



Quarterly Spin

SpiderLogic

Volume I, Issue I

January I, 2007

Steve's Excellent Indian Adventure

By Steve Kronsoble

I recently returned from a trip to India, where I visited our SpiderLogic development center in Pune and also met with various companies and contacts in Mumbai (Bombay) and Bangalore. Quite a country and quite a trip – both from an IT perspective and a personal perspective.

First – the IT view. People here in the US, and especially the IT industry, think India and Software Engineers are synonymous and therefore both plentiful. To paraphrase Bill Gates, if you are looking for a one-in-a-million, go to India – there are 1000 of them. Well our bar is high but not that high! Nonetheless, the labor market in India has become very tight – very much a sellers market. No matter how fast the universities are pumping out computer science and engineering graduates, the companies there (especially the large multi-nationals) are gobbling them up. And caveat emptor to



the buyer. For various reasons, experienced software engineers seem to move out of true programming much quicker than in the US. (where are the gray beards?) This means that many of the software engineers are “freshers” – kids right out of college with no real world experience. So finding and then taking care of employees – train-

ing, mentoring, challenging, motivating is as important in India as it is in the US.

See India page 2



“If you are looking for a one-in-a-million, go to India – there are 1000 of them.”

- Bill Gates

Inside this issue:

<i>Relentless ... Build Automation</i>	1
<i>Steve's Excellent Indian Adventure</i>	1
<i>Miser's Maxims</i>	2
<i>Meta vs. Intentional Programming</i>	2
<i>Puzzle of the Month</i>	3
<i>Technology of Sin City</i>	4



Relentless Build Automation

By Geoff Lane

How far is going too far when you are doing build automation? My answer to that question is that it's basically not possible to go too far. Any task that you need to do more than three times deserves automation. Three times is my heuristic. The longer and more complex the task, the smaller the number you should allow yourself to do it by hand. Compiling for

example is a pain if you don't automate it, so you automate it the first time. Computers are good at one thing, running algorithms. If you are doing those algorithms by hand, you're wasting your computer's potential.

There are two major reasons to automate: Reducing errors and being lazy.

Reducing errors is easy. We want to have a repeatable, dependable process that anyone can use to achieve the same results. If your deploy master has to know 15 steps and the ins and outs of some complex process, what happens when she goes on vacation? You don't do deployment.

See Relentless page 3



India - continued from page 1



Steve Kronsoble and Murali Iyer on route to Pune, India

Second – the personal view. India is a sensory feast! The wafting outdoor conversations of all the many people, the colorful clothes, the rich food, the omnipresent never completed construction projects, the dust and the dirt, the lingering aromas and odors, and the road traffic. Oh the traffic. Buses, trucks, cars, auto rickshaws, mopeds, bicycles, pedestrians, and cows sharing the un-laned roads in a controlled chaos with the continuous sound of horns providing seemingly a sort of divine direction. Yet, I never saw an accident or road rage. The people are just wonderful. To be hon-

est, it was only after 6 days that I finally found the courage to cross a street by myself. As a good friend of mine told me, it is no wonder that India is the home to so many meditative practices. The only place for peace is inside your head!

Now for the roads. Maybe, just maybe we could take a few of those software engineers and turn them into civil engineers!

E-mail Steve Kronsoble
skronsoble@spiderlogic.com



Kartik Rindani and Gautam Kasturi of SpiderLogic - visiting Bangalore



Meta- vs. Intentional Programming

By James Roome

After some reflection, I have decided that meta-programming and intentional programming are two very different things.

Meta-programming is just about using/creating custom programming languages. The only intent meta-programming provides is just that which occurs due to the domain specificity of the languages in use. The actual code generated is still very mundane. Nothing special here. Intentional programming to me

means doing much more than just this. To me, it is about leveraging the intent as much as possible. Part of getting the intent is raising the level, and meta-programming is a great way to do this. But intentional programming can be so much more than writing software at different levels. If intent has been encoded in such a way that we can leverage it, then the possibilities are huge.

For Example:

When about to perform a lower level db query we can, at this point in time look and see what the higher level intent(s) is. Is the intent to just display this data on the UI? Is the intent to do X with the data?

We seem to have got into this rut of having very low expectations of what we can do, creating static code that's very dumb, that doesn't know much. And then,

make up for all these deficiencies.

When we (the developers) write the code, we are taking into consideration the intent and the implementation of many layers. But what we create, is code that doesn't know the big picture, and couldn't help but be brittle and prone to errors. Heck, a person given three parameters and an algorithm in isolation couldn't do much better.



Miser's Maxims by Dan Miser

1. Everything is easy when all you're doing is drawing circles and squares.
2. It takes 3 times to perfect something.
3. Anybody can look good on paper
4. When playing safe, be sure it's safe.
5. If you don't know the shot, don't take the shot.
6. Good enough to work isn't "Good Enough".
7. If you want to act like a prima donna, produce like a prima donna.
8. You can't stand on principle when you have none.
9. Being busy is no excuse for a shortcut.
10. A bug is no reason to lose sight of logic.
11. Nothing in life is a failure if you learn from it.
12. Think Less. Know More.

These are sayings that have evolved over time. I thought I'd share them here. They are meant to promote thought.

Check out Dan's blog at distribucon.com/blog

Relentless continued from pg. 1

By automating, you allow other people to perform the same task and you keep your computer working for you.

But being lazy? Yes! [A truly great computer programmer is lazy](#) and does everything in their power to avoid having to do work that they've already done before. Likewise, if you've done it 10 times, doesn't it get boring to do the same thing over and over again? Wouldn't you rather be getting coffee or reading blogs while your computer works at what it does best? In addition to general laziness, automation will make your life easier when you want to do other things that you know you should be doing like unit testing, integration testing and functional testing.

What to Automate

Going back to my introduction, I say automate everything that you or someone new coming onto a project will need to do more than a few times. A little bit of effort up front can save every member of a team a lot of effort later on. So, as a punch list to get you started:

- Compiling code
- Packaging code into JARs, WARs, etc
- Running unit tests

Most people doing Java development will automate code compilation and packaging all of the time. Some of the times, they'll automate unit tests, sometimes deployment, and rarely a few other things. We usually use some excellent script-based tools for that kind of thing like [Ant](#) or [Ant](#) or [Maven](#). Even if you do all of these things, is that going far enough? I think that for most projects the answer is no.

If you are slightly more ambitious, you will also automate:

- Integration testing
- Functional testing
- Deployment

There, now you've got all of your day-to-day activities automated. So you must be done, right?

Databases are the bane of automation. People don't take the time to script their database creation or to script their default or testing data. When you don't script your database, you make integration and functional testing very difficult. It's really easy to write functional tests and integration tests if they rely on the data being in a known state. If they do not, then running them is very difficult because the data underneath is always changing. That usually means that integration and automated functional testing are deemed too hard and are not done. But don't blame the tests. You're avoiding a little bit of work that if you did it, would allow you to avoid A LOT of work. (I think little jobs that prevent you from having to do big jobs are a good tradeoff.)

Generating your database will make new team members' lives a breeze. They should be able to check out a project from source control and, with one simple command, generate a database, compile the code, run the tests and know with confidence that everything is working for them. When they make any changes to the project, they can repeat this process to know with confidence that they haven't broken anything. Likewise in day-to-day development, being able to get back to a known state is going to give you the ability to develop with more confidence and test with ease.

[Ant](#) has a core task for running [SQL](#) queries, so you don't even need any other tools to accomplish this. One problem that you might run into is that running DDL commands in JDBC can cause some problems. I found a [great article on executing PL/SQL with Ant](#) that you can probably apply to other databases as well. It allows you to run more database specific queries to build your schema from a blank database. This helped me overcome the last hurdle in a recent project to get it so I had the entire database scripted from scratch. So check it out.

I find that having a few standard Ant tasks makes this a breeze:

- A task to create tablespaces (if you have separate tablespaces for each database)
- A task to create a user
- A task to generate the full schema
- A task to delete the user and drop the schema

These tasks can be woven together to create the full database. To support development and testing, it is also handy to have loadable datasets around. Either maintain a list of SQL INSERT statements or use a tool like [DBUnit](#) to load the data into your database. Often it can be nice to have a few sets of base data:

- Unit testing data
- Functional testing data
- Demo data for showing off to clients and managers

The only thing lacking is an easy way to dump a schema from a working database so that you can use nice GUI tools to modify the database and then dump it out using an easy Ant target. Even with this, you'll likely need to maintain change scripts so that you can apply them to working production systems without wiping out all of the data. I guess, since I'm complaining, that will have to be my next side project.



Yes! ...
 "A truly great programmer is lazy"

With a little bit of effort in going all the way with automation, you can make it easy for new people to come onto a project, you can make it easier to create and run all kinds of tests, and you can make the entire development process easier and more confident. The tools exist to help make automation easy, so why not?

What things do you like to automate that you see people skipping on projects?

Check out [Geoff Lane's blog](#) at [zorched.net](#)

Here is the puzzle. Good luck!

		6	9		8
	9	4	5		2
		8		9	4
9		3			2
1	4			6	5
	2		1		3
3	5		9		
6		5	4	7	
	1		6	3	



Medium Puzzle 5,142,893,607 -- Select a puzzle...



SpiderLogic
10000 Innovation Drive
Milwaukee, WI 53226

Phone: 414.290.8015
E-mail: info@spiderlogic.com

Where architects code and coders
architect!



Technology of “Sin City” By Chris Peterson

As a software developer, I always keep an eye out for ways computers impact life. Las Vegas has seen multiple changes due to the cpu. I have only been going to Vegas for nine years, so these observations are based on that time frame.

Slot Machines

One of the most obvious place computers have impacted Vegas is with slot machines. Gone are the days of the reeled machines with a single pay line. Thanks to the increased speed of the cpu now you have video slot machines that have up to 100 pay lines. Each line requires a separate bet and thus making the slot machine the number one money maker for Vegas casinos.

High-speed networks have allowed slot machines to be linked across many different casinos and thus allowing larger progressive jackpots. The way the jackpots work is simple for every coin in a linked machine; a percentage gets added to the progressive. When you have many of these machines linked together, that jackpot increases rather quickly.

Table Games

Table games are always changing. Casinos are always adding new games to the floor to go along with the basic games of blackjack, craps,

and roulette. For the game play, there is very little technology. In fact casinos will not let you sit at a table while using a cell phone, mp3 player or any other portable electronic device (poker room is an exception). Behind the table games, however, is a plethora of technology including some of the highest level security and player-tracking computer systems.

Lets start with the player tracking system. how it works is simple. You give the pit boss your players' card, and they go over to a computer and pull up your history. They can learn everything about you as a player. Things like your average bet, the times and games you like to play, if you are an advantage player, what you drink, and most importantly, your importance to the casino as a player. As you play, the bosses continue to add data to the system. Over time the casino has a very clear picture of their players and thus who deserves free room comp, and who deserves the buffet comp.

Security

Casino security is the most high tech security you will find. There are very few places you can go without the eye in the sky not being able to track you. Also, rarely can you get

to places in a casino you do not belong. Every casino has hundreds of video devices, and linked with that is software that can scan a persons face and pull up possible matches. The computers are smart enough to analyze video of game play and come to a determination if there are any shenanigans going on and can alert security personal to watch and notify pit bosses. The security is so good that after 9/11 airports went to the casinos to learn about security.

Technology you don't see

Other ways computers have helped casinos become even more successful is by tracking how money is made. If you walk around a casino, you see that the table games are arranged in a pit and slot machines are arranged in banks. Casinos have software that track how well individual table games, pits, and groups do. They also track the same for slot machines. By tracking the dollars, the casino knows which games customers want to play and which games are flops. With this knowledge, a casino can maximize its earning potential.

Checkout Chris' blog:

