

# Greater Atlanta Software Symposium

Atlanta Marriott Perimeter Center

May 15 - 17, 2009

<http://www.nofluffjuststuff.com/conference/atlanta/2009/05/index.html>

Fri, May. 15, 2009			
	Salon D	Salon E	Washington/Jefferson
12:00 - 1:00 PM	REGISTRATION		
1:00 - 1:15 PM	WELCOME		
1:15 - 2:45 PM	The Amazing Groovy Weight-loss Plan Scott Davis	Common AntiPatterns and How To Avoid Them Mark Richards	The Busy Java Developer's Guide to Java7 Ted Neward
2:45 - 3:15 PM	BREAK		
3:15 - 4:45 PM	Groovy XML Ninja Skills Scott Davis	On Being a Software Architect Mark Richards	The Busy Java Developer's Guide to Java Platform Security Ted Neward
4:45 - 5:00 PM	BREAK		
5:00 - 6:30 PM	Dim Sum Grails: A Sampler of Practical Non Database-Driven Grails Applications Scott Davis	Transaction Pitfalls and Strategies Mark Richards	The Busy Java Developer's Guide to Advanced Platform Security Ted Neward
6:30 - 7:15 PM	DINNER		
7:15 - 8:00 PM	Keynote: by Neal Ford		

Sat, May. 16, 2009			
	Salon D	Salon E	Washington/Jefferson
8:00 - 9:00 AM	BREAKFAST		
9:00 - 10:30 AM	Architecture and Scaling Ken Sipe	Introduction to JMS Mark Richards	Lizard Brain Web Design Scott Davis
10:30 - 11:00 AM	BREAK		
11:00 - 12:30 PM	Spring 2.5/3 Without XML Ken Sipe	Advanced Topics in JMS Mark Richards	Web 2.0 Checklist: Deconstructing Modern Websites Scott Davis
12:30 - 1:30 PM	LUNCH		
1:30 - 3:00 PM	7 Habits of Highly Effective Developers Ken Sipe	Real-world Refactoring Neal Ford	The Busy Java Developer's Guide to Collections Ted Neward
3:00 - 3:15 PM	BREAK		
3:15 - 4:45 PM	Java Memory, Performance and the Garbage Collector Ken Sipe	Test Driven Design Neal Ford	The Busy Java Developer's Guide to Scala: Objects Ted Neward
4:45 - 5:45 PM	BIRDS OF A FEATHER SESSION		

Sun, May. 17, 2009			
	Salon D	Salon E	Washington/Jefferson
8:00 - 9:00 AM	BREAKFAST		
9:00 - 10:30 AM	Hacking - The Dark Arts Ken Sipe	Emergent Design & Evolutionary Architecture Neal Ford	How to Fail with 100% Code Coverage Stuart Halloway
10:30 - 11:00 AM	MORNING BREAK		
11:00 - 12:30 PM	Security Boundaries Ken Sipe	Dynamic JVM Languages in the Enterprise Pratik Patel	IZero: Starting Projects Right Stuart Halloway
12:30 - 1:15 PM	LUNCH		
1:15 - 2:15 PM	EXPERT PANEL DISCUSSION		
2:15 - 3:45 PM	JConch and java.util.concurrent: The JVM Concurrency Toolset Robert Fischer	Construction Techniques for Domain Specific Languages Neal Ford	Java.next #1: Common Ground Stuart Halloway
3:45 - 4:00 PM	BREAK		
4:00 - 5:30 PM	Architecting Code for Concurrent Execution: Theory and Practice Robert Fischer	Real-world JEE performance tuning: Tips n' Tricks Pratik Patel	Programming Clojure Stuart Halloway

# Greater Atlanta Software Symposium

Atlanta Marriott Perimeter Center

May 15 - 17, 2009

<http://www.nofluffjuststuff.com/conference/atlanta/2009/05/index.html>

## **The Amazing Groovy Weight-loss Plan by Scott Davis**

"The central enemy of reliability is complexity." (Dr. Daniel Geer) Java is a powerful programming language. A smart developer can do nearly anything with Java. So the next question is, "How quickly can it be done? How many lines of code does it take to do common tasks?" Groovy greases the wheels of Java by decreasing the complexity of the language while preserving the raw power. At first glance, you might think that this talk is simply about how Groovy drastically reduces the lines of code you need to write. What this talk is really about is bringing simplicity, clarity, readability, and yes, beauty to your source code.

## **Groovy XML Ninja Skills by Scott Davis**

"XML is like violence: if it doesn't solve your problem, you aren't using enough of it." (Anonymous) XML is everywhere. Whether you are dealing with local configuration files (web.xml, struts-config.xml) or remote web services (SOAP, REST, RSS, Atom), the modern software developer needs to be able to request, slice, and dice XML with ease. That requires a set of razor-sharp tools that reduce the inherent complexity of the problem, not multiply it. Once you see XML tremble in fear at the awesome power of Groovy, you'll wonder what you ever did without it.

## **Dim Sum Grails: A Sampler of Practical Non Database-Driven Grails Applications by Scott Davis**

"The proof of the pudding is in the eating. By a small sample we may judge of the whole piece." (Miguel de Cervantes Saavedra) Most Grails tutorials demonstrate how easy it is to build simple CRUD (Create/Retrieve/Update/Delete) applications. While skinning a database with a web front-end is undeniably one beneficial aspect of Grails, it isn't the only thing Grails is good for. As you'll see here, Grails can be used to build a wide variety of web applications. You won't see a single HTML table with "edit" and "delete" links, I promise.

## **Lizard Brain Web Design by Scott Davis**

"There's an old story about the person who wished his computer were as easy to use as his telephone. That wish has come true, since I no longer know how to use my telephone." (Bjarne Stroustrup) The "lizard brain" is the oldest part of the human brain -- the part responsible for autonomic functions like breathing, heart rate, and navigating websites. OK, maybe not that last part, but your website should be easy to use. Stupid easy. Lizard brain easy. Any time your user spends figuring out how to do something -- even for a split second -- is wasted time due to poor design. Inspired by Steve Krug's book "Don't Make Me Think", this talk answers the question, "Why is that website so hard to use?"

## **Web 2.0 Checklist: Deconstructing Modern Websites by Scott Davis**

"The challenge of modernity is to live without illusions and without becoming disillusioned." (Antonio Gramsci) There are plenty of sarcastic "Web 2.0" checklists out there -- be perpetually in BETA, when in doubt add rounded corners, etc. While we can all laugh at the superficial aspects of the Web 2.0 revolution, there are plenty of serious aspects to it as well. Is your website mash-up friendly or hostile? Do you tell your visitors when things change (via RSS or Atom syndication), or do you expect them to check in daily for updates? Is your website a silo or a part of a larger ecosystem?

## **JConch and java.util.concurrent: The JVM Concurrency Toolset by Robert Fischer**

JConch is a library that provides a few high-level tools for high-concurrency environments on the JVM. The java.util.concurrent package in the Java standard library provides low-level structures for managing concurrent communication. Learn here how to use both of them in order to produce clean, highly-concurrent, and highly-tunable code.

## **Architecting Code for Concurrent Execution: Theory and Practice by Robert Fischer**

The power of multicore machines and cloud computing is all dependent on an application's ability to successfully leverage concurrency. Although concurrency has traditionally been considered fatally difficult in Java, a few simple architecture principles can make all the difference. This session will review some of those principles in both theory and practice.

## **Keynote: On the Lam from the Furniture Police by Neal Ford**

When you were hired by your current employer, you may think it's because of your winning personality, your dazzling smile, or your encyclopedic knowledge of [insert technology here]. But it's not. You were hired for your ability to sit and concentrate for long periods of time to solve problems, then placed in an environment where it's utterly impossible to do that! Who decides that, despite overwhelming evidence that it's bad for productivity and people hate it, that you must sit in a cubicle? The furniture police! This keynote describes the frustrations of modern knowledge workers in their quest to actually get some work done, and solutions for how to gird yourself against all those distractions. I talk about environments, coding, acceleration, automation, and avoiding repetition as ways to defeat the mid-guided attempts to sap your ability to produce good work. And I give you ways to go on the lam from the furniture police and ammunition to fight back!

### **Real-world Refactoring by Neal Ford**

Refactoring is a fine academic exercise in the perfect world, but we don't really live there. Even with the best intentions, projects build up technical debt and cruffy bad things. This session covers refactoring in the real world, at both the atomic level (how to refactor towards composed method and the single level of abstraction principle) to larger project strategies for multi-day refactoring efforts. This talk provides practical strategies for real projects to effectively refactor your code.

### **Test Driven Design by Neal Ford**

Most developers think that "TDD" stands for Test-driven Development. But it really should stand for "Test-driven Design". Rigorously using TDD makes your code much better in multiple ways.

### **Emergent Design & Evolutionary Architecture by Neal Ford**

Most of the software world has realized that BDUF (Big Design Up Front) doesn't work well in software. But lots of developers struggle with this notion when it applies to architecture and design. Surely you can't just start coding, right? You need some level of understanding before you can start work. This session describes the current thinking about emergent design & evolutionary architecture, including both proactive (test-driven development) and reactive (refactoring, composed method) approaches to discovering design. The goal of this talk is to provide nomenclature, strategies, and techniques for allowing design to emerge from projects as they proceed, keeping you code in sync with the problem domain.

### **Construction Techniques for Domain Specific Languages by Neal Ford**

This talk covers language techniques in Java, Groovy, and Ruby on how and why to create DSLs, and also covers the very important topic of implicit context, and how language constructs can allow you to write less verbose and more expressive code.

### **How to Fail with 100% Code Coverage by Stuart Halloway**

Over the last few years, we have taken dozens of projects to 100% coverage, and there are still plenty of things that can go wrong. We will look at examples the various problems, and show how to prevent them from infecting your project.

### **IZero: Starting Projects Right by Stuart Halloway**

If an iteration is the heartbeat of an agile development process, then Iteration Zero (IZero) creates the heart. While you can (and should) retrospect and adjust throughout the software lifecycle, few things are as valuable as a good start. In this talk, you will learn how we run Iteration Zero at Relevance.

### **Java.next #1: Common Ground by Stuart Halloway**

In this talk, we will explore and compare four of the most interesting new JVM languages: Clojure, Groovy, JRuby, and Scala. Each of these languages aims to greatly simplify writing code for the JVM, and all of them succeed in this mission. However, these languages have very different design goals. We will explore these differences, and help you decide when and where these languages might fit into your development toolkit. For more information see <http://blog.thinkrelevance.com/2008/8/4/java-next-common-ground>.

### **Programming Clojure by Stuart Halloway**

Find out why Clojure is Java.next: \* Clojure provides clean, fast access to all Java libraries. \* Clojure provides all the low-ceremony goodness you know and love from dynamic languages such as Ruby and Python. \* Clojure includes Lisp's signature feature: Treating code as data through macros. \* Clojure's

emphasis on immutability and support for software transactional memory make it a viable option for taking advantage of massively parallel hardware.

### **The Busy Java Developer's Guide to Java7 by Ted Neward**

Even though the Java 7 JSR has yet to be formed, some interesting things are beginning to emerge from Sun about what Java7 may include when its formal release contents are finally made public.

### **The Busy Java Developer's Guide to Java Platform Security by Ted Neward**

Permissions, policy, SecurityExceptions, oh my! The Java platform is a rich and powerful platform, complete with a rich and powerful security mechanism, but sometimes understanding it and how it works can be daunting and intimidating, and leave developers with the basic impression that it's mysterious and dark and incomprehensible. Nothing could be further from the truth, and in this presentation, we'll take a pragmatic, code-first look at the Java security platform, including Permissions, the SecurityManager and its successor, AccessController, the Policy class and policy file syntax, JAAS, and more.

### **The Busy Java Developer's Guide to Advanced Platform Security by Ted Neward**

So you know the platform security model, and now you want to use it in new and interesting ways, like creating a custom Policy implementation, a custom Permission, or create a custom security context in which code will execute. Perhaps you even wish to make certain objects accessible only to those with the right permissions, or cryptographic key. Nothing could be easier, despite Java security's reputation as a dark and arcane place. **Prerequisite:** *The Busy Java Developer's Guide to Platform Security*

### **The Busy Java Developer's Guide to Collections by Ted Neward**

For so many Java developers, the `java.util.*` package consists of List, ArrayList, and maybe Map and HashMap. But the Collections classes are so much more powerful than many of us are led to believe, and all it requires is a small amount of digging and some simple exploration to begin to "get" the real power of the Collection classes.

### **The Busy Java Developer's Guide to Scala: Objects by Ted Neward**

Scala is a new programming language incorporating the most important concepts of object-oriented and functional languages and running on top of the Java Virtual Machine as standard "dot-class" files. Sporting the usual object-oriented concepts as classes and inheritance, Scala also offers a number of powerful functional features, such as algebraic data types, immutable objects by default, pattern matching, closures, anonymous functions and currying, and more. Combined with some deep support for XML generation and consumption, Scala offers Java programmers an opportunity to write powerful programs with concise syntax for a new decade of Java programming.

### **Dynamic JVM Languages in the Enterprise by Pratik Patel**

Dynamic languages running on the Java Virtual Machine are starting to gain traction for software development, specifically for large enterprise projects. This session explores obstacles to introducing dynamic languages into the enterprise, example applications that can ease the way, and issues surrounding integrating a dynamic language to Java projects. Using several code examples that demonstrate the power of using a dynamic language like Jruby or Groovy, attendees will gain insight into how dynamic languages are making in-roads to the enterprise. This session focuses on non-GUI related usages – whereas most people think of dynamic languages for Web development. The target audience for this session is enterprise developers and enterprise architects.

### **Real-world JEE performance tuning: Tips n' Tricks by Pratik Patel**

Performance tuning any application is a black art that can consume much time. Fortunately, Java has many tools that can aid in this effort. There also are a number of basic tips that can help to analyze and fix performance problems. The Java memory model is usually something that you don't need to tune, but for high performance applications it is necessary to tweak. While there are a number of advanced things that can be done to performance tune an application, we'll discover that the simple, basic things are all that are usually needed to make your apps fly.

### **Common AntiPatterns and How To Avoid Them by Mark Richards**

In the book "97 Things Every Software Architect Should Know" (O'Reilly, 2009) I wrote about the importance of design patterns as a useful means of communication between architects and developers. Equally important to patterns is an understanding of AntiPatterns - things that we repeatably do that produce

negative results. AntiPatterns are used by developers, architects, and managers every day and are one of the main factors that prevent progress and success. In this session we will look at some of the more common and significant development and architecture antipatterns. Through coding and design examples, you will see how these antipatterns emerge, how to recognize when the antipattern is being used, and most importantly, how to avoid them. By attending this session, you will be part of a movement to reduce the AntiPattern catalog from hundreds of entries to only a few. **Prerequisite:** None

### **On Being a Software Architect by Mark Richards**

One way to stop a conversation dead while at a party or gathering is to mention you are a software architect. Why? Because it takes about an hour (complete with Powerpoint slides) to explain what you do for a living. By then the person you are talking to is so bored they would rather sit in a corner licking nine-volt batteries. The problem is that no one inside or outside our industry really knows what a software architect is or what they do. In this highly interactive (and slightly humorous) session we will take a deep dive into the role a software architect plays in the IT industry. We will explore the characteristics an architect needs to have, and the elements that make a good architect and a bad architect. Through amusing antidotes and real-world examples, we will see how to become an effective software architect and help shape the industry in terms of the role and title of software architect. **Prerequisite:** None

### **Transaction Pitfalls and Strategies by Mark Richards**

In previous years I have given sessions related to my book "Java Transaction Design Strategies", where I have reviewed the basics of programmatic and declarative transactions and outlined the basic patterns described in the book. In this new session for 2009 I will focus on some of the pitfalls encountered while dealing with transactions and then how to develop an effective transaction strategy. I will start this session by describing and illustrating some of the common pitfalls I continue to see in both Spring and EJB. I will then describe four common transaction strategies you can use and implement, including a transaction strategy for high-speed transactions, a transaction strategy for client orchestration, a transaction strategy for use with API's, and finally a strategy for highly concurrent environments. Note: This session assumes you know a little bit about transactions and have been using them in either Spring or EJB. It is not intended to be an introductory session on how transactions work. You can obtain a free PDF download of my transaction book at <http://www.infoq.com/minibooks/JTDS> to quickly come up to speed with transactions. **Prerequisite:** Java, Spring or EJB; some knowledge of transactions and JTA.

### **Introduction to JMS by Mark Richards**

There's no doubt about it - messaging is quickly becoming a standard part of most application architectures, particularly as more and more companies struggle to find ways to integrate heterogeneous environments due to mergers, acquisitions, or to streamline existing application portfolios. The Java Message Service (JMS) API allows Java applications to implement messaging using a standard API, therefore removing the dependency of any particular messaging provider. In this introductory session we will take a look at the basics of messaging and the JMS API. I will start by discussing the different messaging models, the structure of a basic JMS message, and the JMS API interfaces and how they interrelate. Then through interactive coding I will show the basics of sending and receiving messages using the point-to-point messaging model and how to do request/reply processing. NOTE: this session is meant to be an introduction to messaging and JMS - no prior JMS or messaging experience is needed for this session. **Prerequisite:** None

### **Advanced Topics in JMS by Mark Richards**

This session covers some of the more advanced features of JMS messaging, and is intended for those who are familiar with JMS and messaging in general. Some of the topics I will be covering in this session include message grouping (where I will demonstrate sending a large JPG image using messaging), transacted sessions, client-based acknowledgement, and some various messaging design considerations and things to watch out for from a design and coding perspective. I will be doing live coding demonstrations to illustrate the techniques described in this session. Although this session is entirely JMS provider agnostic, I will be using ActiveMQ, a popular open source JMS provider, during the live coding demonstrations. **Prerequisite:** Some knowledge of messaging and JMS would be helpful

### **Architecture and Scaling by Ken Sipe**

Scale... what is scale... how do you applications which are scalable. How do you know if the application scales?

### **Spring 2.5/3 Without XML by Ken Sipe**

Spring 3 is brand spanking new, with a number of fantastic features. With growth of large and complex Spring applications which struggle with xml manageability and with the added pressure of Guice and SEAM there is a push for less XML, with solution leaning towards annotations. Spring 2.5 adds to the toolset provided in Spring 2.0 to provide a development environment where XML is greatly reduced... or eliminated if you so choose.

### **7 Habits of Highly Effective Developers by Ken Sipe**

Thoughts lead to words, words lead to action, actions lead to habits. In this session we'll sharpen the development saw in the process of understanding what makes a hyper-productive programmer. The focus will consist of developer habits and development processes.

### **Java Memory, Performance and the Garbage Collector by Ken Sipe**

You are using Java, whew!!! No need to worry about memory, the garbage collector will handle that. Those who have had a memory issue in Java are not so naive any more. Often memory utilization and heap sizes are an after thought and are not recognized until the application is in production, often caused by application uptime, production request volume or production sets of data. When the OutOfMemory Error occurs, often the science of development seems to brake down and knobs are turned. First the (-mx) maximum heap space gets adjusted... More is better right. The next OutOfMemory, heads start scratching, code reviews start in earnest, and Google gets several new hits. Did you know that it is possible to get an OutOfMemory error without running out of heap space?

### **Hacking - The Dark Arts by Ken Sipe**

A live Hacking demonstration exposing the tools and techniques used by Hackers.

### **Security Boundaries by Ken Sipe**

Security is a large concern in today's world of software development. Security is a multi-dimensional problem requiring skills at a number of different levels. This session is a security overview of a typical Java web development stack.