Extreme Java Productivity – Enterprise Applications in Just Minutes

Ben Alex, Senior Staff Engineer, SpringSource Division, VMware

Agenda

- Introducing Spring Roo
- IDE Support and Conventions
- Advanced Features
- Roadmap and Resources

Introducing Spring Roo

Mission Statement

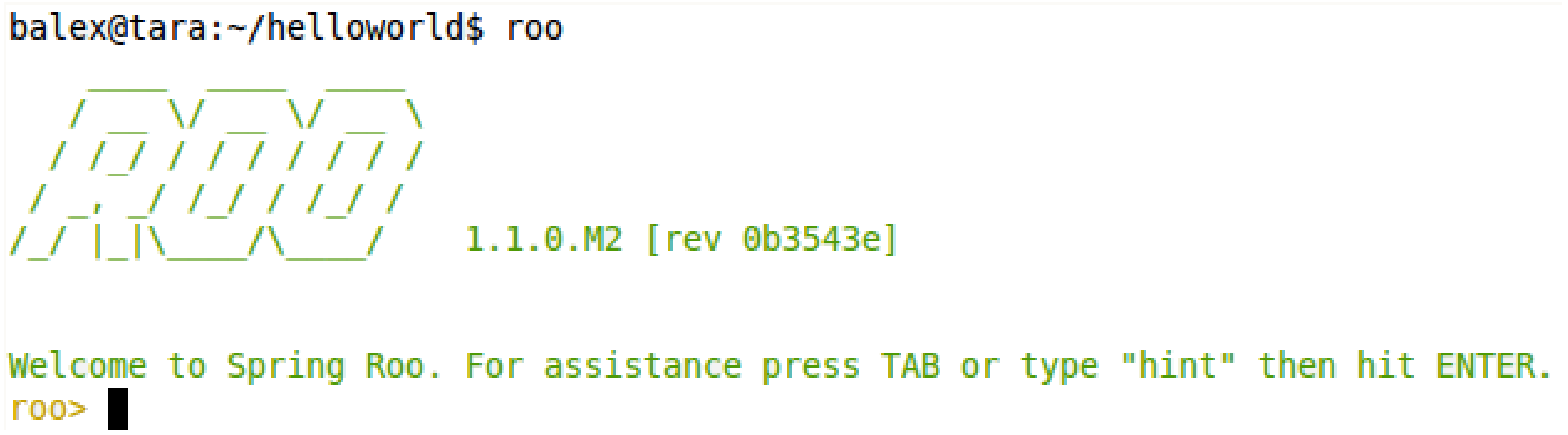
“Roo's mission is to fundamentally and sustainably improve Java® developer productivity without compromising engineering integrity or flexibility

End User's Description

“Roo is a little genie who sits in the background and handles the things I don't want to worry about

What Is Spring Roo?

- Easy-to-use, extensible, text-based RAD tool for Java® developers
- Development-time only (no runtime, no lock-in)



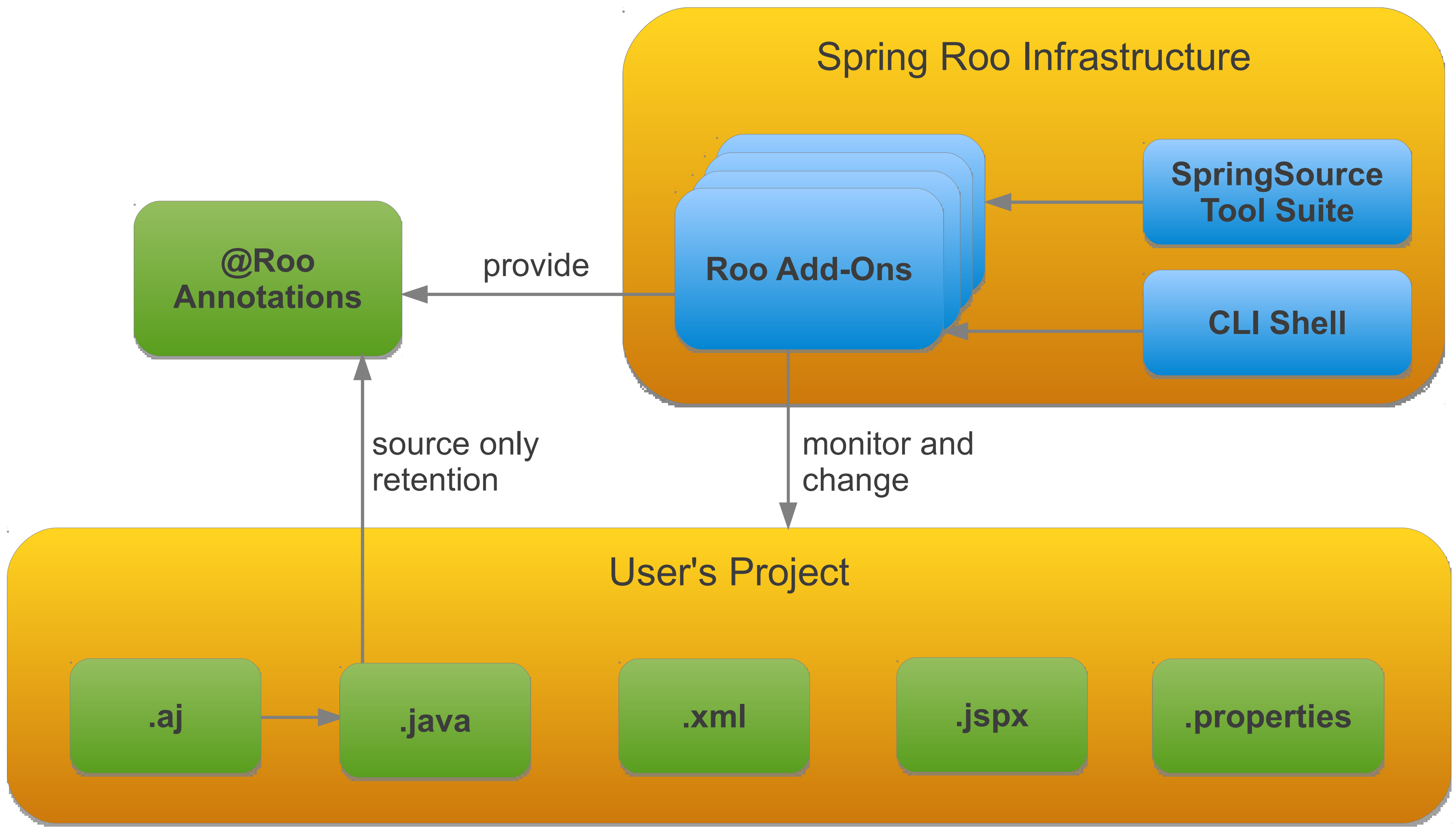
```
balex@tara:~/helloworld$ roo
    ____  ____  ____
   / __ \/ __ \/ __ \
  / /_/ / / / / / / /
 / _, _/ /_/ / /_/ /
/_/ |_|\____/\____/    1.1.0.M2 [rev 0b3543e]


Welcome to Spring Roo. For assistance press TAB or type "hint" then hit ENTER.
roo>
```

Rapid Delivery

- **Java® Platform**
  - Java 5+
  - Java Bean Validation
  - Java Database Connectivity
  - Java Message Service
  - Java Transaction API
  - Java Server Pages
  - Java Persistence API
- **Java® Servlet Technologies**
  - Jetty
  - Apache Tomcat
  - Apache Tiles
  - Apache Solr
  - Spring MVC
  - Spring Web Flow
  - Web Application Resource (WARs)
- **JPA Implementations**
  - Hibernate
  - Apache OpenJPA
  - EclipseLink
  - Google App Engine (Data Nucleus)
- **Popular Open Source Libraries**
  - Apache Maven, Google Web Toolkit, Adobe Flex, Dojo Toolkit, Apache ActiveMQ, Log4J, Eclipse, Selenium, JUnit, AspectJ, Spring Platform...

Implementation Overview



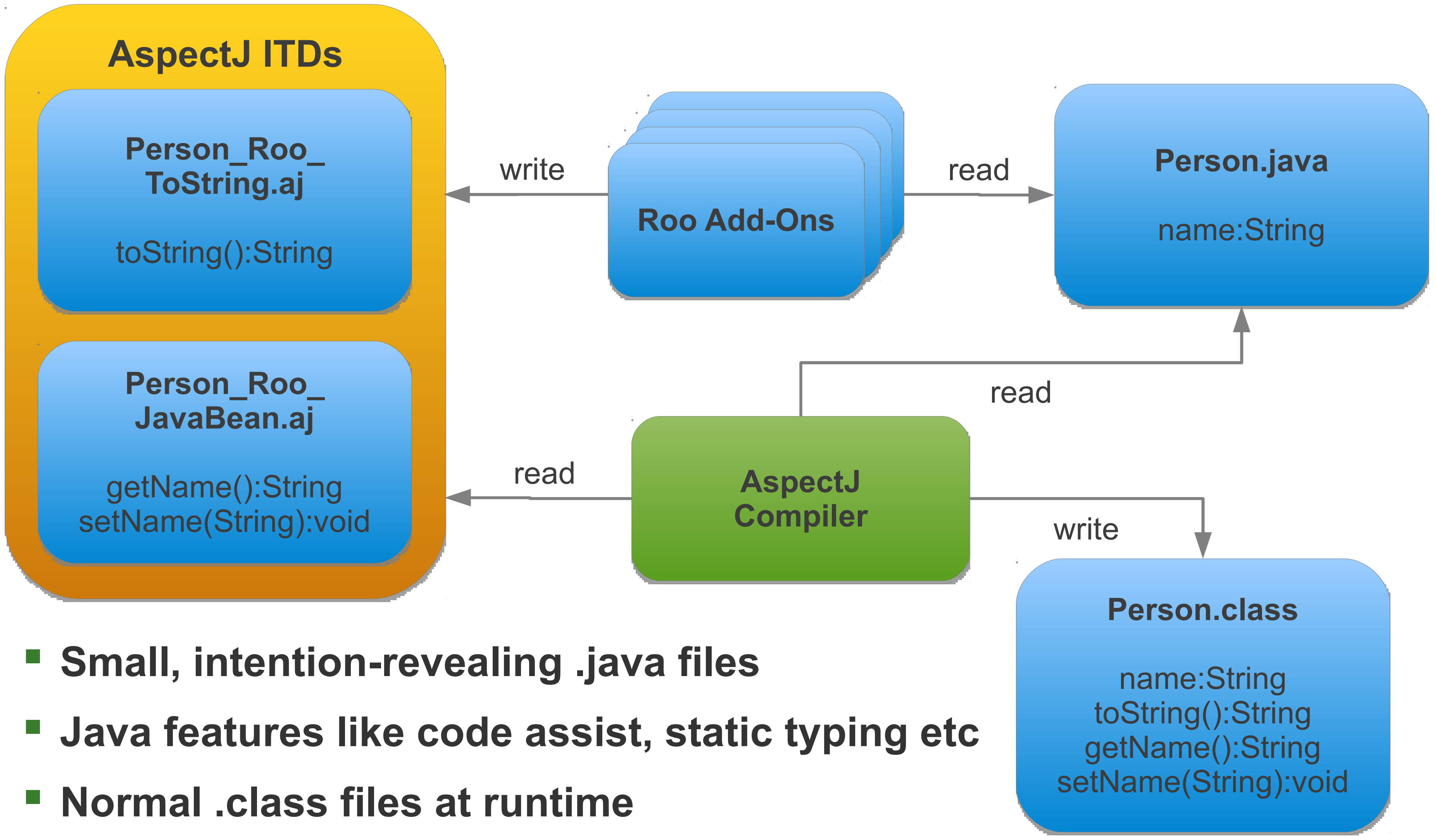
Code Generation

- **Roo is a “hybrid” code generator**
  - It selectively merges the best of the passive and active generation models

- **Passive generation**
  - Passive generation is a one-time generation (eg IDE “generate getters/setters”)
  - Roo performs passive generation in response to your shell commands
  - Passively-generated files are very small and are easily edited in your IDE

- **Active generation**
  - Active generation automatically updates certain files as you work on a project
  - We've probably all used at least one badly-written active code generator
    - Special build scripts, unnatural type models, lock-in, weird templates, crude output etc
  - Roo overcomes these problems and makes active generation elegant...

Elegant Active Generation



- **Small, intention-revealing .java files**
- **Java features like code assist, static typing etc**
- **Normal .class files at runtime**

Demo

Building a Web Application

Conventions and IDE Support

Getting Started

- **Minimum Requirements**
  - Java 5+
  - Maven 2.2+
  - Tested with Windows (including Cygwin), OSX and Linux

- **Recommendations**
  - IDE: SpringSource Tool Suite
  - OSX Users: iTerm (better ANSI support)

- **Installation**
  - Download www.springsource.org/roo, unzip and add to your path
  - Spring Roo is also pre-installed in SpringSource Tool Suite

IDE Interoperability

- **You can use Spring Roo without any IDE**
  - Roo directly parses .java source files (no compiler step is needed)
- **For the best IDE experience, install your IDE's AspectJ plugin**
  - Eclipse users can add the Eclipse AspectJ Development Tools (AJDT) plugin
  - IntelliJ users have AspectJ support available (see ticket IDEA-26959)
- **Load Roo in a separate window while using your IDE**
  - This allows Roo to discover file changes
  - If you forget to load Roo, it will automatically “catch up” when you next load it
- **SpringSource Tool Suite has extra Roo-specific features**
  - Such as embedded Roo, so STS users don't need not load Roo separately

User Interface Conventions

- **Usability tips**
  - Press TAB to complete
  - TAB also displays option help (eg --foo)
  - Failures automatically rollback changed files
  - Commands never prompt you for further information once invoked
- **Useful commands**
  - “hint” for step-by-step advice
  - “help” for detailed information about any command
- **There is a “flash notification area” in the top-right corner of the shell**
  - Long running operations
  - Low-level diagnostic information if activated

File Conventions in Spring Roo 1.1.0

- **By default you are responsible for all files in your project**
  - You can use a text editor or IDE to change any file at any time
- **Automatically managed files**
  - *.jspx files: edited automatically (your changes are automatically preserved)
  - *_Roo_*.aj files: edited automatically (do not edit these files yourself)
  - *Record.java files: edited automatically (do not edit these files yourself)
- **You shouldn't need to edit the AJ and Record files**
  - Use “push in refactor” (or copy and paste) to move content to .java
  - Record files are used by the GWT add-on to represent your member structure for GWT and as such do not contain any behavior or content you'd need to edit

Project Defaults

- **Maven 2**
  - Standard Maven directory structure (src/main/java etc)
  - Automatically adds correct plugins for AspectJ weaving etc
  - Projects start as a “jar” type, but become “war” once you add a web tier
  - Compatible with m2eclipse
  - Multi-project support will be added to Spring Roo 1.2 (see ROO-120)

- **Project Footprint**
  - AspectJ and Spring are the only defaults (used for AOP and IoC respectively)
  - Everything else is optional and added only when you ask
    - You decide which JPA provider (if any) you'd like to use
    - You decide which web tiers (if any) you'd like to use (Spring MVC, GWT, Flex etc)
  - Even a “full” enterprise web app WAR is ~13 Mb (quite small by 2010 standards)

Demo

Exploring IDE Support and Conventions

Advanced Features

What Is Database Reverse Engineering?

- **Producing a Java tier from an existing relational schema**
- **Very commonly performed**
- **Eclipse has a “JPA entities from tables” wizard**
  - Generates entities from a JDBC connection
  - Can be tailored to change generated type and field names
  - Does not handle tables with no primary keys
- **JPA implementations also offer this feature**

Limitations of Existing DBRE Tools

- **Complex and long-winded wizard style interactions**
  - Is that a many-to-one, which side is the owner, which inheritance strategy...?
- **May produce files with JPA implementation-specific annotations**
  - Locking you into that JPA provider
- **Java files become cluttered with noisy JPA declarations**
  - These auto-generated and thus inferable declarations belong elsewhere
- **No incremental updates**
  - Application requires manual adjustment if the schema changes
  - Or worse still, deleting the entities and starting again

Roo's Incremental Database Reverse Engineering

- **Most requested feature in history of Roo**
- **Quality reverse engineering**
  - Places declarations in ITDs, keeping your Java files clutter-free
  - 100% JPA 2 annotations (no JPA implementation-specific annotations)
  - Fine with large schemas (400+ tables), handles complex PKs/FKs etc
- **Easy to use**
  - Just one command does it, and there are zero questions to answer
  - Add a Spring MVC web tier for the new entities in just one more command
- **Incrementally updates your domain model as schema evolves**
  - At last, Java type safety based on an evolving database schema

DBRE Commands

- **database introspect --schema <name> [--file <name>]**
  - This command is optional – it's mostly for testing the connection
  - Displays database metadata in XML format in the Roo shell
  - Optional --file <file name> saves metadata to specified file
  - Provides a preview of the mappings used in the final model
- **database reverse engineer [--schema <name>] [--package <name>]**
  - This is the main command
  - Creates entities in the specified package
  - --schema and --package options required only for first time run of command
  - Automatically generates type and field names from table and column names
  - In Roo 1.1.1 there is now an --excludeTables option with wildcard support

Web Tier Support

- **Spring MVC**
  - Mature and popular add-on
  - Full .jspx round-tripping, REST (with JSON), JavaScript tag library and more
  - Use “web mvc embed” if you'd like social media content from 16 sites including YouTube, Vimeo, Screenr, Flikr, Picasa, SlideShare, Google Maps, Twitter etc
- **GWT**
  - Extensive and ongoing collaboration with the Google GWT team
  - Uses new features in GWT 2.1 including RequestFactory for optimised remoting
- **Adobe Flex**
  - Available as a separate Roo add-on, with full ActionScript and Java services
- **Community projects building add-ons for Vaadin, Wicket and JSF**

Add-On Infrastructure

- **Spring Roo is built on OSGi to enable anyone to write new features**
- **OBR allows Roo to automatically discover and install new add-ons**
  - Try this: type “welcome” into a Roo shell and notice it suggests an add-on
  - Every URL in the OBR index is published with the httppgp:// scheme
  - Our RooBot tool maintains a central OBR index of all Roo add-ons
- **PGP is used to deliver a decentralised trust model**
  - A httppgp:// URL will only download if a trusted key signed the resource
  - Use “ppg list trusted keys” and “pgp status” to view your trust database
  - Use “pgp trust” and “pgp untrust” to manage which keys you trust
- **Summary: automatic add-on discovery with a robust trust model**

Internal Geeky Stuff...

- **“development mode”**
  - Provides full exception traces
- **“poll status”**
  - Prints file monitoring statistics
- **“metadata status”**
  - Indicates metadata statistics
- **“metadata trace”**
  - Lots of low-level notifications
- **“osgi scr component list”**
  - Dig into the active OSGi components
- **“process manager debug”**
  - Flashes system status messages
- **“system properties”**
  - As provided by the JVM
- **“help”**
  - Discover plenty of other goodies

Removing Spring Roo

- **It's easy to remove Spring Roo from your project**
  - Roo has no runtime portion to worry about

- **Five minutes and it's gone**
  - Step 1: Use AJDT's “Push In Refactor” feature (relocates content from .aj files)
  - Step 2: Remove the Roo annotation JAR entry from your pom.xml
  - Step 3: Remove all the @Roo annotations (use a global find and replace)

- **But you can change your mind again...**
  - You can still run Roo on your project again later and re-add the annotation JAR

Demo

Database Introspection

Roadmap and Resources

Roo 1.1.0 Release

- **Current release is Spring Roo 1.1.0.RELEASE**
- **Planned upcoming releases:**
  - 1.1.1 planned for 17 December 2010
  - 1.1.x series in Q1 2011
  - 1.2 milestones from March/April 2011
- **GWT 2.1 releases have similar timing to maximize compatibility**
- **Roo 1.1 uses Spring 3.0.x GA**

Community Resources

- **Home → http://www.springsource.org/roo**
  - Contains links to all other resources
- **Forum → http://forum.springsource.org**
  - Roo team actively monitor forum and answer queries
- **Issues → http://jira.springframework.org/browse/ROO**
- **Twitter → @SpringRoo**
  - Follow for updates, or include in tweets so we see them

Conclusion

- **Spring Roo delivers serious productivity gains to Java developers**

- **Highlights**
  - Popular, proven Java technologies you already know
  - Easy to learn, easy to use, easy to extend
  - Builds on Java's strengths
  - Extreme performance
  - No runtime, no lock-in, no risk
  - Active, open source project and community

- **Contact details: balex@vmware.com and @benalexau**