Ruby on Rails: The Whats, Whys, and Hows

Joe Francis, Brooke Patterson

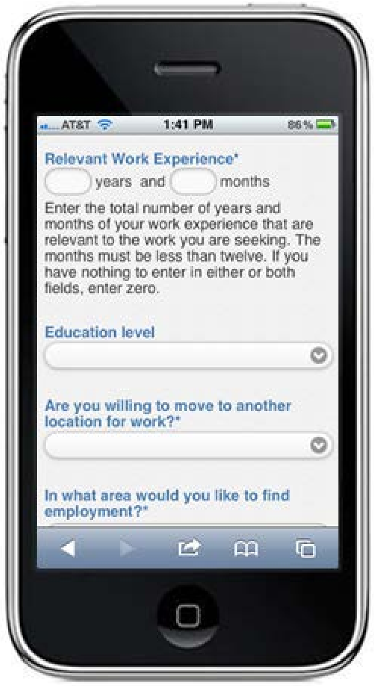
February 2014

In 2012, America's Job Link Alliance–Technical Support (AJLA–TS) partnered with the Delaware Department of Labor to design a new resume builder for our America's JobLink (AJL) application. The fast timeline combined with complex business and technical requirements meant a "business as usual" approach would not be successful. Thinking outside the box led us to Ruby on Rails (Rails), an open source web development framework written in the Ruby programming language. If "Ruby on Rails" sounds like a cross between a tennis bracelet and a theme park ride, read on. This white paper will answer the whats, whys, and hows of Rails and explain its role in both the resume builder project and comprehensive revision of AJL.

Disclaimer: This White Paper is issued for information only. It does not constitute an official position of America's Job Link Alliance–Technical Support (AJLA–TS), AJLA, or its members. While the authors have used their best efforts in preparing this White Paper, neither the authors nor AJLA–TS guarantee the accuracy, relevance, or completeness of the information or any linked information presented, referenced, or implied. All critical information should be independently verified. AJLA–TS declines all responsibility for any errors and any loss or damage resulting from use of the contents of this White Paper. AJLA–TS also declines responsibility for any infringement of any third party's Intellectual Property Rights (IPR), but will be pleased to acknowledge any IPR and correct any infringement of which it is advised.

BACKGROUND

AJLA–TS launched its new resume builder in the state of Delaware on March 18th, 2013, only nine months after the project start date. Leadership from the Delaware Department of Labor reports highly favorable feedback from the field. Said one job seeker, “I purchased a similar product for my home computer. It is not even as good as what you offer and I had to pay money for it.” The resume builder’s success in Delaware proved the viability and desirability of using Rails for future AJL projects. But let’s backtrack a bit to examine what prompted our search for a different development framework in the first place.



Relevant Work Experience*

years and months

Enter the total number of years and months of your work experience that are relevant to the work you are seeking. The months must be less than twelve. If you have nothing to enter in either or both fields, enter zero.

Education level

Are you willing to move to another location for work?*

In what area would you like to find employment?*

The requirements for the new resume builder reflected recent shifts in job search and recruitment practices. Today’s job seekers and employers expect mobile accessibility, email alternatives like instant messaging and text messaging, and social media integration. With ongoing staff reductions in American job centers, the new resume builder had to minimize staff intervention through intuitive, user-friendly design. It also needed to accommodate ever-shortening attention spans by dramatically decreasing the time required to produce a quality resume—all this and more, while still complying with federal reporting requirements.

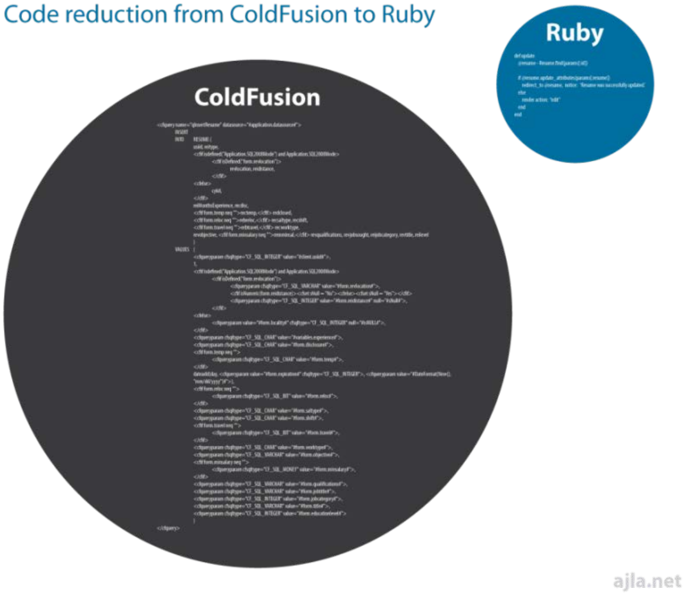
Not only did Delaware demand a cutting-edge product, they wanted it fast. We soon realized that meeting these expectations would be nearly impossible within AJL’s existing Microsoft Windows and ColdFusion framework. While AJL’s original architecture has served our customers well for nearly 15 years, it’s showing its age in many ways. To keep up with increasingly rapid developments in technology we needed to explore alternatives.

With no time for a learning curve, we began with a skills inventory of our programming staff. We rejected platforms that were incompatible with our current system, and gave preference to those known for rapid development. Lastly, we asked ourselves, “Which solution makes the best business sense in the long run?” We wanted a sustainable path to the future, not a band-aid fix. The answer was clear: develop the project using Ruby on Rails deployed to Linux servers.

THE WHATS

Ruby is an open-source programming language created in the mid-1990s by the Japanese computer scientist, Yukihiro “Matz” Matsumoto. Ruby is often described as an “elegant” language, because of its concise, logical syntax. It requires approximately 50% less code to accomplish the same task in Ruby as it does in Java or C++. Ruby’s efficiency is even more apparent in a ColdFusion to Ruby comparison as seen in the graphic below.

Code reduction from ColdFusion to Ruby

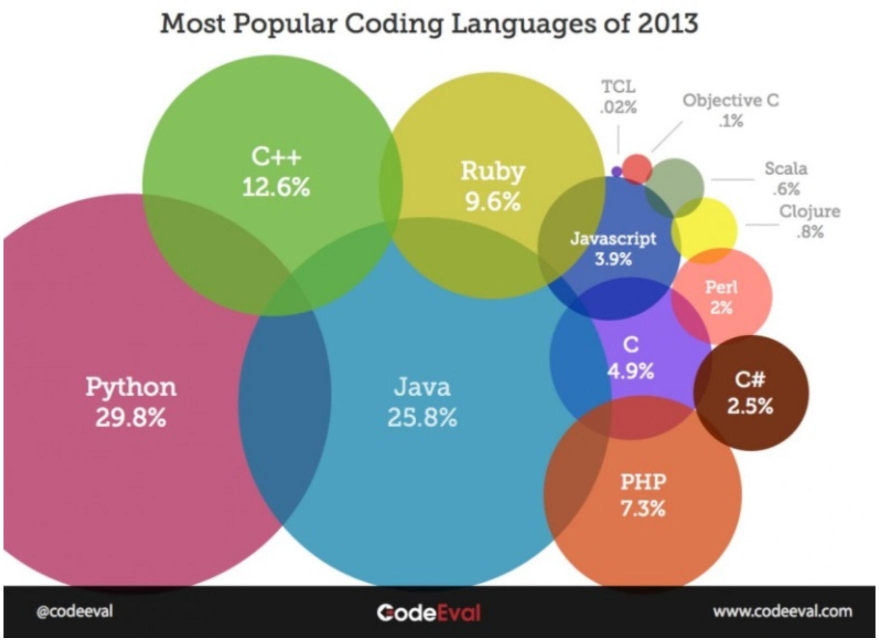


```
<cfquery name="qInsertResume" datasource="#application.datasource#">
	INSERT
	INTO	RESUME (
		uaid, rettype,
		<cfif isdefined("Application.SQL2008Mode") and Application.SQL2008Mode>
			<cfif isDefined("form.relocation")>
				relocation, reldistance,
			</cfif>
		<cfelse>
			cyid,
		</cfif>
		militarytotalexperience, recdisc,
		<cfif form.temp neq "">rectemp,</cfif> reddeleted,
		<cfif form.reloc neq "">relreloc,</cfif> recsaltype, recshift,
		<cfif form.travel neq "">rebtravel,</cfif> recworktype,
		revobjective, <cfif form.minsalary neq "">remminsal,</cfif> revqualifications, revjobsought, reijobcategory, revtitle, releveel
		)
	VALUES (
		<cfqueryparam cfsqltype="CF_SQL_INTEGER" value="#client.uaid#">,
		1,
		<cfif isdefined("Application.SQL2008Mode") and Application.SQL2008Mode>
			<cfif isDefined("form.relocation")>
				<cfqueryparam cfsqltype="CF_SQL_VARCHAR" value="#form.relocation#">,
				<cfif isNumeric(form.reldistance)><cfset isNull = "No"><cfelse><cfset isNull = "Yes"></cfif>
				<cfqueryparam cfsqltype="CF_SQL_INTEGER" value="#form.reldistance#" null="#isNull#">,
			</cfif>
		<cfelse>
			<cfqueryparam value="#form.locality#" cfsqltype="CF_SQL_INTEGER" null="#isNULL#">,
		</cfif>
		<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#variables.experience#">,
		<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#form.disclosure#">,
		<cfif form.temp neq "">
			<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#form.temp#">,
		</cfif>
		dateadd(day, <cfqueryparam value="#form.expiration#" cfsqltype="CF_SQL_INTEGER">, <cfqueryparam value="#DateFormat(Now(), "mm/dd/yyyy")#">),
		<cfif form.reloc neq "">
			<cfqueryparam cfsqltype="CF_SQL_BIT" value="#form.reloc#">,
		</cfif>
		<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#form.saltype#">,
		<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#form.shift#">,
		<cfif form.travel neq "">
			<cfqueryparam cfsqltype="CF_SQL_BIT" value="#form.travel#">,
		</cfif>
		<cfqueryparam cfsqltype="CF_SQL_CHAR" value="#form.worktype#">,
		<cfqueryparam cfsqltype="CF_SQL_VARCHAR" value="#form.objective#">,
		<cfif form.minsalary neq "">
			<cfqueryparam cfsqltype="CF_SQL_MONEY" value="#form.minsalary#">,
		</cfif>
		<cfqueryparam cfsqltype="CF_SQL_VARCHAR" value="#form.qualifications#">,
		<cfqueryparam cfsqltype="CF_SQL_VARCHAR" value="#form.jobtitle#">,
		<cfqueryparam cfsqltype="CF_SQL_INTEGER" value="#form.jobcategory#">,
		<cfqueryparam cfsqltype="CF_SQL_VARCHAR" value="#form.title#">,
		<cfqueryparam cfsqltype="CF_SQL_INTEGER" value="#form.educationlevel#">
		)
</cfquery>
```

```
def update
  @resume = Resume.find(params[:id])

  if @resume.update_attributes(params[:resume])
    redirect_to @resume, notice: 'Resume was successfully updated.'
  else
    render action: "edit"
  end
end
```

Ruby's ease-of-use and active global user community have consistently kept it among the top 10 most popular programming languages in the world according to most industry measures. In 2013, it ranked as the fourth most popular coding language according to CodeEval. These results were drawn from a sample size of over 100,000 development challenges on CodeEval's programming competition platform.



Ruby on Rails, often shortened to "Rails," is a "full-stack" (end-to-end) framework for developing web applications. It was built using the Ruby programming language by Danish programmer and Ruby enthusiast, David Heinemeier Hansson, and released in 2004. Rails consists of a large, open source library of software administration tools (RubyGems), and an application programming interface (API) that defines how software components should interact (1).

RubyGems allows programmers to freely share Ruby software packages (gems) that extend or modify functionality within Ruby applications. Free access to gems means programmers don't have to constantly reinvent the wheel. The Rails API documents conventions that guide and simplify Ruby development. Although there may be more than one way to accomplish a task, both the Rails API and community encourage the use of a standard or "Rails way."

"The Rails way" is to host Rails applications on Linux. Linux is a free operating system used as an alternative to Windows. Linux runs everywhere—from your home wireless router, to your Android phone, to Amazon's web server farm, to the largest supercomputers in government and industry, even the White House (2). Linux comes in various packages or "distributions" designed for different uses. AJLA–TS uses the Ubuntu Server distribution. NASA also uses Ubuntu to collect data from spacecraft, satellites, and other exploratory ventures (3).

Rails, hosted on Linux, is popular with start-ups and large-scale companies alike. Countless small applications run on Rails and Linux, as do many well-known websites, such as Chowhound, Get Satisfaction, GitHub, Groupon, Hulu, SlideShare, SpiceWorks, Twitter, and Yellow Pages. AJLA–TS has used a Rails product, Redmine, as our issue tracking system for many years. We also host our source code repositories in Subversion for Rails on a Linux server.

THE WHYS

Now that you have a basic understanding of Ruby, Rails, and Linux, we can tackle the question, "Why Rails?" A comprehensive analysis of Rails' strengths is beyond the scope of this paper, so we'll focus on those strengths most pertinent to the resume builder project and AJL development in general.

The number one benefit most people associate with open source solutions is cost savings. Our entire suite of development and production tools for the resume builder project was free. While "freeware" is a bit of a misnomer (businesses must still invest in hardware, staff, support, etc.), open source solutions do typically offer a lower total cost of ownership than proprietary solutions (4). Open source products carry lower administration, hardware, upgrade, and of course, licensing costs. Ruby code in particular is simple, well-structured, and thus easy to maintain. This helps make Rails not just inexpensive in the short-term, but highly cost-effective in the long-term. Simply avoiding the procurement, legal, and licensing delays associated with proprietary software provides notable cost savings. After all, time is money.

This brings us to perhaps the most talked about (or more accurately, *blogged* about) Rails advantage: time to market. As noted by RedKivi Consulting, "The biggest challenge in the Software Industry is to 'Ship Usable Code On Time.' Most projects are delayed causing intangible increase in costs in terms of adverse business impact and missed opportunities" (6). Rails, known for high-velocity development, is the natural choice for projects with short timelines. And what project doesn't have a short timeline these days?

> "Many high-profile consumer Web firms are choosing Ruby on Rails to rapidly build scalable web applications. Ruby on Rails has the potential to emerge as a strong alternative platform to traditional choices based on Java and .NET for next-generation enterprise applications, as companies seek improved agility, development speed, and time to market" (5).
>
> *Eric Knipp, Managing Vice President at Gartner*

Some of Rails' speed comes from the streamlined nature of coding in Ruby. One of Ruby's fundamental precepts (supported by vast, open source template libraries) is DRY, "Don't Repeat Yourself." A wide variety of free Rails automation tools, like Vagrant and Chef, provide additional time savings. Automated management of development, test, and production environments leads to increased quality and faster delivery. Furthermore, by automating menial, repetitive tasks, Rails frees up human resources for more creative, collaborative activities. This makes Rails ideal for developing custom web applications that stand out from the competition—a critical consideration as Delaware and all of our customers strive to be the top recruitment and job matching source in their states.

Speed is part and parcel of another Rails trait pertinent to our discussion: agility. Unlike many widely-used proprietary frameworks, Rails is tremendously nimble. To stay competitive, we must be quick on our feet as we respond to rapid changes in business and technology. Rails also supports agile development practices like continuous customer feedback through quick delivery of prototypes. This is particularly helpful in projects with a lengthy "discovery phase," where requirements are determined and may evolve during the course of development. The resume builder project fit this description, as do many AJL development and implementation projects.

As previously mentioned, we selected Rails with the future in mind, not for the resume builder project alone. Lower cost of ownership, development speed, agility—all of these advantages mean little without stability. Predicting the longevity of software (or any product for that matter) is far from an exact science. Still, several signs point to the staying power of Rails. Open source solutions are gaining popularity worldwide, even in the government sector (7). The Rails user community is thriving and constantly contributing new features and functionality. The latest version, Rails 4.0.2, was released in December 2013, which makes Rails more state-of-the-art than Microsoft's ASP.NET. So, how long will Rails be around? In the absence of a crystal ball, we turned to our knowledge of IT history. Software that is simple to use, easy to scale, and capable of delivering a quality product tends to stick around. Rails fits that bill.

THE HOWS

Armed with new tools and a new development stack, our team tackled the resume builder. Of course, the rest of the AJL application was (and still is) written in ColdFusion and hosted on Windows. Mixing an application between two development platforms poses major challenges. We took it as a non-negotiable goal, however, to make the integration as simple and transparent to the user as possible. The look and feel of the new resume builder does differ from the legacy application, but it appears as a different section of the site rather than a separate application.

Out of practical necessity, both the Rails and ColdFusion sides of the application work with the same SQL Server database. The entire site runs under one domain with one Secure Sockets Layer (SSL) certificate, and all logins, authentication, and timeout logic are handled in one place. A solution involving Rack middleware in the Rails side and simple REST web services in the ColdFusion side provides seamless navigation between both areas of the application.

> “After researching the market, Ruby on Rails stood out as the best choice. We have been very happy with that decision. We will continue building on Rails and consider it a key business advantage” (8).
>
> *Evan Williams, Creator of Blogger, ODEO, and Twitter*

Overall, the resume builder was constructed to work within the confines of the legacy system without making any major compromises in the new code base. While a layer or “shim” of compatibility code was required, it’s invisible to the user and was designed with an eye towards its eventual removal. Integrating the two systems in this way is a solid accomplishment of which AJLA–TS is quite proud.

By all accounts, the resume builder project was one of the largest victories in recent memory. First and foremost, it was delivered on time without the need of additional programming staff. The entire application is mobile-capable, and the look and feel (including instructional text and labeling) is easily customizable to fit the unique requirements of each of our customers. In fact, this same support can be used to provide native foreign language support if needed.

AJLA–TS plans to build on the success of the resume builder project by continuing to port additional sections of the AJL application to Rails. A piecewise porting approach is preferable in many ways to a wholesale port. A wholesale port simply takes too long, while state and federal mandates ensure you are trying to hit a moving target. Breaking the project into bite-sized chunks is much more tenable, as demonstrated with the resume builder.

We are currently working on the remaining self-service portions of the site. These portions are most closely related to the resume builder and will benefit the most from mobile capabilities and a new unified communications and notification layer that will allow the user to select whether they are best contacted via email, text message, or even via Facebook or other social networking sites. Once the self-service side of AJL is done, we will move on to other sections of the application.

SUMMARY

The decision to revise the AJL application in Rails was not taken lightly, nor was it made out of personal preference. Rather, after carefully analyzing the constraints and demands of the resume builder project (and projects to come), Rails stood out as our best chance at success.

None of this is to say that Rails is the only way to develop software successfully. Many shops employ various languages, frameworks, and techniques to great effect. A key part of our analysis was to make sure we best utilized the skills and talents we already have at AJLA–TS. Other solutions may be superior for other teams, but we needed a solution for our team. These days we're all asked to do more with less. Using Rails is our way to do more.

We believe our success with the resume builder proves that we have developed a workable method to port the legacy ColdFusion code into a modern web application framework. As we port more and more of the application, the benefits to both functionality and development velocity will snowball, allowing AJLA–TS to be more responsive to our customers' needs in a marketplace that's changing faster than ever.

REFERENCES

1. Daniel Kehoe, "What is Ruby on Rails?" October 11, 2013, http://railsapps.github.io/what-is-ruby-rails.html.

2. Tim O'Reilly, "Thoughts on the Whitehouse.gov Switch to Drupal," *O'Reilly Radar*, October 25, 2009, http://radar.oreilly.com/2009/10/whitehouse-switch-drupal-opensource.html.

3. Sean Kerner, "Why NASA Uses Open Source," *InternetNews.com*, July 20, 2010, http://www.internetnews.com/skerner/2010/07/why-nasa-uses-open-source-and.html.

4. David A. Wheeler, "Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!" April 16, 2007, http://www.dwheeler.com/oss_fs_why.html.

5. Tom Mornini, "Here's Why Ruby on Rails Is Hot," *Business Insider*, May 11, 2011, http://www.businessinsider.com/heres-why-ruby-on-rails-is-hot-2011-5.

6. Redkivi Consulting, Ruby on Rails Is for You [Infographic], http://www.redkivi.com/blogs/ruby-rails-you (accessed January 7, 2014).

7. Miranda Newbauer, "Open-Source Benefits to Govt Outweigh Misconceptions, Report Says," *TechPresident*, November 27, 2013, http://techpresident.com/news/24570/open-source-benefits-outweigh-misconceptions-report-says.

8. Ruby on Rails official website, "Quotes," http://rubyonrails.org/quotes (accessed January 7, 2014).

ABOUT AJLA–TS

America's Job Link Alliance–Technical Support (AJLA–TS) is the technical support arm of America's Job Link Alliance (AJLA), an association of state workforce agencies dedicated to providing powerful, affordable workforce development systems. Headquartered in Topeka, KS, AJLA–TS provides systems development, maintenance, and enhancement; hosting; product and customer support; and user and technical training for local and state workforce development staff. More information is available at www.ajla.net.

ABOUT THE AUTHORS

Joe Francis is Application Development Manager at AJLA–TS. He has a BS in computer science and 15 years' experience as a software developer and systems administrator.

Brooke Patterson is an Instructional Designer at AJLA–TS. She has a BA in English and seven years' experience in technical communications in the publishing and software industries.

AJLA
America's Job Link Alliance

America's Job Link Alliance–Technical Support
1430 SW Topeka Boulevard
Topeka, KS 66612
Phone: (800) 255-2458
ajladesk@ajla.net
www.ajla.net