Exploring Ruby and Java Interoperability for Building Converged Web and SIP Applications

Edin Pjanić and Amer Hasanović

University of Tuzla, Faculty of Electrical Engineering
Franjevačka 2, 75000 Tuzla, Bosnia and Herzegovina
{edin.pjanic,amer.hasanovic}@untz.ba

ABSTRACT

In this paper we present a Ruby infrastructure that can be used for rapid development of Web applications with SIP signaling capabilities. We construct this infrastructure by combining the Java based Cipango SIP/HTTP Servlet Application Server with the Ruby on Rails Web development framework. We provide detailed explanations of the steps required to build this infrastructure and produce a realistic SIP application with an integrated Web interface. The described infrastructure allows Ruby applications to utilize the entire functionality provided by the SIP Servlet API and can be used as a good starting point for the development of Ruby-based domain specific languages for the SIP protocol. We also compare the proposed infrastructure with the existing Ruby frameworks for SIP application development. Furthermore, we present the performance analysis results of the demo application using SIPp, a SIP traffic generator tool.

KEYWORDS

software infrastructure, SIP, converged applications, dynamic languages, interoperability, Ruby, JRuby, Rack, Cipango, Jetty

1 INTRODUCTION

In the nineties, software development was mostly focused on desktop applications utilizing tools, libraries and paradigms built around Java and C++, the two most dominant programming languages of that time. During the last decade, this focus started shifting towards Web applications, agile practices and dynamic programming languages [1][2][3].

The Ruby programming language [4] received a lot of attention in the software industry, primarily because of the flexible Web frameworks, such as Ruby on Rails [5] and Sinatra [6], as well as Ruby's support for the development of domain specific languages (DSL) [7][8].

With the recent rise of social Web applications, there is an increased interest to bring other services to the Web domain, real-time voice and video in particular. Based on the HTTP model, the Session Initiation Protocol (SIP) [9] was developed to support these services. The telecommunications industry has accepted the SIP protocol, which is now widely implemented and used in telecom application servers.

While Ruby has excellent support for HTTP, due to the Web frameworks, it lacks the adequate support for SIP. On the other hand, Java has well developed APIs, libraries and tools for the SIP protocol, such as the SIP Servlets [10] and JAIN SLEE [11]. Therefore, it would be beneficial to combine Ruby's proven agility [3] and Java robustness in a single infrastructure for rapid development of

API and control all the features provided by the Cipango application server, including the creation and specification of listener objects, as shown in the demo application. Furthermore, this gives us a possibility to develop and integrate Ruby DSLs in order to speed up and simplify VoIP applications development.

Moreover, the lightweight infrastructure of the Cipango engine allows faster application startup times, which is very important in short development iteration cycles.

2.2 Sipatra

Sipatra [18] is a Ruby DSL for SIP Servlets and targets SIP Servlet 1.1 compatible application servers. Its syntax is much like the syntax used in Sinatra Web DSL, but adapted to support the specific features of the SIP protocol.

Sipatra SIP message handlers are defined as blocks of code that are called when certain conditions are satisfied. The conditions are specified as regular expressions that must match the message's SIP request method, response code, URI or headers in order for the handler to be invoked. There can be more than one handler for the same SIP message type. The handlers are evaluated in the order they appear in the code and the first handler that matches the condition is executed. An example of the syntax is shown in the Figure 1 listing that is self explanatory.

Similar to our approach, Sipatra uses the Cipango application server. However, the application deployment procedure is similar to the one used in the TorqueBox project. Hence, it suffers from the same limitations discussed in Section 2.1.

```
invite /sip:.*@sipatra.org/ do
    ...
end

invite /sip:.*@otherdomain.org/ do
    ...
end

invite do
    ...
end

response :INVITE, 200 do
    ...
end
```

Figure 1. Example of the Sipatra syntax.

3 JRUBY AND JAVA INTEROPERABILITY

Matz's Ruby Interpreter or Ruby MRI is the reference implementation of Ruby, written in C by Yukihiro Matsumoto. However, this is not the only implementation of Ruby. One especially successful open source implementation, called JRuby, is written for the Java Virtual Machine. While being fully compatible with Ruby MRI, it offers certain advantages. First, since it is implemented on top of the Java Virtual Machine (JVM), JRuby threads are mapped to the kernel threads, and Unicode strings are automatically supported in JRuby. Second, JRuby can seamlessly interoperate with Java. Meaning, that we can use Java objects as normal Ruby objects, and vice versa. Hence, we can exploit the wealth of Java libraries, using the power of Ruby's flexible syntax. To demonstrate the interoperability between JRuby and Java, we use the Java listing shown in Figure 2, in which we define the `Foo` Java class.

We compile the class and put it in the `example.jar` library. Now, we can use the `Foo` class in JRuby. To