has been repeated in the JRuby community with JIRB.

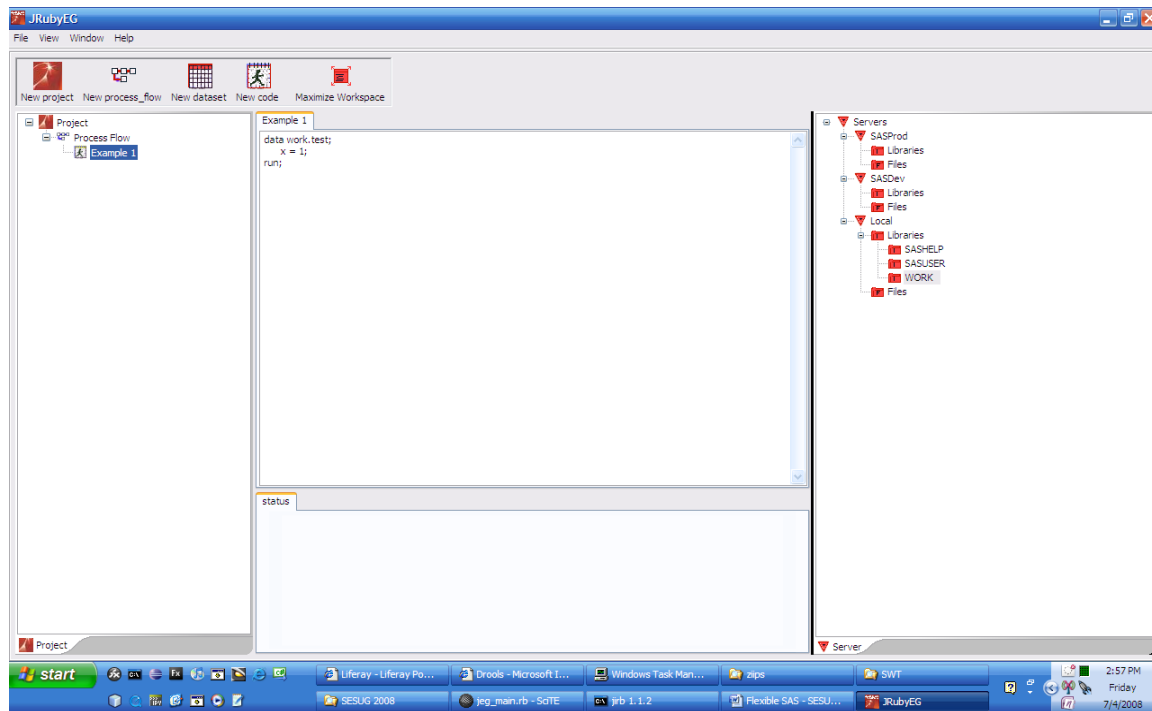


Figure 3. The application connected to Local with Task, Example 1, open in the workspace.

The central area is the workspace where tasks and/or resources can be viewed and edited. Actions in the workspace and the project view are synchronized as expected. The views can be resized by standard methods of dragging or they may be closed and opened by menu commands. A common user interaction is to maximize the workspace to avoid scrolling of long lines and this capability is available both on the menu and toolbar.

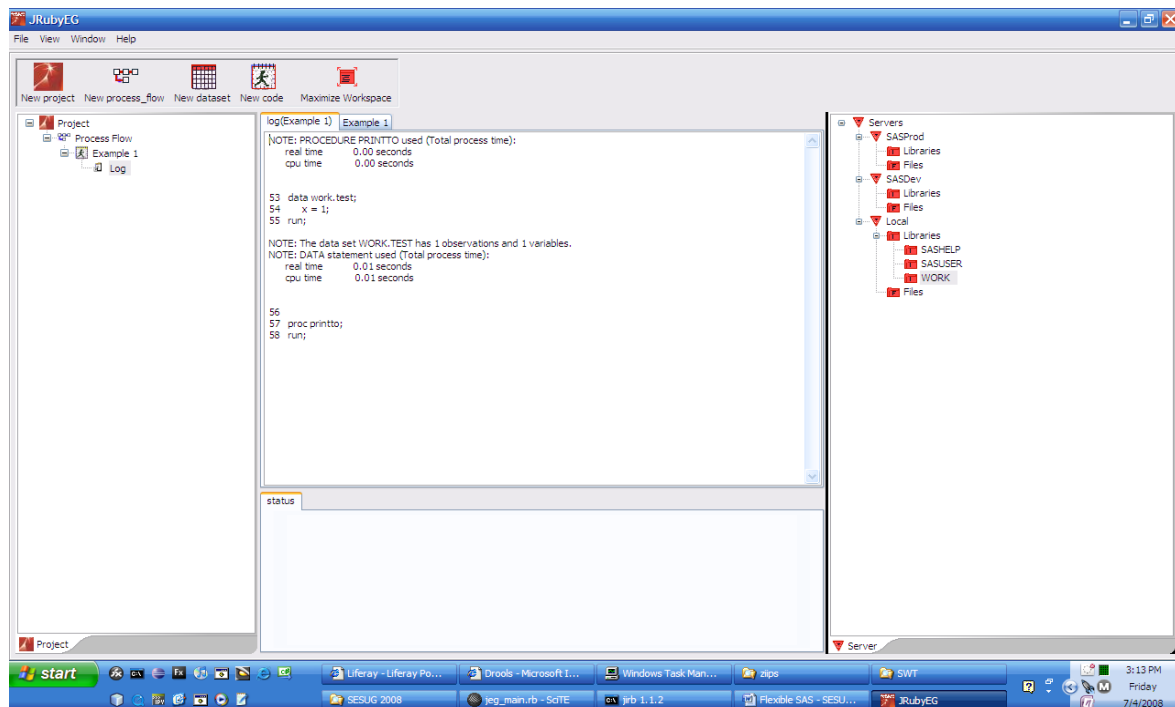


Figure 4. The log from running Example 1.

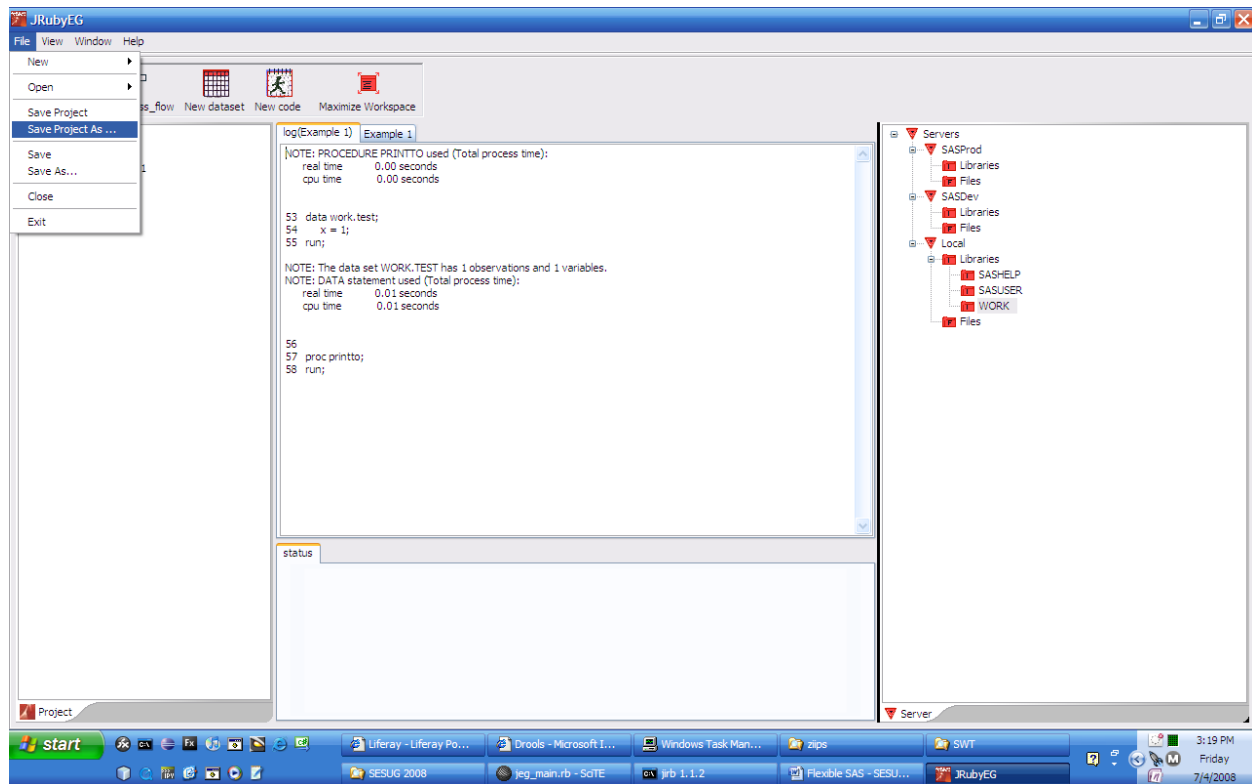


Figure 5. Enterprise Gateway supports standard menu actions.

The project can be saved in several formats, none of them proprietary, but the default format is YAML a text based serialization format which is both machine and human readable and which is used routinely in Ruby applications. Since the serialization format is text based, standard text compression utilities can be used to archive and transport JRuby Enterprise Gateway projects. A partial listing of a .jeg project in YAML follows.

```

--- &id001 !ruby/object:Project
name: Project
sub_tasks:
- &id002 !ruby/object:ProcessFlow
  name: Process Flow
  parent: *id001
  resources: []
  sub_tasks:
  - !ruby/object:Task
    name: Test
    server: Local
    log: "NOTE: PROCEDURE PRINTTO used (Total process time):\r\n      real time
\

```

Interfacing with the Middle-Tier

The interface to the middle-tier is handled by a JRuby proxy class. Ruby/JRuby provides for an elegant implementation of the Proxy pattern. Ruby classes/objects support a `method_missing` method which makes for easy development of proxies. If an object is sent a message it doesn't support then the message and its parameters are sent to the `method_missing` method which can resend to the 'real' proxied object.

The Strategy pattern is used to determine which class does the actual communication with SAS and how. For example, if you have SAS Integration Technologies installed and licensed you can use the `SASIOMWorkspaceManager` JRuby class which, with the help of the IOM jar files, to communicate with your IOM workspace server.

However, since both Ruby and Java already have excellent distributed communications capabilities you can still use Enterprise Gateway even if you only have base SAS installed. `SASOLEWorkspaceManager` can be used in a local Windows environment (as used in developing this paper) or `SASBatchWorkspaceManager` in a distributed environment.

```
sas.rb (partial listing)

class SASWorkspaceManagerProxy
  attr_accessor :model
  attr_reader :aka, :host, :port

  def initialize(aka, host, port, type='IOM')
    ...
  end
  def connect(user, password)
    ...
    case
      when @type.upcase == 'IOM' then
        @mgr = SASIOMWorkspaceManager.new(@host, @port, @user,
@password
      when @type.upcase == 'OLE' then
        @mgr = SASOLEWorkspaceManager.new(@host, @port, @user,
@password)

    end
    ...
  def method_missing(name, *args)
    ...
  end
end
```

The Middle-Tier

The middle-tier is handled by JRuby using DRb, a standard Ruby distributed communications capability or through Web Services in standard Java format (but hidden by JRuby façade) or by a combination of the two. For a more complete description of the Web Services technique see my SESUG 2008 paper entitled, 'Simple SAS Web Services with Ruby'.

The partial listing below shows the SASIOMWorkspaceManager initializing for communications with the IOM servers.

```

sas.rb (partial listing)

class SASIOMWorkspaceManager
  attr_accessor :output, :log, :lang_service, :data_service
  def initialize host, port, user_name, password
    @output = ''
    @log = ''
    @properties = java.util.Properties.new
    @properties.put "host", host
    @properties.put "port", port
    @properties.put "userName", user_name
    @properties.put "password", password
    @string_writer = java.io.StringWriter.new
    @print_writer = java.io.PrintWriter.new(@string_writer, true)
    prop_ary = [@properties].to_java(java.util.Properties)
    @factory = SAS::WorkspaceFactory.new(prop_ary,
    java.util.Properties.new(), @print_writer)
    begin
      @workspace = @factory.createWorkspaceByServer(@properties)
      puts "Connected!"
      @lang_service = @workspace.LanguageService
      @data_service = @workspace.DataService

      declare_libs(user_name, password)
      self
    rescue Exception => e
      puts e.to_s
    end
  end
  ...
end

```

CONCLUSION

Open-source software provides us the tools to create the applications we need with the features we want on the schedule of our choosing. A strong, thriving open-source community would benefit both SAS and its customers. It is hoped that Enterprise Gateway is seen as a step in that direction.

REFERENCES

Adobe Flex 3 at <http://www.adobe.com/products/flex/>

Creative Commons License at <http://creativecommons.org/licenses/by-sa/3.0/>

JRuby homepage <http://jruby.codehaus.org/>

Eclipse homepage <http://www.eclipse.org/>

SAS support Website

http://support.sas.com/rnd/itech/doc9/dev_guide/dist-obj/javacInt/javaprogram/workfact/workfact_direct.html

see also the titles in the Recommended Readings list

RECOMMENDED READING

The Ruby Programming Language, David Flanagan & Yukihiro Matsumoto, O'Reilly Media, Inc. 2008

Ruby Cookbook, Lucas Carlson & Leonard Richardson, O'Reilly Media, Inc. 2006

Design Patterns in Ruby, Russ Olsen, Addison-Wesley Inc., 2008

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Daniel Olguin
2408 Spring Vale RD
Jacksonville, FL 32246
E-mail: olguind@bellsouth.net

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

Complete Program Listings may be downloaded from <http://olguind.clearwire.net/downloads>