CODE
RAGE

Code Rage 2.0

Rants of Code and Other Assays

Get That Raise

I was recently asked for a few tips for negotiate a pay raise. There is not much to asking for a raise, the key is to ask at the right time. Ask for a raise when the company is on a high note, an up swing such as having recently closing a big deal or accomplished a critical milestone. Before asking for a raise you should get a good sense of the current job market and the value you add to the company. If you want a 10% raise ask for 15%, if you want 15% ask for 25%. In addition to just a 25% raise, I would ask for a new title, more training, higher team leadership and management role. When asking for a raise you will have a better chance if you also step up and ask for more responsibility, not just money but a bigger and more visible role in the team. Just to make sure you make every point in asking for a raise, print out a one to two page document which highlights your accomplishments for the past year, take some initiate and make some objects for yourself for the coming year, and also detail your request for a pay raise, new title, additional responsibilities, and other fringe benefits.

Some manager have canned answers for when employees ask for a raise, such as the pay structure is pretty flat in the company, or that you don't have a required skill or degree required for a given salary range. No matter what, do not walk away with nothing, negotiate either for concessions of other benefits, perhaps educational reimbursements or corporate training.

So go ahead, get that raise, you deserve it.

The $100,000 Customer

In today's age, customer service has become a oxymoron, or a cliché at best. I recently had a bad customer experience at the Fry's Electronics that made me think about just how right the customer actually is. On recent trip to Fry's they employed a bait and switch on a in store sale. This had been the second consecutive time this had happened and I asked the casher to see the manager. After going from the Department Manager to store Manager to Customer Service Manager I finally had it. As a customer I thought I was right, they incorrectly and misleadingly promoted a big sale and placed fliers on half an aisle and they were not owing up to their mistake. I demanded to file a complaint, but only after I told him that this was the second time I fell victim to this sort of bait and switch tactic and that I was personally unsatisfied with the service I was getting did he ask me to put my name and number in a little, white, and meaningless piece of paper, which he undoubtedly threw away after I left the store. But before I left the store, and the additional $200 worth of merchandise I was going to buy, I informed him, and half the store that they had just lost a $100,000 customer.

You might be wondering how I figure I am a $100,000 customer if at I had a little over $200 worth of merchandise. We'll, I've been a customer of Fry's for over ten years. I bought my first desktop there, paid over $2,400 in cash for it too. Since then I have bought other desktops, laptops, video, SLR, and point and click cameras, books, movies, and a ton of other electronic gadgets and accessories. If you add it all together, I've been a loyal customer until now.

In any given year, you may spend up to $5,000 dollars in your local grocery store, 20 years of that makes you a $100,000 customer. Take Amazon for example, how long would it take you to be a $100,000 Amazon customer? Not long, if you are a happy customer. Take apple for example, if you are like me you are reading this on your third or forth mac computer and you own and have given an iPod for just about every gift occasion. A satisfied customer has no qualms about spending on reliable service and quality goods, even at a premium.

The retail business is a tough business. Most retailers have a razor thin profit margin of 1-3 percent, and they just can't afford to mistreat their customer in a rude, unprofessional, deceiving, unethical fashion. When Amazon often has better deals on just about everything, with free premium shipping, how is Fry's going to differentiate and win back customers like myself?

In the New York Times best seller The Last Lecture, Randy Pausch tells the story of the $100,000 salt and pepper shaker. The story is about how when, as a child, the authors parents took the whole family to Disney World in Orlando for the first time the author and his kid sister bought a souvenir salt and pepper shaker for their parents. In the excitement, the salt and pepper shaker fell and broke minutes after purchasing it. As any kid would, they were upset to the point of attracting the attention of a fellow guest who suggested for them to ask the casher for it to be replaced. They did and to their surprised the got a new shaker set and apologies for not wrapping it correctly. The salt and pepper shaker that they had bought was not worth $100,000, it was worth no more than $10 dollars. But on learning about this, and having the family vacation of their lives, the authors family gave Disney World over $100,000 in business over the years. The authors father in his capacity as a volunteer in English as Second Language organized student trips to Disney World well worth $100,000.

You are a $100,000 customer, so demand $100,000 customer service. Take back and take control your purchasing power.

In a Startup

After working for upstart startup, I thought of a few axioms of working for a small and agile team in a fast and merciless marketplace.

- In a startup, you can have any title you want, say VP of Version Control, but no one reports to you other than yourself.
- In a startup, if you code it, break it, test it, or fix it then you own it.
- In a startup, if one guy call in sick a third of your development team is out.
- In a startup, if you been there longer than 3 years you are most likely been there longer than your CEO.
- One year in a startup equals 2.7 in a large corporations.
- Would you rather start a startup or upstart a business?

How To Kill An Open Source Community

If you don't understand Open Source licensing, don't start an Open Source project. Keep your code! ExtJS, a JavaScript framework for building business forms, recently made big news, the bad kind, when it changed it's license from LGPL to GPL. ExtJS started as an extension to the Yahoo! UI library.

ExtJS had been in my radar for a long time, but I never downloaded it, used it, wrote about it, or contributed to the community in any way because since its foundation the licensing of the library seemed awkward to me. If my memory serves me right, earlier releases of ExtJS had interesting clauses that prohibited you bundling ExtJS in frameworks. For me, I keep on using jQuery and YUI.

Open Source is not so much about the code, it is about the community and how that community interacts with other communities. Open Source is community building. In this Age of Meetoo, companies are sprung with VC money simply by cloning services and products of other companies. Look at all the 'social viral video sharing' sites are just imitations of YouTube. In this age, the real value of code does not lie in the source code, the value lies in the knowledge and expertise of the community. The same can be said of a service, the value lies in the user base.

The folks behind ExtJS feel that this license change to GPL adheres to the quid pro quo principle. This is true if all you want is code, but community evangelism is worth is worth more than its weight in code. Look at the spectacular growth and good will around jQuery. For every one line of code in jQuery, there is at least one plugin written by a third party. For every one line of code in Ruby on Rails, there is at least one coder-blogger-evangelist promoting the framework.

It is true that you can shoot yourself in the foot with just about any programming language, but with changing the license of an Open Source project you can shoot your whole community, execution style.

Graeme Roche, project leader of Grails, said, "What they have effectively done is built up a community, taking full advantage of the open source model by accepting user contributions and patches and then turned around and kicked their own community up the backside."

Jack Slocum, the lead developer and founder of ExtJS, responded to all the criticism on his blog. Jack complains, "Shortly before 1.0 is released, there numerous Ext “clones” started popping up that were hacking Ext themes." Other developer hacking, learning, promoting, evangelizing, and cloning is the great benefit of releasing an Open Source application, ExtJS itself was a 'clone' and a hack of Yahoo! UI.

What I find interesting of the whole event is that this is history repeating itself. This is not the first time nor will it be the last time that some organization has leverage a license for some perceived monetary benefit.

Software Piracy Express Edition

On a recent Java Posse episode, while talking about a price increase for IntelleJ IDEA, Joe Nuxoll of the Java Posse went on a bit of a rant about software piracy. Joe said, "Being a software engineer that has made my living selling software I don't steal it, don't copy it, I don't even do that for music or movies or anything. I totally pay for everything, happily." On the Posse forums, a fellow programmer said that for programmers to steal software is essentially like stealing from themselves.

Even though we could all agree with the sentiment, the absolutism and preachy righteousness tone of Joe's comments made me think about his premise, that as software developers, we should never download software we are not 'entitled' too.

Let me first be clear, I don't pirate. I pay for software but not because I write code. I pay for the convenience, quality, and utility that the software provides me, but perhaps most importantly because I can afford it and use it professionally. I don't pay because I feel forced to pay because I also write software applications. I don't feel that if my software is pirated I won't get paid. As a developer, it is not like we get royalties on our work, so what do I care that 5% of my software is pirated, those that do pay for it make up the perceived loss.

Like all of you, I know a lot of fellow programmers that do torrent software, but the funny thing is that they really don't use the software professional. In a way, the programmers that I know to be torrent, download, and crack software do so to try it out for longer than the trail versions allow. These software pirates fall into one of the following categories, they are just software pack rats that download and install every new shiny piece of software and use only once in a blue moon, or they are students of sorts and are trying to learn or experiment with new technology.

To good thing is that for whatever software you are interested their is a free open source equivalent. The open source part of the equation usually doesn't much matter for end users, but the free part does. Software companies are beginning to understand that this perceived loss in revenue to piracy is negligible, that is why you are seeing a lot of free 'express' versions of software packages like Visual Studio, Oracle. Software companies, instead of taking RIAA-like stance of suing college students and the occasional user, are providing the people a legit option to using a free, express, limited version of the same software package and offer professional editions for those that need and use it on a daily basis. Companies like Oracle know that the greatest loss does not come from pirated versions of their software but from the loss market share to Open Source solutions.

Software companies should listen to their customers, even those that use their software a few times, instead of their lawyers, that most likely never have used their software.

Debugging Users and Invisible Characters

As a developer working for a startup with live software in the field, I often have to problem solve some interesting and out of the box bugs. One recent bug was related to changes to the Daylight Standard Time and Sun's Java Timezone Updater, which is far from the control of my code. The Timezone Updater is a simple Java application that can be run as follows.

```
"c:\Program Files\Java\jre1.5.0_10\bin\java" -jar tzupdater.jar -f –bc
```

After documenting, testing, and validating the process of running the Timezone Updater from the command prompt we had our sales engineer upgrade the all of our clients' JRE with the current timezone data. After some time we had reports that some clients had a problem running the above command, they where getting the following Java error.

```
Exception in thread "main" java.lang.NoClassDefFoundError: ûjar
```

It seemed that Java could not find a class named ujar (notice that funny hat over the u). Our sales engineer, with the aid of a software engineer, spent all day going back and forth with the client. As you might have guessed, updating the time zone data worked for our engineers but at the client site they consistently got the above error. After a long email thread full of instructions, screen shots, and frustration I was pull in.

I too was unable to reproduce the error until I considered the error. The only clue in the above error is the funny character u. After some time it occurred to me that the -jar had been corrupted into ujar. But what would distort the dash? Word. The instructions to run the Java Timezone Updater where written and sent in Outlook and cut and pasted from Outlook. Outlook uses Word as the default mail editor and Word has the awful tendency of distorting plain old ASCII text into special characters at will, try typing :) in Word.

This bug was not obvious to our client's IT personnel, our front line sales engineer, and an experienced software engineer. Having code working on the developer's machine is not a valid solution; software needs to work at the client's site too!

One lesson that most developers don't learn is to debug outside the debugger. As an engineer there are times you need to trouble shoot, problem solve, and debug not just your software from the comforts of your favorite IDE but the whole software stack, network, hardware, user's environment, and even the user himself.

Every problem, issue, and bug experienced by the end user directly and indirectly with your software eventually needs to be implicitly and explicitly dealt with by your software development team. Bugs that are on the fringes of the code base are the most difficult to solve, that is why each member a software development team needs to take full ownership of the whole code base, and every known issue.

Developing Your Programmer’s Intuition

The other day I solved a bug. I didn’t think much of it. The bug basically manifested itself when the user search for data but none was returned. We write an ‘enterprise-grade’ software application, so there is a lot of code and tiers between clicking the search button and displaying the data in the client. But in thinking about this bug I had pin pointed only a few methods in a few classes. The problem ended up being that in the database there was a value that was padded with an extra white space and one of the SQL queries used returned an empty set. I didn’t think much about the fix to the bug, but when I described the problem and solution to my coworker he said, ‘How did you know there was an extra white space? I would never have thought of that, you can’t tell if there is a white space by just doing a select.”

I think that as software programmers, we do develop of certain intuition that allows us to better understand software. This code sixth sense aids us in debugging, problem solving, and trouble shooting your own code, as well as third party applications. After seeing soo many null pointer and cast class exceptions, you develop a noise for sniffing out smelly code.

Here are a short list of root causes for stupid bugs I have encountered. I have seen bugs in hash maps because the key is case sensitive. I have seen a lot of problem when trying to parse a file extension type and someone uses the String’s indexOf (use lastIndexOf instead). You are comparing the equality of a string against a string literal, code defensively to avoid null pointer exceptions (”LITERAL”.equals(stringVariable)). Encapsulation means nothing if you provide a public getter and setter for every field for every class. Code against interfaces. Objects that are globally cached should be immutable. Runaway memory leaks are caused by maps that cache large objects that are not removed. In batch files, you need to quote paths if they have spaces.

If you are a novice programmer you can read into smelly code and try to write effective code to develop your programmer’s intuition. Programmer’s intuition is developed through experience.

CommunityOne 2007: Lunch with the Java Posse

The JavaPosse podcast is hosted by Tor Norbye (Sun), Carl Quinn (Google), Joe Nuxoll (Apple), and Dick Wall (Google). The regular format for the show is for the posse to run down trough and pontificate the current news in the Java world. For this live recording of the Java Posse podcast the posse adopted the Top Ten count down made famous by David Letterman. Some of the posse’s top predictions for JavaOne 2007 included mobile mayhem, desktop Java push, and that applets will no longer suck.

According to the posse, the top missing features in Java include the missing LINQ, cross platform solutions for recording and playing back audio, device support for USB and FireWire, browser level Swing HTML rendering, language level properties and events, component model for the desktop, component model for the web, component model for the mobile, and more importantly a consistent component model for all. Additional missing features in the language include true generics, regexp literals ala Groovy, multiple return values or parallel assignment, closures, @Nullable annotation, type inference ala Scala, invoke dynamic, and modules and super packages.

The posse then listed the top APIs to nuke from Java which include anything related to date and calendar, java.awt.List, java.awt.*, java.beans.*, java.swing.LookAndFeel, java.util.Stack, java.util.hashtable, the close method if it throws an exception, java.io.File since it is just a path, cloneable, binary serialization, Enumeration, and CORBA.

The top software industry pet peeves according to the Posse include the not invented here syndrome, paper architects (if you can’t code, retire), developers with nothing left to learn, IDE zealots, framework zealots, snap judgment of technology, frivolous patentry, lack of User Interface sex appeal, coding styles, meetings, tHE cAPS lOCK mUST dIE, unskippable intros, and general lack of manners.

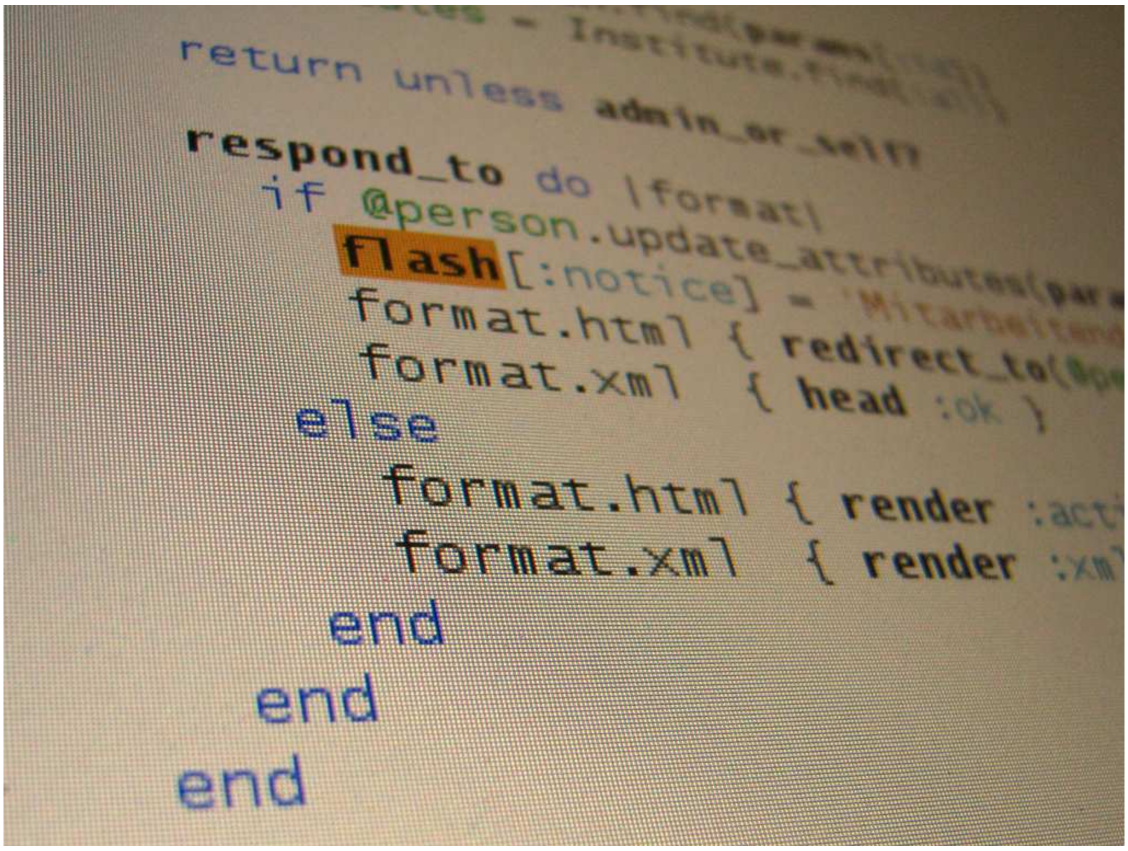
Here are some memorable quotes from the posse members.
The definition of Hell is working with dates in Java, JDBC, and Oracle. Every single one of them screw it up. — Dick Wall
This project is running late, quick lets have a meeting. — Dick Wall
The recommendation I would give you is not to use Swing, but to be using a rich application platform like NetBeans. — Tor Norbye

You Might Be in a Broken Project If

I think that some of the best Java Posse rants happens when they do a live show. The following is my transcription from their live show at JavaZone 2007.

Top reasons indicating you might be in a broken project…

- It adds three months to add a checkbox on a web UI.
- Everything starts to look like it would be quicker to rewrite.
- Everyone on the project has architect on their business card.
- Conversations always start with ‘oh that, that is really simple…’ followed by a thirty minute discussion of what is required.
- You time your life around The Build.
- The GUI is written in AWT or HTML 3.2.
- The GUI is written in something you wrote yourself.
- The Project X is a homegrown web framework.
- Someone just added five more hours of meetings to your day because the release is late.
- The guy that is supposed to train you, throws a file full of notes, and runs away.
- The lines of XML outnumber the lines of Java 10 to 1.
- You ask about JUnit tests and you get blank stares from everyone.



```
[illegible] = Institute.find(:all)
return unless admin_or_self?
respond_to do |format|
  if @person.update_attributes(para
    flash[:notice] = 'Mitarbeitend
    format.html { redirect_to(@pe
    format.xml  { head :ok }
  else
    format.html { render :act
    format.xml  { render :xm
  end
end
end
```

Java, Ruby, and even Python Sucks

There is a bit of a flame war unfolding between Javanistas and Rubyists. I could trace this most recent scuffle to an article posted on JavaLobby by Daniel Spiewak about a little Java library called ActiveObjects. The aritcle is promoting this Java-based but Rails inspired Object Relational Mapping library. ActiveObjects is a Java implementation of the Active Record pattern made famous by Ruby on Rails and it's use of convention over configuration. In touting the benefits of ActiveObjects, Daniel complains and grossly exaggerates that in Hibernate "you have to write more XML than code!"

Gavin King, the founder of Hibernate, responded that XML was soo 1999 and now they overuse annotations. Gavin wrote, "Hibernate Annotations has been around since early 2005 and there is no longer any good reason for people to define mappings in XML."

Out of the blue and into the blogosphere, Obie Fernandez responded to Gavin's 'FUDdly' remarks with his top 10 rant why Java sucks ass! Basically Obie resorts to fighting FUD with FUD. Obie goes ballistic on compilers, IDEs, frameworks, libraries, High School Musical 2, and Java developers themselves. Obie rants that most Java Programmers are morons. From his writing, it is clear that Obie idol worships DHH as the fucking second coming of the fifth generation computer language era. Obie's top ten reasons why Java sucks include that the language makes money for vendors. From his Ruby rage you think he hopes to make Java vendor money by writing Ruby books.

To add FUD to the flames, Daniel, which started the whole my milkshare (ORM) is better than yours drama, responded with his top ten reasons why Ruby sucks ass infinity times ten. Daniel writes, “Some of Obie’s points are either misinformed, or deliberately misleading.” According to Daniel, Ruby sucks goatse ass because more of most Ruby programmers are morons, Ruby has an annoying community, and because Ruby has DHH. Touché.

Now, I may be a bit late to the party but I also have my favorite reasons why Java sucks. I also think Ruby sucks too. And just for good measure, Python sucks the most!

Ruby Sucks Because

- Ruby does not any major corporate sponsorship or backing.
- At RubyConf everybody wants to talk about Rails.
- At RailsConf half the audience is writing books from lectures given by the other half.
- Ruby is a DSL for Rails.
- Most people think that DHH wrote Ruby or never have heard of Matz.
- There is no language specification for Ruby.
- Ruby will fragment to different wannabe successor implementations.
- JRuby will ultimately go where Ruby can’t, the enterprise.
- Closures are by no means unique to Ruby, get over it.
- Ruby and Rubyist seem to have an inferiority complex when it comes to Java, get counseling.
- Rails is too tightly mind controlled, managed, copyrighted, owned by DHH and 37signals.

Java Sucks Because

- Java has way to many freaking frameworks.
- There is sooo much to learn in developing an application.
- Java has far too many APIs and libraries that are designed by committee.
- People still think Java is slow.
- Java still suffers from the fiasco that was Applets.
- Java still suffers from the fiasco that was EJB 2.x
- There too many flavors of the JRE (Java 5, Java 1.4, J2SE, JEE, J2ME, JWTF)
- The JRE has a lot of junk in the trunk.
- Methods gets deprecated, nothing gets removed, the API gets bloated.
- Java does not always run everywhere, sometimes you need to go native.
- Java is still primitive.
- Java developers are a dime a dozen.

Python Sucks Because

- Python cares about whitespace, I don’t care for that.

- Python programmers are morons.
- I like David Chappelle better than Monty Python.
- Guido van Rossum doesn’t work at 37signals.
- Python has no personality like JavaScript or personalities like Dave Thomas.
- Python will never go mainstream, not even with Google’s backing.

To be honest, the back and forth almost sound like ‘yo mama’ comebacks. I mean, this is all to reminiscent of the east coast/west coast baby mama drama between Biggie Fries and Six Pack Shakur. There’s even some would be wanksta crew hating on CGI, Pascal, Ada, and Basic.

Here are my Jerry Springer final thoughts, if programming languages evolve beyond closures and white space to natural languages, are we still going to have these language wars? If we where to program in a natural language, say in English or German or Hindi, we would be as writers. Our JEE applications would be as thick as Moby Dick and our Ruby on Rails CRUD applications thin as Dear John letter. We as programmers tend to think we are artists, do writers and artist act as such? Oh Hell Ya!!

Is Software Development Dead?

It seems to me that there is a common theme, or meme, propagating in the background of developers’ mind as I keep reading how such and such technology is dead. Take for example the latest incarnation of this question asked by Zarar Siddiqi, Are JSPs Dead? I would suppose that JSPs are dead if Java was dead. But being pronounced dead by pundit engineers does not inflict Java alone. A few days ago Paul Graham made a uproar in technology circles by proclaiming that Microsoft is Dead and that Redmond rigor mortis was setting in on the software giant. Personally I think that Microsoft has soo much money in the bank that it won’t die that easy, they could just buy kidneys, patents and technology from whomever they want. But even with all that cash at hand someone asked on the Joel on Software forum if .NET was dead.

You can find premature obituaries not only for programming languages such as Python and Perl, but for the whole software industry.

Is “Free Software” Dead? No! Is commercial software dead? No! Is Enterprise Software Dead? No! No! No!

I feel that these questions are troll-level orange as they don’t help to answer any software questions or help in any design debate. These questions are not food for thought, but food for FUD. Just to answer your questions, Fortran is not dead, Perl still takes care of business, and Lisp is still alive and kicking.

JSP, and Microsoft, and all these technologies must be quoting Mark Twain right about now when he said, “The reports of my death are greatly exaggerated.” Now, can we all just get back to coding?

Daylight Standard Time Fiasco with Java

One of the hardest things to get right in Java is to figure out which class to use when working with time and dates. You have a recipe for disaster when you mix Java dates with time zone information with energy policy and a tight schedule. The U.S. Energy Policy Act of 2005 changed the start and end dates used for the Daylight Saving Time (DST). This policy had the potential negative effect of breaking millions of desktop computers, similar to the Y2K bug.

Software updates and patches where provided by software vendors such as Microsoft and Sun with plenty of time to spare before the new DST went into effect. The Microsoft patch was available as one of the thousands of security updates that you need to run on a Windows computer every so often. The Microsoft DST update was simply an exe installer. All you had to do to install the Microsoft DST update was to run it and restart your computer. Sun provided a Timezone Updater tool to patch Java with the new time zone data. The Java Timezone Updater is a jar file that you need to run from the command prompt with a series of command options. You need to run the Java Timezone Updater for each and every JRE available on the client machine! A typical client machine will have four to five different versions of the JRE.

Earlier this year when I was writing documentation and instructions for our field engineers for the new time zone change, I knew that the Java Timezone Updater was difficult for end users, even IT professionals, to run correctly. As mentioned you needed to run the updater manually from the command prompt, for every machine, for every JRE on a given machine, and there is no validation or test to notify you of success.

Well, just last week, three months after the DST change took effect one of our clients site started reporting a bug where they select April 2007 but our date widget rolls back to March 2007. After some time of head scratching, code reviews, and debugging we still were not able to reproduce the problem. After some time, it dawned on me that this might be a DST problem and sure enough we were to reproduce the bug on an updated JRE. For some reason, not yet completely explained but most likely related to human error, a whole set of computers where not patched correctly with the Java Timezone Updater. This DST bug locked out our users at this site from key UI screens at a critical time in their work flow.

We had people at the client’s site try to update our client machines with the time zone Updater, but the bug still keep popping up. We then thought of fixing this in code with some sort of conditional chunk of code but that is just trying to fix a bad bug caused by a bad policy decision with a bad software hack because of a bad JRE update. Since our desktop client runs on Java Web Start, we thought that the best solution was to have our Web Start client run only on a newer version of the JRE that already has the correct and current time zone data. This is not entirely the most accommodating solution for the end user because each client machine now needs to have the correct JRE installed or else our application will not start.

In the end, we went with that hack, to have our JNLP restrict the version of the JRE used on our application. But this is not the best solution, since you have to update every client machines, just the scenario that Web Start promised you would not have to do! The takeaway message from this experience is that dates in Java are hard.

Migrating to Java 6

I work with a large code base consisting of over 4,000 classes, with a custom ORM persistence layer, layout managers, Rapid Application Development (RAD) User Interface (UI) builder. The code is written in Java 1.4.2 but we have been considering a move to Java 5 or even up to Java 6. We currently have been debating the need, real or imagined, to migrate to Java 5. As developers, we just want to play with something new and shinny but there are business considerations that come into play.

As an exercise I migrated our code to Java 5 and then up to Java 6 and ran into few compilation and runtime errors. Upgrading to Java 5 revealed compilation errors with XPathAPI. To fix the XPathAPI error I just needed to add xalan as a dependency or import com.sun.org.apache.xpath.internal.XPathAPI. I also had a compilation error because it seems that compareTo(Object) had been removed form BigDecimal and BigInteger. As you know, Java 5 introduced enum as a keyword and wouldn’t you know I got a ton of warnings about this but this can easily be refactored.

Our system had a few runtime problems when moving to Java 5. We have some objects serialized to disk and changing the version of Java throws an InvalidClassException when reading in these objects. I also encountered some runtime exceptions thrown because of FileLock/FileChannel where none was throw before in Java 1.4.2. The errors where thrown because we had a programming error, we tried to release a file lock after the FileOutputStream for that lock had been closed.

Moving up to Java 6 I encountered a ton of compilation errors because new abstract methods have been introduced into JDBC classes such as Connection, Wrapper, and ResultSet.

Knowledge Transfer

My direct manager for the past two years has decided to leave his current position in the firm to look for other opportunities. Our CTO said that he “had served his time with the company.” To which I responded, “You make it sound as if it’s a jail sentence to work here.”

Over the last two days my manager was in the office, the whole development team spent that time with him in knowledge transfer sessions. Earlier this year I wrote about past knowledge transfer sessions with a former colleague. What follows are a few words of advice from my former manager which he gave us before his departure…

Use standard off the shelf components whenever possible, replace our custom ORM component with Hibernate, replace our client scaffolding code with NetBeans Platform. Use, reuse, and make use of open source software, don't reinvent the wheel. In essence work smarter, not harder!

When things are built right, nothing is heard.

If you want to scale up, you need to follow the chosen process. Doing software consistently requires strong discipline and a process that fits the given environment. Nothing should be done heroically, don't ever say "Ooh, there is a critical bug, let's stay overnight, get high on caffeine, and come up with a temporary hack." Software development is not a sleep over party.

Doing business software is not a challenge, developing software consistently, and scaling up is the real challenge.

An intelligent release system is integral to the success of a software organization, if you think about it, code is your bread and butter and your release system is what bakes it. You need to be able to branch, build, test, and release in a painless, hassle free, and error free way. Again, your build system is an area where you don't have to be a hero, don't prove your manhood or mad hacking skills with one off patches. In regards to the build system, and most business processes in general, if you have to do something twice, automate it.

When working in a development team with a lot of smart individuals, it often doesn't matter which solution is the 'best.' Any solution that works that is simple to maintain is good enough. The key is to debate, communicate, and follow through with commitments! As Larry Wall says, "There is more than one way to do it." Just make sure that whatever you do, you do with integrity.

I have never seen a company fail due to a specific technical problem or limitation. I have seen companies not scale up their business and talent pool.

Find a chance to do a deeper level architecture. Take on more responsibility, demand a major chuck of functionality and own it from analysis, design, implementation, and testing. When you have a certain breath of knowledge what you need is a certain anchor point so as to not get lost in the noise.

Especially for a small shop like us, be aware of the psychology of the development process. Once you bozo bit a guy, it is really hard to re-engage him. I have seen a guy become a star player, to later burn out and quit.

Business logic is hard only to understand, not to develop. Once you understand the requirements and the domain, the code follows. What is hard is building a scalable system and learning what you are building. Analyze, design, and comment, test, and think the problem first before coding.

Belated Spring Cleaning

I was reorganizing and cleaning my technical bookshelf earlier today. I have a whole bunch of books that I don’t use and are just taking space. I want to donate some of these books to my local library, maybe up and coming techies will get some use out of them. Some of these books are a little dated but others are just not useful in day to day software development. Now that I am into Ruby, Python, and Groovy I just don’t have much room in my bookshelf for Perl books. I have a whole pile of Perl books that I am going to toss out, books like Perl Cookbook, Programming Perl, and Advanced Perl Programming. The only Perl book that I am thinking of keeping is Learning Perl. I also don’t have space for antiquated web frameworks so out goes Programming Jakarta Struts, Professional Struts Applications, Enterprise JavaBeans Component Architecture, and Enterprise JavaBeans.

I think these books will be replaced with up coming Groovy titles and maybe the second edition of Agile Web Development with Rails.

Keep It Simple

Recently I started working closely with new developer to our company. I feel he over thinks problems and under estimates solutions. My personal philosophy is that less code is the best code. I guess this is the same thing that Ward Cunningham meant when he said to do the simplest thing that could possibly work. All this is another way of saying Keep It Simple, Stupid. Code should always strive to be simple.

Wiki Wiki Ward

Last week I went up to the Computer History Museum to listen to Ward Cunningham, the inventor of the Wiki Wiki Web. From what I heard, it seems like Ward was also heavily involved with Design Patterns, Extreme Programming, and CRC (Class, Responsibilities, and Collaborations) cards. Ward is a veteran computer scientist, he said that it is “a little scary when you walk in a history museum and you say, ‘I programmed in that, and that, and that.’” Speaking about those first computers that he programmed, he said that was that era where ever new machine was a new idea.

What I got from his talk, and what I have been seeing in the web, is that great code comes from communities. Great examples of this is Apache, Eclipse, Rails, and Wiki itself. Ward also suggested to surround ourselves in a community of learners.

Devgone

I wanted to continue with some of my diary entries which relate to programming. I don’t know what I was thinking when I wrote this one here dated the 8th of August of 2005:

A coworker moved on to greener code, before his last day we had a lot of knowledge transfer sessions. This guy was the process guy and a stickler for maintaining JUnit tests. During one of the knowledge transfer sessions he said, “Testing saved my butt. I test often because I'm not that good of a developer.” I have to agree with his statement, testing saves your butt. Because he test so much he is a better developer. JUnit makes better developer out of us all. Do it early, do it often, like making love. If only sex had a green bar at the end to indicate your success! He also asked of all of us staying behind, “If you can’t write the unit test for the thing you are doing, how do you know what you are doing?” Again, like having sex! If you don’t have a JUnit for it, I don’t think you know what you are doing!

Less Talk, More Code

Why are technology books sooo voluminous, they are usually 500 page coffee tables as opposed to a coffee table book. I feel that most of the pertinent information can be condensed to a 40 page manual or how-to cookbook if they remove all the fluff. When dealing with new technology, lets say Struts, a developer already has, or should be assumed to have, adequate knowledge of Java, the internet, and HTML. Not a single page should be dedicated to the history of Java or HTML in a Struts book and such information should be available online. Books that deal with web application frameworks such as Struts or Ruby of Rails all have the same first chapter on the MVC pattern that these frameworks are built on. These books should reference some wiki in the sky for these background concepts and histories. From my experience the installation chapter is always a waste. Most applications are usually double click installation and usually contain a README file. I almost feel that books should ask you, do you know Java, do you know MVC, do you know HTML, if you answer yes to all these then skip to chapter 7! But maybe this is a publishing issue, I am sure there is a certain number of pages and dimensions that a book should have to be priced at a certain price range and be guaranteed a certain shelf space. All I want is less talk (I don't care for the kayaking allegories Bruce Tate starts his chapters with) and more code. I really want a cross between the O'Reilly cookbook and hacks series. The cookbook series has plenty of code and the hacks show you some really interesting examples.

ARRR

What is up with file extensions ending in ar. In a daily and weekly basis I have to deal with tar, jar, rar, war, and ear files. I think we still need a car extension for Nascar fans, and var for the JavaScript web devheads, mar for the Spanish speakers, par for all the golf fans, oh wait EJB 3.0 makes use of a new par extension for its persistence entity beans.. To put everything into perspective I think we still need a far file extension. If I would to come up with a new compression algorithm I would use an extension that denotes my favorite hang out; bar.

The Hype Framework

In the words of Public Enemy, don't believe the hype. Technology is full of jargon and full of hype. The current hyped up terms are full-stack, web framework, Model 2, Web2.0, and Ajax. Every language under the sun has a web Model 2 based fully-stack framework these days. I am coming out with what I call the Hype Framework. The Hype Framework has everything you can dream off in whatever language you want but you have to implement it yourself.

On a more serious note here is a list of web frameworks that have received some attention, in no particular order. In the Java space you have Struts, WebWork, Spring, Wicket, Tapestry, and tons more. In Python CherryPy, Quixote, Twisted, WebWare and Zope come to mind. And of course, Ruby has Rails.

As a software engineer, I feel that every developer should know at least two languages, for example Java and Ruby (Java and C# count as one language. C# is the bizarro version of Java), and at least one of the above frameworks.

Copyright

Code Rage 2.0: Rants of Code and Other Essays is Copyright © 2008, Juixe.com. It is licensed under the terms of the Creative Commons Attribution-Share Alike 2.5 agreement: http://creativecommons.org/licenses/by-sa/2.5

Money! is Copyright © 2005, Tracy Olson, http://flickr.com/photos/tracy_olson/. It is licensed under the terms of the Creative Commons Attribution-Share Alike 2.0 agreement: http://creativecommons.org/licenses/by-sa/2.0/deed.en

Spaghetti Code is Copyright © 2007, Balakov, http://flickr.com/photos/balakov/. It is licensed under the terms of the Creative Commons Attribution-Noncommercial 2.0 Generic agreement: http://creativecommons.org/licenses/by-nc/2.0/deed.en

Rails Code is Copyright © 2008, Davestfu, http://flickr.com/photos/davestfu/. It is licensed under the terms of the Creative Commons Attribution-Noncommerical Share Alike 2.5 agreement: http://creativecommons.org/licenses/by-nc-sa/2.0/deed.en

Da Vinci Code Cryptex HDR is Copyright © 2006, David Reeves, http://flickr.com/photos/dcreeves2000/. It is licensed under the terms of the Creative Commons Attribution-Noncommercial 2.0 Generic agreement: http://creativecommons.org/licenses/by-nc/2.0/deed.en