

# Greater Atlanta Software Symposium

Atlanta Marriott Perimeter Center

October 23 - 25, 2009

<http://www.nofluffjuststuff.com/conference/atlanta/2009/10/home>

## Fri, Oct. 23, 2009

	Salon A-C	Salon D	Salon E	Monroe/Jackson	Jefferson/Washington
12:00 - 1:00 PM	REGISTRATION				
1:00 - 1:15 PM	WELCOME				
1:15 - 2:45 PM	Common AntiPatterns and How To Avoid Them Mark Richards	JSF 2.0: An Introduction David Geary	Groovy XML Ninja Skills Scott Davis	Effective Java Venkat Subramaniam	IZero: Starting Projects Right Stuart Halloway
2:45 - 3:15 PM	BREAK				
3:15 - 4:45 PM	On Being a Software Architect Mark Richards	JSF 2.0: Advanced Topics David Geary	Groovy Testing Scott Davis	Programming Scala Venkat Subramaniam	Taking Agile From Tactics to Strategy Stuart Halloway
4:45 - 5:00 PM	BREAK				
5:00 - 6:30 PM	Transaction Pitfalls and Strategies Mark Richards	Flex for Java Developers David Geary	RESTful Grails Scott Davis	Cleaning up Code Smell Venkat Subramaniam	Agile, Relevance Style Stuart Halloway
6:30 - 7:15 PM	DINNER				
7:15 - 8:00 PM	Keynote: Ted Neward				

## Sat, Oct. 24, 2009

	Salon A-C	Salon D	Salon E	Monroe/Jackson	Jefferson/Washington
8:00 - 9:00 AM	BREAKFAST				
9:00 - 10:30 AM	Introduction to JMS Mark Richards	GWT fu, Part 1 David Geary	Tackling Concurrency on the JVM Venkat Subramaniam	Lizard Brain Web Design Scott Davis	Visualizations for Code Metrics Neal Ford
10:30 - 11:00 AM	BREAK				
11:00 - 12:30 PM	Advanced Topics in JMS Mark Richards	GWT fu, Part 2 David Geary	Building External DSLs Venkat Subramaniam	Web 2.0 Checklist: Deconstructing Modern Websites Scott Davis	Real-world Refactoring Neal Ford
12:30 - 1:30 PM	LUNCH				
1:30 - 3:00 PM	Emergent Design & Evolutionary Architecture Neal Ford	The Busy Java Developer's Guide to ClassLoaders Ted Neward	JavaScript: the Good, the Bad, and the Ugly Nathaniel Schutta	Java.next: Clojure, Groovy, JRuby, and Scala Stuart Halloway	Estimating vs. Guessing - How Agile Teams Estimate Their Work David Bock
3:00 - 3:15 PM	BREAK				
3:15 - 4:45 PM	What's New in Spring 3 Ken Sipe	Virtualization for development Pratik Patel	Making Web Apps Suck Less Nathaniel Schutta	Clojure Stuart Halloway	Surviving Middle Management David Bock
4:45 - 5:45 PM	BIRDS OF A FEATHER SESSION				

## Sun, Oct. 25, 2009

	Salon A-C	Salon D	Salon E	Monroe/Jackson	Jefferson/Washington
8:00 - 9:00 AM	BREAKFAST				
9:00 - 10:30 AM	Architecture and Scaling Ken Sipe	Project Integrity, Part 1: tools and techniques for managing source code quality David Bock	REST : Information-Driven Architectures for the 21st Century Brian Sletten	The Busy Java Developer's Guide to Collections Ted Neward	Hacking Your Brain for Fun and Profit Nathaniel Schutta
10:30 - 11:00 AM	MORNING BREAK				
11:00 - 12:30 PM	So you want to be an Architect Ken Sipe	Project Integrity, Part 2: tools and techniques for managing project complexity David Bock	Rich Web Pages : Publishing Semantic Content with GRDDL and RDFa Brian Sletten	The Busy Java Developer's Guide to Functional Java Ted Neward	Test Driven Design Neal Ford
12:30 - 1:15 PM	LUNCH				
1:15 - 2:15 PM	EXPERT PANEL DISCUSSION				
2:15 - 3:45 PM	Java Memory, Performance and the Garbage Collector Ken Sipe	Intermediate Maven David Bock	SPARQL: Querying the Data Web Brian Sletten	The Busy Java Developer's Guide to Advanced Collections Ted Neward	Communication Skills for Knowledge Workers Neal Ford
3:45 - 4:00 PM	BREAK				
4:00 - 5:30 PM	Debugging your Production JVM Ken Sipe	Easy mobile development (iPhone, Android, Palm Pre, Blackberry) without native code Pratik Patel	Semantic SOA : Meaningful Service Strategies Brian Sletten	The Busy Java Developer's Guide to Hacking with the JDK Ted Neward	Hands-on Agile Development Neal Ford

# Greater Atlanta Software Symposium

Atlanta Marriott Perimeter Center

October 23 - 25, 2009

<http://www.nofluffjuststuff.com/conference/atlanta/2009/10/home>

## **Estimating vs. Guessing - How Agile Teams Estimate Their Work by David Bock**

Estimating is regarded as little little more than 'educating guessing', but so much can hang on the quality of those estimates. With good estimates we can set clear expectations for project delivery, but with bad estimates we can run over schedule and over budget, or worse. We often estimate when we know the least about the work that needs to get done - so how can we make the best of what is potentially a bad situation?

## **Surviving Middle Management by David Bock**

Most good developers eventually have the opportunity to be managers. Whether they call you the "project manager", "Technical Lead", "Lead Developer", or some other classic middle-management title, you become the 'goto' guy between management and developers. You're the guy who is expected to keep the project in-line, track a schedule, and occasionally answer the question "How's it going?", and perhaps still contribute at a technical level. So how do you do that?

## **Project Integrity, Part 1: tools and techniques for managing source code quality by David Bock**

How many times have you started a new project only to find that several months into it, you have a big ball of code you have to plod through to try to get anything done? Have you ever been the 'new guy' on a project where it seems like the code grew more like weeds and brambles than a well-tended garden? With a few good tools to help analyze the code, we can keep our project from turning into that big ball of mud, and we can salvage a project that is already headed down that path.

## **Project Integrity, Part 2: tools and techniques for for managing project complexity by David Bock**

How many times have you started a new project only to find that several months into it, you have a build process that mysteriously fails, a bunch of 'TODO' and 'FIXME' comments in the source, and problems that come and go because "it works on my machine"? Does your project have a little bit of 'folk wisdom' that isn't well-known, but is necessary to get things done? How easily could you recreate your development environment if you got a new machine today?

## **Intermediate Maven by David Bock**

Maven is a build tool that does a lot, demos well, and leaves the build maintainers managing what seems like unbridled complexity. It doesn't have to be that way - Maven is driven by some strong 'build process methodology', and that complexity can become manageable by wrapping your head around it. Furthermore, you can migrate to Maven 'piecemeal', by mapping your existing ant build to the Maven Lifecycle and calling your existing Ant tasks - you can decide to sip the Maven kool-aid. Ideally, a build tool should be so simple and approachable that it fades into the project background and allows anyone to maintain it. Unfortunately, Maven's power comes at the expense of this ideal - Maven's philosophy is more like "the build process is so important that the people maintaining it should be steeped in the ways of Maven". This talk will give you the exposure you need without elevating The Maven Way to a religion.

## **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.

## **Groovy Testing by Scott Davis**

"Tests don't break things; they dispel the illusion that it works." (Anonymous) In this era of "Test-First" and "Test-Driven" development, the modern software engineer knows that testing is no longer an optional part of the process. You need to have the best tools at your fingertips: a set of utilities that maximize your results with a minimum of effort. Groovy offers Java developers an optimal set of testing tools.

## **RESTful Grails by Scott Davis**

"Any intelligent fool can make things bigger, more complex, and more violent. It takes a touch of genius - and a lot of courage - to move in the opposite direction." (Albert Einstein) REST and Resource-Oriented Architecture (ROA) are popping up in technical discussions more and more frequently. Here, you'll see practical examples of adding RESTful web services to your Grails application.

### **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?

### **Visualizations for Code Metrics by Neal Ford**

Judicious use of metrics improves the quality of your code. But interpreting metrics presents a challenge. You have a list of numbers for a project - what does it mean? And what does it tell me about the health of the project overall? This session shows how to produce visualizations for software metrics, making them easier to understand and more valuable. It covers metrics at the individual method level all the way up to the overall architecture of the application. This isn't just a talk about how some tools produce visualizations: this session shows you how to generate your own visualizations, allowing you to customize it to the level in information density that shows real value on your project. I show how to produce projected graphs from dependencies, heat-maps for cyclomatic complexity and code coverage, using XSLT to extract visual information from XML configuration documents, and others. Metrics can't help you if you can't understand them. By creating visualizations, it helps leverage metrics to make your code better.

### **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.

### **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.

### **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.

### **Communication Skills for Knowledge Workers by Neal Ford**

Software is fundamentally a communications game, and good skills differentiates between good and great developers. This session describes communication techniques and skills to people who skipped English

102 to hack some code. I talk about effective communication techniques for presentations, documentation, memos, and how to sell your technical ideas to a non-technical crowd.

### **Hands-on Agile Development by Neal Ford**

BRING YOUR LAPTOP WITH YOU, BUT A LAPTOP ISN'T REQUIRED! Reading and hearing about agile practices is one thing, but actually doing it is completely different. This session puts you to work in an agile fashion, applying agile developer practices.

### **JSF 2.0: An Introduction by David Geary**

This session introduces JSF 2.0 fundamentals, with emphasis on new features in JSF 2.0. **Prerequisite:** *Familiarity with JSF, or other component-based frameworks*

### **JSF 2.0: Advanced Topics by David Geary**

This session covers two of the most important features of JSF 2.0: composite components and built-in Ajax. **Prerequisite:** *Familiarity with JSF, or other component-based frameworks. Familiarity with Ajax. This session builds on the JSF 2.0 Introduction talk, so it is helpful, although not required, if you attend the intro talk before coming to this session.*

### **Flex for Java Developers by David Geary**

An introduction to Flex for Java developers. **Prerequisite:** *Familiarity with Flex and at least one other web application framework*

### **GWT fu, Part 1 by David Geary**

Learn to implement web applications with GWT. **Prerequisite:** *Familiarity with a component-based framework, preferably a desktop application framework*

### **GWT fu, Part 2 by David Geary**

Learn to do amazing stuff with GWT. **Prerequisite:** *GWT fu, Part 1 is not a prerequisite for this session, but it will help if you have some familiarity with GWT.*

### **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.

### **Taking Agile From Tactics to Strategy by Stuart Halloway**

Teams adopting agile should begin at a tactical level, but they shouldn't end there. The Agile Manifesto operates at many different levels. Learn to apply the principles of agile at a strategic level. Otherwise you can have a great agile ground game and still lose.

### **Agile, Relevance Style by Stuart Halloway**

The Agile Manifesto, like any good scripture, admits of many interpretations. There is no one "right path." What works for us may not work for you. At Relevance we have tried many paths, and learned many lessons. Join us to see dozens of ideas that have worked for us, plus some that haven't.

### **Java.next: Clojure, Groovy, JRuby, and Scala by Stuart Halloway**

In this talk, we will explore and compare four of the most interesting 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/2008/9/24/java-next-overview>.

### **Clojure by Stuart Halloway**

In recent years, the Java community has embraced a variety of new languages that target the JVM, but also offer productivity advantages over traditional Java coding.

### **Clojure by Ted Neward**

In recent years, the Java community has embraced a variety of new languages that target the JVM, but also offer productivity advantages over traditional Java coding.

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

If you've ever gotten a `ClassCastException` and just knew the runtime was wrong about it, or found yourself copying `.jar` files all over your production server just to get your code to run, then you probably find the Java `ClassLoader` mechanism to be deep, dark, mysterious, and incomprehensible. Take a deep breath, and relax--`ClassLoaders` aren't as bad as they seem at first, once you understand a few basic rules regarding their operation, and have a bit more tools in your belt to diagnose `ClassLoader` problems. And once you've got that, and hear about `ClassLoaders`' ability to run multiple versions of the same code at the same time, and to provide isolation barriers inside your application, or even compile code on the fly from source form, you might just find that you like `ClassLoaders` after all... maybe.

### **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 Functional Java by Ted Neward**

Much noise has been made in recent years about functional languages, like `Scala` or `Haskell`, and their benefits relative to object-oriented languages, most notably `Java`. Unfortunately, as wonderful as many of those benefits are, the fact remains that most `Java` developers will either not want or not be able to adopt those languages for writing day-to-day code. Which leaves us with a basic question: if I can't use these functional languages to write production code, is there any advantage to learning about them? The short answer is yes, for the fundamental premise--"I can't use functional code on my `Java` project"--is flawed. `Java` developers can, in fact, make use of functional ideas, and what's better, they don't even have to reinvent them for `Java`--thanks to the `FunctionalJava` library, many of the core primitives--interfaces that serve as base types for creating function values, for example--already exist, ready to be used.

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

Once you've learned the core `Collections` classes, you're done, right? You know everything there is to know about `Collections`, and you can "check that off" your list of `Java` packages you have to learn and know, right?  
**Prerequisite:** *Busy Java Developer's Guide to Collections*

### **The Busy Java Developer's Guide to Hacking with the JDK by Ted Neward**

Ever since its 1.1 release, the `Java Virtual Machine` steadily becomes a more and more "hackable" (configurable, pluggable, customizable, choose your own adjective here) platform for `Java` developers, yet few, if any, `Java` developers take advantage of it. Time to take the kid gloves off, crack open the platform, and see what's there. Time to play.

### **Virtualization for development by Pratik Patel**

We've all heard about virtualization for deploying applications. How about leveraging virtualization for development? In this session, we'll look at some time saving tips and build a virtual VM for development and testing.

### **Easy mobile development (iPhone, Android, Palm Pre, Blackberry) without native code by Pratik Patel**

So you have a great idea for an `iPhone` app, you've tried learning `Objective-C`, but it's just too hard. What about those other new platforms like `Palm Pre` and `Android`? Who wants to write the same app three times? Four times if you count `Blackberry`! Fear not, there is a much easier way for you to develop on the `iPhone`. Using a development style called "hybrid mobile applications" you can write apps for `iPhone` and other platforms using stuff you already know: `HTML`, `CSS` and `Javascript`. In this course, we'll go over the basics for hybrid development

### **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

### **JavaScript: the Good, the Bad, and the Ugly by Nathaniel Schutta**

Thanks to Ajax, JavaScript is cool again and developers are taking a second look at this much maligned language.

### **Making Web Apps Suck Less by Nathaniel Schutta**

We've all used web applications that had us screaming at their creators - unfortunately sometimes we're the ones being cursed. Believe it or not, there are some simple steps we can take to ensure that our users have a great experience.

### **Hacking Your Brain for Fun and Profit by Nathaniel Schutta**

The single most important tool in any developers toolbox isn't a fancy IDE or some spiffy new language - it's our brain. Despite ever faster processors with multiple cores and expanding amounts of RAM, we haven't yet created a computer to rival the ultra lightweight one we carry around in our skulls - in this session we'll learn how to make the most of it. We'll talk about why multitasking is a myth, the difference between the left and the right side of your brain, the importance of flow and why exercise is good for more than just your waist line.

### **What's New in Spring 3 by Ken Sipe**

The Spring Framework has led the industry in innovation for years. Starting with dependency injection and promoting testing through removal of framework dependencies. Spring 3.0 continues that innovation in a way that takes full advantage of the Java 5 platform. There are a number of significant changes to the framework. So whether you are new to the framework or an experience Spring developer, this is a great session to come up to speed on the latest from SpringSource. **Prerequisite:** Java 5

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

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

### **So you want to be an Architect by Ken Sipe**

This session is a quick look at all aspects of being a corporate software architect. Whether you are a developer looking to move into the role of architect, needing to have an understanding of what is expected or already in the role of software architect looking for new and interesting ideas, this session is for you.

### **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?

### **Debugging your Production JVM by Ken Sipe**

So your server is having issues? memory? Connections? Limited response? Is the first solution to bounce the server? Perhaps change some VM flags or add some logging? In today's Java 6 world, with its superior runtime monitoring and management capabilities the reasons to the bounce the server have been greatly reduced.

### **REST : Information-Driven Architectures for the 21st Century by Brian Sletten**

There is a shift going on in the Enterprise. While still used and useful, the promises of the SOAP/WSDL/UDDI Service-Oriented Architecture (SOA) stack have failed to live up to their promise. A new vision of linked information is enveloping online and Enterprise users. The REST architectural style is squarely behind this thinking as a way of achieving low-cost, flexible integration, increased data security, greater scalability and long-term migration strategies. If you have dismissed REST as a toy or are unfamiliar with it, you owe it to yourself to see what is so interesting about this way of doing things.

### **Rich Web Pages : Publishing Semantic Content with GRDDL and RDFa by Brian Sletten**

The human web is reasonably well in hand by now. We are getting pretty good at building systems that people find valuable and entertaining. We have not spent as much time concerned about our software friends. There is a ton a rich content available on the web that is too difficult to extract in automated ways using just XHTML, the meta tag and microformats. This talk will introduce you to some emerging technologies from the Semantic Web camp to enrich your web pages with useful information for both

automated extraction and improved browsing experiences. **Prerequisite:** *The Semantic Web: The Future Now would be helpful, but not required*

### **SPARQL: Querying the Data Web by Brian Sletten**

The human-friendly Web is about nicely-formatted, accessible content for users to browse. There is an emerging Data Web that relies on technologies from the Semantic Web stack to link increasingly rich connections between various data sources. SPARQL and RDF are the main tools for expressing and using this connectivity. This talk will introduce you to one of the practical and accessible aspects of employing these ideas on the Web and in the Enterprise. **Prerequisite:** *The Semantic Web: The Future, Now and Rich Web Pages : Publishing Semantic Content with GRDDL and RDFa would both be helpful but are not required*

### **Semantic SOA : Meaningful Service Strategies by Brian Sletten**

The goal for web services was always to reduce our burden by increasing the potential for reuse of business functionality. Somehow, we got lost along the way in a morass of confusing, unfulfilling and downright broken technologies. While we are interested in pursuing REST-based systems for managing information, we need some strategies for tying it all together sensibly. If we abandon WSDL, SOAP and UDDI, what do we replace them with? This talk will walk you through combining resource-oriented strategies with technologies from the Semantic Web to describe, find, and bind to services in dynamic, flexible and extensible ways.

**Prerequisite:** *The Semantic Web: The Future Now, Give it a REST and SPARQL : Querying the Data Web would all be helpful talks to have attended*

### **Effective Java by Venkat Subramaniam**

Java is a well established language, that has been around for more than a decade. Yet, programming on it has its challenges. There are concepts and features that are tricky. When you run into those, the compiler is not there to help you.

### **Programming Scala by Venkat Subramaniam**

Scala is a static fully object-oriented, functional language on the JVM. While taking advantage of the functional aspects, you can continue to make full use of the powerful JVM and Java libraries.

### **Cleaning up Code Smell by Venkat Subramaniam**

Projects often start out simple, but soon become complex and turn into a lose cannon. Organizations are struggling to maintain and evolve software. Poor code quality is a significant part of that problem. Improving the quality of code is critical to success of enterprise projects.

### **Tackling Concurrency on the JVM by Venkat Subramaniam**

In this presentation we will take a quick walk though the issues with concurrency and how the solutions provided in Scala and Clojure help address those.

### **Building External DSLs by Venkat Subramaniam**

Domain Specific Languages (DSLs) are languages targeted at a particular problem and domain. They have context and are fluent. They help users of applications at various levels to easily communicate with your application. Developing DSLs, however, are not easy. You could easily get dragged into using parsers and tools with steep learning curve.