
Rod Johnson | SpringSource

# The Future of Enterprise Java

Keynote, JAX (Wiesbaden), April 22, 2008

# DOES ENTERPRISE JAVA *HAVE* A FUTURE?

# Does enterprise Java have a future?

*Ruby on Rails killed off enterprise Java*

*No one still builds web applications using Java today*

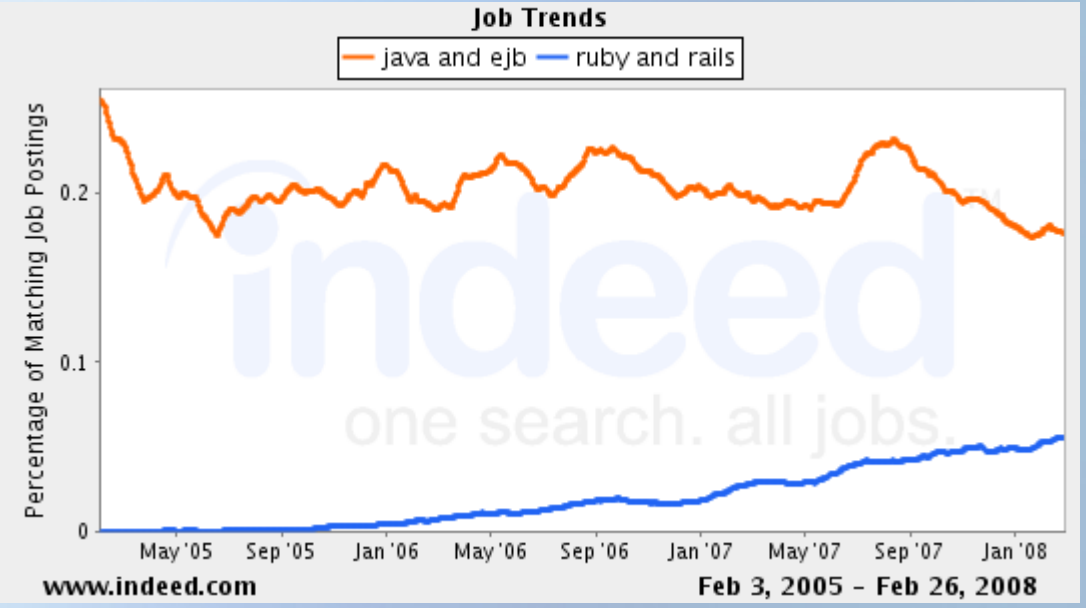
*No one will date a Java programmer*

# The facts don't bear out the hype

- The Java market is not shrinking
- It is *many times* larger than the Ruby market

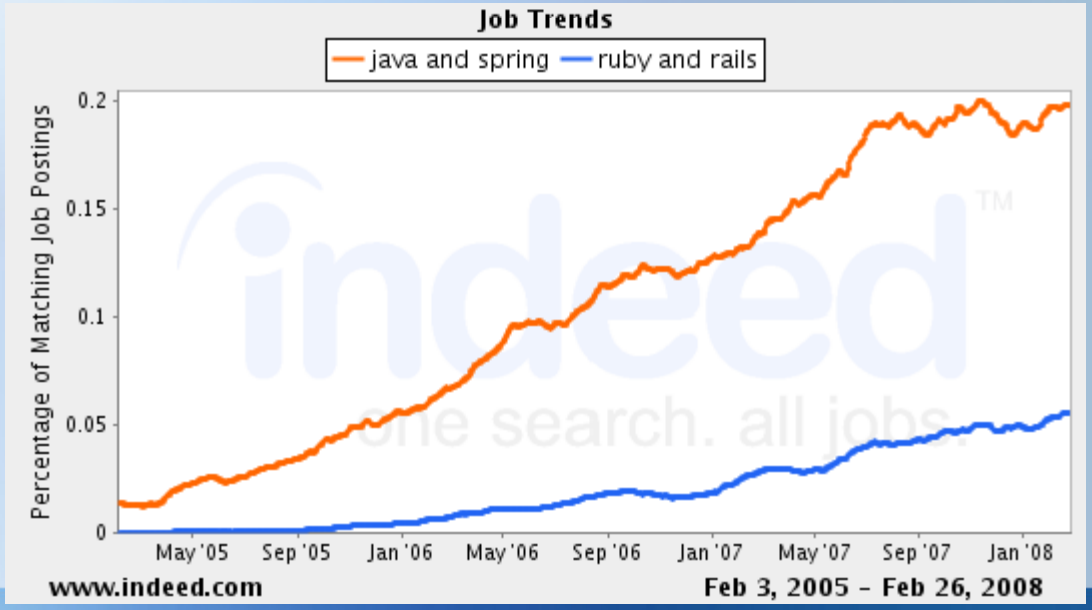
# It's what you compare…

- Legacy Java technology vs Ruby on Rails looks depressing
- *Measures what Java vendors are promoting*



# …It's what you compare

- Growth Java technology vs Ruby on Rails looks positive
- *Measures what people are actually doing*



# But let's not get complacent…

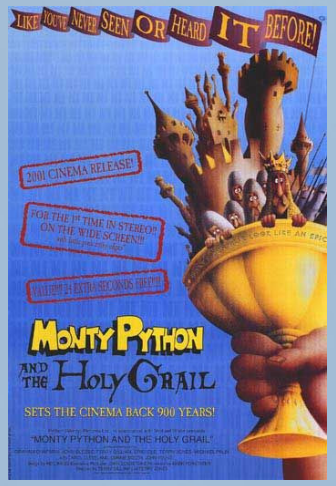
- Enterprise Java needs to change
- There are strong competitive threats, and it has not fully delivered on its promise

# Topics

- Forces for change
- Current trends
  - Introduction to Java EE 6
  - The market scenario
  - Technology forces
- Six Predictions for the future
  - Two totally free bonus predictions

# The *Real* Topics…

- The Cold War
- Monica Lewinsky
- Monty Python

# Why does enterprise Java need to change?

- Productivity challenge
- Need for modularity to reduce bloat and improve maintainability
- New requirements
  - SOA
  - Web 2.0

# *Trends in Platform Middleware: Disruption is in Sight*

*The popular Java Platform, Enterprise Edition (Java EE) and .NET platform middleware technologies are increasingly inadequate to cover needs for extensive scalability and performance, event-based programming styles, advanced service-oriented architecture (SOA) and dynamic application developments.*

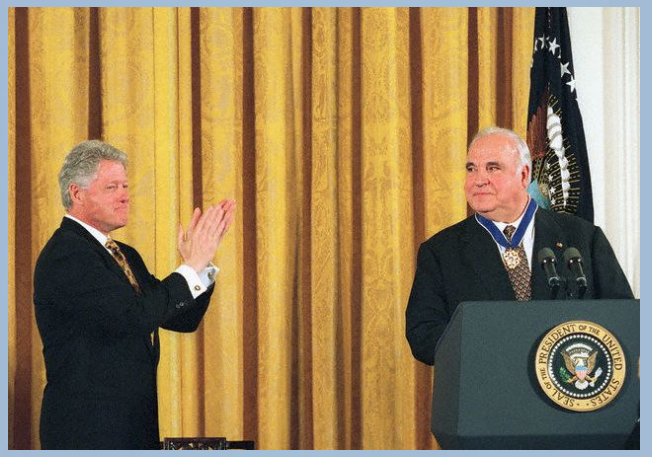
- Gartner Group, September 2007

# The Productivity Challenge

- Ruby on Rails challenge
  - Demonstrates common sense productivity solutions such as practical code generation
- No great leaps forward in enterprise Java productivity since Spring/Hibernate
  - Only incremental improvements
- Java community tends to miss low hanging fruit
  - This needs to change

# Baggage slows enterprise Java down

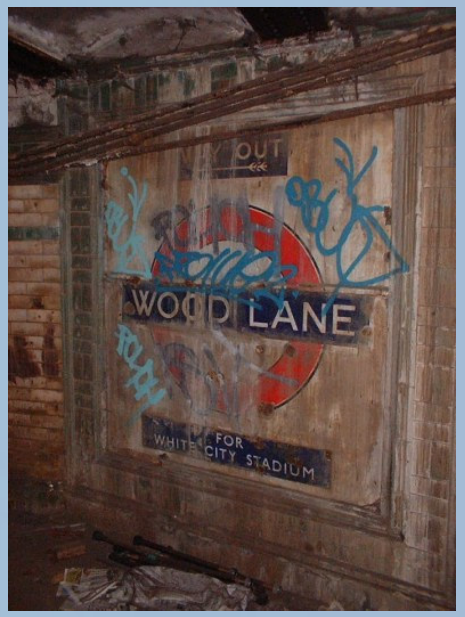
- When J2EE was conceived, Bill Clinton was President and Monica Lewinsky was in the news
  - There may be another President Clinton, but technology has moved on



# Baggage


Collection of London Transport Museum

- Systems accumulate baggage over time
- Need a regular clean out

# Java EE 6: An attempt to clean house

- The first radical rethink of the platform
- Set to be final late 2008 or early 2009
- Starts cleaning up the baggage

# The Two Philosophies Behind EE6

- *Extensibility*
- *Profiles*

- Read the proposal (JSR-316) for information from the source

# Extensibility

- *It would not be appropriate for the Java EE platform to grow without bound to include all the interesting and useful technologies desired by web and enterprise application developers. Instead, we believe it is desirable to enable more of these technologies to cleanly layer on or plug in to Java EE application servers. By adding more extensibility points and more service provider interfaces, these other technologies can plug in to platform implementations cleanly and efficiently, and* ***be just as easy to use for developers as the facilities that are built into the platform***
  - **Java EE 6 JSR**

- Summary: Java EE 6 aims to make it easy and more natural for frameworks to build on the platform, recognizing that this is a key strength of the Java ecosystem

# Profiles

- Different sets of platform technology for different purposes
- Finer-grained TCKs to allow for compatibility testing of subsets
- Three profiles

# Profile A – Minimal, Lightweight – Equivalent to Tomcat

- Servlet 3.0
- JSP 2.2
- JSR-45
  - Debugging API
- EL 1.2
- JSTL 1.2
- JSR-250
  - Common Annotations for Java Platform

# Key element of Profile A: Servlet 3.0

- Biggest change relates to extensibility
- API allowing programmatic registration of resources at runtime without everything being tied to web.xml
- Intended to make it easier for frameworks to integrate “natively” with Java EE platform

# Profile B – Adds persistence and two(?) component models

## From Profile A

- Servlet 3.0
- JSP 2.2
- JSR-45
- EL 1.2
- JSTL 1.2
- JSR-250

## Additional technologies

- EJB 3.1 (Lite)
- JTA 1.1
- JPA 2.0
- JSF 2.0
- Web Beans 1.0?

# Profile B

- Still relatively lightweight
- Numerous question marks
  - EJB 3.1 “Lite” still not clearly defined
    - Subset likely to be restricted to local session beans and EJB 3.0 “simplified programming model” without backward compatibility
    - Essentially @Resource and @Interceptors
  - Inclusion of Web Beans (JSR-299) still unclear
    - Unclear how this *new* component model relates to EJB

# Profile C – “Full Platform”

## The whole enchilada

- What Java EE used to be
- Sorry, I can’t make the font a readable size, there’s too much baggage

## Specifications

- Servlet 3.0
  JSP 2.2
  JSR-45
  EL 1.2
  JSTL 1.2
  JSR-250
  JTA 1.1
  JSF 2.0 *
  Web Beans 1.0 *
- EJB 3.1
  JPA 2.0
  JSF 2.0
- JAX-RS 1.0
  Connectors 1.6
  JAX-WS 2.2
  JAXB 2.2
  JSR-109 1.2
  JSR-181 1.1
  JMS 1.1
  JAF 1.1
  JavaMail 1.4
  JSR-115
  JSR-196
  JSR-88 1.2
  JSR-77 1.1
  JAX-RPC1.1
  JAXR 1.0

# Profile C

- "Old J2EE"
- In Cold War terms, the Titan nuclear missile of enterprise Java
- Less and less relevant to today's requirements
  - Prevents agility in implementations
  - Contributes to bloat affecting development and operations
- Modularity of the profiles A and B more relevant to today's problems

# Java EE 6: Overall Significance

- Recognizes that the world has changed
- Standardizes the kinds of real-world infrastructure that people use, while giving a brand guarantee
- **Breaks up the cosy franchises of licensees who have relied on their ability to implement irrelevant APIs as barriers to entry for competitors**

# Opposing forces: Modularity and Monopoly

- Java EE 6 promotes modularity
  - Opens up competition
- Old application server market consolidates more and more, reducing competition
  - When Bill Clinton was president, there were as many application server vendors as interns in the Oval Office
  - Now just a handful
  - Oracle's acquisition of BEA takes this to a new level

# Two clear market leaders, with a clear third place

- WebLogic and WebSphere together account for 70% of the market
- JBoss maintains a strong market position but has lost momentum since Red Hat acquisition
- Glassfish (Sun) is a promising product but is a dark horse, far behind market leaders

# The two market leaders are part of much bigger plays

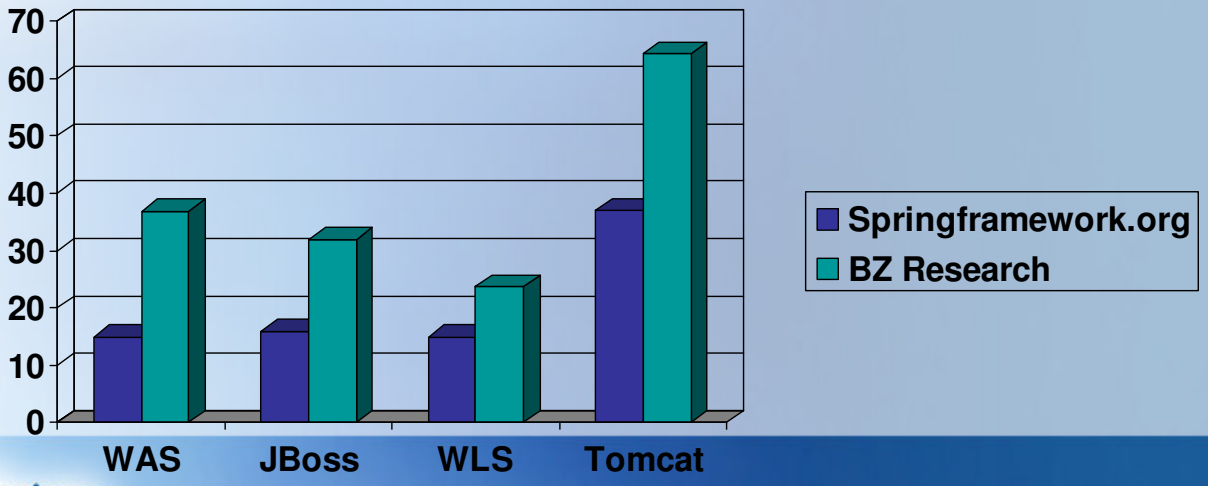
- Oracle has an aggressive vertical play
  - Applications
  - Middleware
  - Database
  - Even attempting to get into the OS business
- IBM has a massive services business and a vast software portfolio

# …The two market leaders are part of much bigger plays

- BEA was an independent middleware vendor, with Java middleware its key product
  - BEA needed to win on “best of breed”
  - Could only survive as a company if WebLogic was oustandingly good
- Oracle and IBM view their Java middleware as part of a Microsoft-like full-stack solution
  - Neither needs it to be outstandingly good to meet their goals

# But how do we measure the market?

- Fortunately, the old measurements of the market are obsolete
- Tomcat has overtaken the supposed market leaders and continues to pull ahead *in production usage* as well as in development
- More evidence of the gap between the official world and reality that is bound to close



# PREDICTIONS

# Six predictions

1. Real competition will return to the application server market
2. Tomorrow's application server will be lightweight and modular
3. Tomorrow's application server will not merely implement JCP specifications
4. The market will address the gap between Tomcat and WebLogic/WebSphere
5. The gap between application servers and ESBs will be bridged
6. EJB will die

# Prediction 1: Real competition will return to the application server market

- One of two things will happen
  - Java EE will cease to relevant
    - In which case the existing franchises will be less relevant, *or*
  - (More likely) Java EE 6 will rejuvenate Java EE
    - Most interest will be in Profiles (A) and (B)
    - The existing “whole enchilada” franchise will be less important

# Prediction 1a

- Economic value will become more aligned with what people actually use
- Example
  - While WebSphere use is static, the WebSphere franchise returns IBM massive and growing revenue
  - A huge proportion of the IBM user base uses only a tiny part of the product and actively *does not want* the size and complexity
- Market forces will drive realignment

# Prediction 2: Tomorrow's application server will be lightweight and modular

- OSGi provides a technical basis
- With Java EE 6 even Java EE is becoming modular
- The tend toward "self-assemble" of parts such as Spring, Tomcat Hibernate and other open source projects shows the success of this approach in practice
- Natural trend towards getting the stack supported out of the box rather than building yourself

## Prediction 3: Tomorrow's application server will not merely implement JCP specifications

- New sources of relevant specifications and de facto standards
- The diversity of sources of innovation is a key strength of the Java platform

# Enterprise Java is no longer a one party state

- Not just the Party (the JCP)
- OASIS
  - SCA
  - Web Services standards
- OSGi Alliance
  - Dynamic modularization standards
  - More enterprise standards
- Open source projects
  - Eclipse Foundation is flexing its muscle on the server side

# The JCP must look at wider world and accept that it doesn't need to reinvent everything

| JCP technology | Ignored existing technology | Negative consequences |
|---|---|---|
| Entity beans | TopLink and all other ORM solutions | •Two complete failures (EJB 1.x and 2.x)<br>•ORM in Java loses at least 6 years<br>•Billions of dollars of wasted development effort from customers |
| java.util.logging | Log4J | Added complexity of pointless abstraction layers such as Commons Logging |
| EJB (DI) | Spring, PicoContainer, Hivemind | Limited DI functionality in EJB 3 specification misses opportunity to match best practice |
| EJB3 (interception) | Spring, AOP Alliance, AspectJ, AspectWerkz | Lack of knowledge of AOP in the expert group produces fragile, clunky API missing central AOP concepts |
| JSR 277 (modularization) | OSGi | •Ignoring input and experience from OSGi<br>•May split JCP as many organizations are deeply committed to OSGi |

# Signs of progress…

- Recent signs suggest that the JCP is becoming more open
  - Likelihood of an accommodation of JSR 277 and JSR 294 and OSGi
  - JCP Chair Patrick Curran is committed to listening to the community and increased openness

# Prediction 3a

- Within 18 months, the JCP will change to be run through open source
- Sun is becoming an open source company
  - MySQL acquisition spells the end of the Soviet era of the JCP

# Prediction 4: The market will address the gap between Tomcat and WebLogic/WebSphere

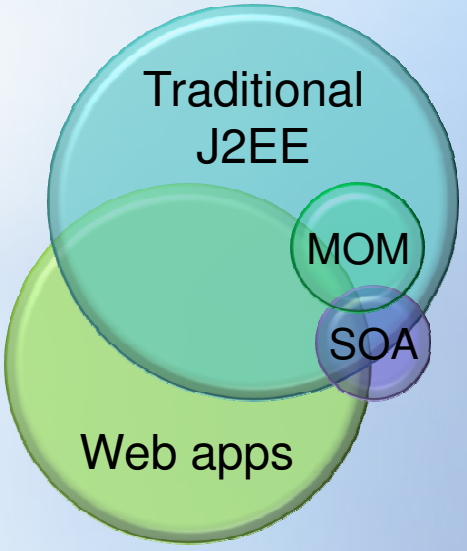
- Currently the product division is an API division
  - A full-blown EE server with many APIs, many of which are not relevant, plus open source libraries
  - A servlet engine plus the same libraries
- There is also a division on QoS and operations
  - WebLogic and WebSphere still lead here
  - Useful features for data center rollout
- There is a demand for a manageable, scalable product without the API baggage
  - So far, has not been a focus of open source projects

## Prediction 5: The gap between application servers and ESBs will be bridged

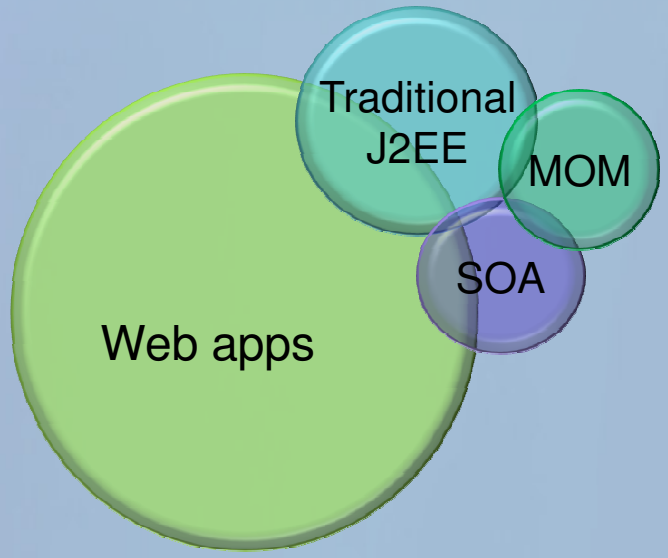
- Old stovepipe architectural model is no longer relevant
- SOA is spreading across enterprises
- The same modular infrastructure should be able to support different architectural scenarios
  - BEA's mSA initiative (now probably on hold) was an interesting recognition of this

# Today's requirements drive need for consistent infrastructure solution

- Monolithic J2EE once server seen as runtime for everything



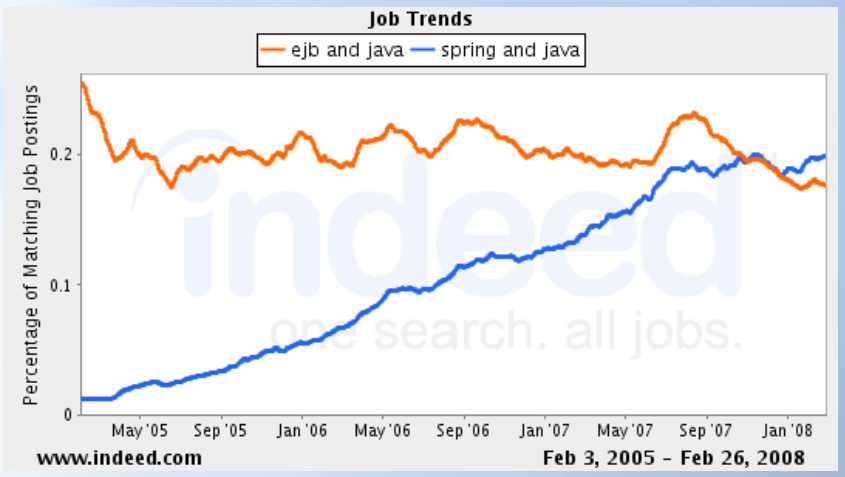

- Reality is more like this
- Distinct requirements need distinct infrastructure
  - *But it should be based on common underpinnings*



# Prediction 6: The Black Knight will be defeated

- It isn't just a flesh wound

# EJB is dying





- Partly a reflection of Prediction 5…
  - EJB was tied to a particular architectural scenarios
- Technical arguments against EJB have long been clear
- Evidence in the market is now also clear
- EJB is the Cobol of enterprise Java
  - Declining in parallel

# Why does it matter?

- EJB is the centerpiece of “old J2EE”
- It’s vital that it is put aside for progress to be made
- No amount of reinvention is enough

# Conclusion

- We’re in for a period of rapid change
- Java EE 6 may keep Java EE relevant, but Java EE no longer shapes the future
- One of the key technologies that *will* shape the future is OSGi
  - Try to catch OSGi sessions at JAX
- It will be an exciting time