SpringSource tc Server

Read this paper if you answer yes to the following questions:

Are you unsatisfied with your ability to rapidly deliver the software solutions your business requires?

Does your current enterprise Java application server suffer from excessive cost and complexity?

Are you interested in understanding the hardware, software, and productivity-related savings associated with lean application infrastructure versus traditional enterprise Java application servers?

Is your development team familiar with lightweight containers such as Spring and Apache Tomcat?

Are you interested in understanding how a lightweight application server like Apache Tomcat can streamline your web application lifecycle while providing the enterprise-class capabilities you need?

Prepared by: Shaun Connolly, Vice President of Product Management, SpringSource

April 20, 2009

SpringSource 411 Borel Ave, Suite 101, San Mateo, California 94402, USA **T** +1 650 425 3515 **F** +1 650 212 3515 **E** insidesales@springsource.com

Executive Summary 2

Who is SpringSource? 3

Occam's Razor 4

Tomcat and Spring: KISS-ing Cousins 4

The Hidden Cost of Complexity 5

- Installation and Provisioning 5
- Configuration and Quality 6
- Upgrade Process 6
- Migration Process 6
- Infrastructure and Server Expansion 6
- Operational Management 6
- Developer Productivity 6
- Fluidity of Development Resources 7
- License and Maintenance Costs 7

Apache Tomcat 8

- Tomcat is the #1 Application Server 8
- Job Market for Tomcat Developer Skills 8
- Deploying Tomcat Widely Across the Enterprise 9

Introducing SpringSource tc Server 11

- Tomcat Compatibility Enables Seamless Upgrade 11
- Enterprise Ready Tomcat 11
- Business Critical Operational Capabilities 13

SpringSource Enterprise, ERS and tc Server 16

Groovy and Grails 17

Conclusion 18

End Notes 19

Executive Summary

Many enterprises are awash in complex, heavyweight, and brittle application infrastructure and processes that significantly impact their agility. The lean approach to creating software has emerged as an answer to this problem – brought into the mainstream by the maturity and adoption of technologies and processes specifically designed to remove complexity, increase productivity and quality, and accelerate the application lifecycle.

Forrester defines the "lean software" movement as: "*an approach to building, delivering, and running software that values fit-to-purpose, simplicity, and time to results above all. Lean approaches minimize complexity, startup time, and resource usage and avoid features and methods not essential to fulfilling the application's business purpose. Developers can easily combine lean software components with others when large systems require more features.*" [1]

The lean software movement has been building for years and Forrester lists SpringSource as one of four companies at the forefront of the movement. Today's economic climate is only accelerating the need for organizations to embrace technologies and processes that enable them to rapidly build, run, and manage the software solutions the business requires today and tomorrow. The combination of technologies such as **Spring and Apache Tomcat** provides a proven alternative to the complexity inherent in today's heavyweight Java Enterprise Edition (Java EE) platforms.

This paper provides an overview of the **business and technical benefits** of **SpringSource tc Server**, an enterprise ready version of Apache Tomcat hardened for enterprise use and coupled with key operational capabilities, advanced diagnostics and backed by mission-critical support. This paper provides information that helps justify using SpringSource tc Server to run web applications that may be currently running on Apache Tomcat or Java EE platforms from IBM, Oracle, Red Hat, or Sun.

Who is SpringSource?

SpringSource is at the forefront of rapid enterprise adoption of "lean software" that dramatically cuts cost and complexity, increases productivity, and accelerates the delivery of high-quality, business-critical applications. By addressing excessive complexity in the application architecture, underlying platform, and the processes used to create business solutions, SpringSource enables customers to accelerate the delivery of solutions their business demands.

SpringSource, its employees, and a massive open source community are the source of innovations transforming the enterprise Java software market. SpringSource employs the **open source leaders** who created and continue to drive innovation for Spring, the de facto standard programming model for enterprise Java applications. SpringSource also employs the Java and Web thought leaders within the Apache Tomcat, Apache HTTP Server, Groovy and Grails open source communities.

SpringSource forges this open source innovation into a complete suite of software products designed to accelerate the entire build, run, and manage lifecycle:

- from high productivity developer tools and frameworks,
- to lightweight application server runtimes,
- to complete application infrastructure management and monitoring.

Already, most of the Global 2000 turn to SpringSource for commercial software and support for powerful and lean application infrastructure software products powered by Spring, Groovy, Grails, Apache Tomcat, Apache Web server, and other popular open source technologies.

This whitepaper focuses on:

- **SpringSource tc Server: Apache Tomcat** hardened for enterprise use and coupled with key operational capabilities, advanced diagnostics and backed by mission-critical support.

This paper also briefly touches on:

- **Spring**: The de facto standard programming model for enterprise Java applications, including centralized configuration, declarative transactions, declarative security, messaging, remoting, Web MVC, Web Flow, Web Services, persistence integration, enterprise integration, batch, and other capabilities.

Please read the companion white paper entitled **SpringSource tc Server Migration Strategies** if you are interested in the technical strategies and detailed steps required to migrate enterprise Java Web applications from traditional Java EE application servers, such as Oracle Weblogic and IBM WebSphere, to the lean yet powerful Apache Tomcat-based SpringSource tc Server.

- **SpringSource Enterprise**: An enterprise version of the ubiquitous Spring container that provides full instrumentation for detailed insight into health and performance of Spring applications via a Hyperic HQ-powered application management console, as well as performance and reliability optimizations for Oracle Database environments.
- **Apache HTTP Server**: The most widely deployed Web server on the market today.
- **SpringSource ERS**: An enterprise ready version of Apache HTTP Server that provides better performance and security and improves service quality, reliability and scalability of Web infrastructures.
- **Groovy and Grails**: Grails is a next-generation web application framework designed for rapid application development that is built on Spring and based on Groovy, a powerful dynamic language that provides seamless integration with Java.to lightweight application server runtimes,

Occam's Razor

Occam's Razor (also Ockham's Razor) is a principle attributed to the 14th-century English logician and Franciscan friar William of Ockham [2]: **When confronted with multiple solutions to a problem, choose the simplest one**. Moreover, Albert Einstein said "*Everything should be made as simple as possible, but not simpler*" [3]. Many developers and technologists these days state it even more succinctly: KISS (keep it simple stupid).

The Java Enterprise Edition (Java EE) specification contains a wide range of interesting and advanced technologies, and software vendors have managed to create impressive, feature-rich implementations of the Java EE platform. A side effect of the monolithic Java EE platform specification, however, is that businesses end up paying for far more Java EE technology than they actually need, making their applications more complicated and expensive as a result.

Since the majority of Web applications typically only need a Java EE compliant Web container that supports the Java Servlet and JavaServer Pages specifications and provides HTTP session clustering support for scalability, companies have turned to Apache Tomcat for their application server needs. Tomcat is a lightweight Java application server that has been around for a decade and has a very small download footprint of less than 10MB.

Tomcat is an open source implementation of the Java Servlet and JavaServer Pages specifications from the Apache Software Foundation and provides HTTP session clustering support for scalability. Tomcat perfectly illustrates Occam's Razor, and this fact has made it the most deployed Java application server in the market today.

Tomcat and Spring: KISS-ing Cousins

For organizations that require a full featured container for implementing business logic but also want a lightweight modular programming model, a popular approach is to use Spring with Tomcat. Spring has grown into an alternative

lighter weight approach to the full-stack Java EE platform, and according to Gartner, there is a "*Large, loyal community of developers using open-source Spring Framework in combination with market leading application servers, especially Apache Tomcat*" [4].

Spring is a set of open source technologies that provide a comprehensive set of enterprise Java infrastructure services. Spring provides a number of features normally only found in full stack application servers, including declarative transactions, declarative security, messaging and remoting. Spring also provides additional high value features, including configuration, Web MVC, Web Flow, persistence integration, enterprise integration, batch, and other functionality.

One reason for Spring's widespread adoption is that it enables developers to write their application code as Plain Old Java Objects (POJOs), avoiding the need to implement special complex interfaces or be aware of the container lifecycle as required with the classic Java EE programming model. Developers find the pairing of Tomcat and Spring a natural fit since both offer lean yet powerful approaches driven by the KISS principle.

The Hidden Cost of Complexity

The current economic conditions are driving businesses to take a deeper look at the cost of their software solutions. These costs are not only license costs, but costs across the entire application lifecycle (cost to build, cost to deploy, cost to run, cost to manage, cost to maintain and so forth). The time has come to look at technologies that not only provide a solution to your problems, but do so in the simplest most cost effective manner. The following examples are intended to illustrate how the costs of complexity can really add up versus a simpler alternative.

Installation and Provisioning

- Significant hardware and software costs as well as lost time and productivity can be incurred when installing and provisioning heavyweight platforms across the Developer, QA, Staging, and Production environments for 100's of applications.
    - Heavyweight application server environments typically require more expensive servers with multi-gigabytes of storage.
    - A formal IT process that takes days or longer is typically required to handle the provisioning and installation process.
- The tc Server runtime environment is simple to install and requires significantly less disk capacity (up to 100X less) than heavyweight alternatives and much lower memory requirements, resulting in a much lighter and more efficient application infrastructure.
    - More system resources are available for your applications rather than the underlying application infrastructure.
    - Developer self-service scenarios are possible when dealing with environments that take minutes to setup vs. hours or days.

Configuration and Quality

- Heavyweight platforms require more time to configure properly.
    - The costs of debugging complex configuration settings add up as the application moves through the Dev, QA, Staging, and Production lifecycle due to differences between environments.
- Lean application infrastructure is less complex to configure
    - Developers, administrators, and operators are able to focus on creating higher quality business critical systems rather than debugging complex configuration-related issues.

Upgrade Process

- Many organizations find the process of upgrading from older versions of existing heavyweight application servers to a newer version to be as complex and time consuming as migrating from one application server to another.
- Upgrading to the latest version can incur additional license and annual maintenance costs.

Migration Process

- Many organizations find that moving to a lighter weight platform actually speeds the migration process since they only have to deal with the complexity of the older source version and have a much simpler destination environment to deal with.
- Until tc Server, Tomcat has lacked the management capabilities required for production. As a result, organizations frequently resort to developing on Tomcat and deploying for production on a full Java EE application server. This in turn introduced additional cost and complexity involved in migrating applications from Tomcat to a Java EE platform.
- Please read the companion white paper from SpringSource entitled **SpringSource tc Server Migration Strategies** if you are interested in more information regarding migrating enterprise Java Web applications from heavyweight Java EE application servers to the lean yet powerful Apache Tomcat-based SpringSource tc Server.

Infrastructure and Server Expansion

- As your number of applications expands, you need to account for the additional license and hardware costs required to expand your footprint of your existing Java EE applications servers.

Operational Management

- The larger and more complex the application platform is, the more time and effort is required from the operations staff to operate and support the infrastructure. While traditional Java EE platforms may provide tools for the administrators and operators, these tools often add even more complexity into the environment.

Developer Productivity

- Heavyweight application servers can take minutes to start up and shut down even with small applications due to the overhead associated with the application server itself.
    - A typical developer can lose 30 – 60 minutes of time each day waiting for the server to restart during their typical develop/test/debug process.

- Lost productivity adds up for larger teams working on 100's or 1000's of applications.

- The Tomcat-based tc Server takes a matter of seconds to start up and shut down.

  - Developers have embraced Tomcat since it enables them to be much more productive throughout the develop/test/debug process.

Fluidity of Development Resources

- If an organization adopts a platform that is light, simple, and accessible to the public, it will achieve better fluidity of resources, benefit from a simpler learning curve, and be able to fill job vacancies more quickly.

  - There is no additional learning curve for developers using Tomcat, since a majority of developers know and use it already.

  - Below you will find an overview of the Job Market for Tomcat Developer Skills which shows the market for Tomcat skills is as vibrant as other enterprise Java alternatives.

License and Maintenance Costs

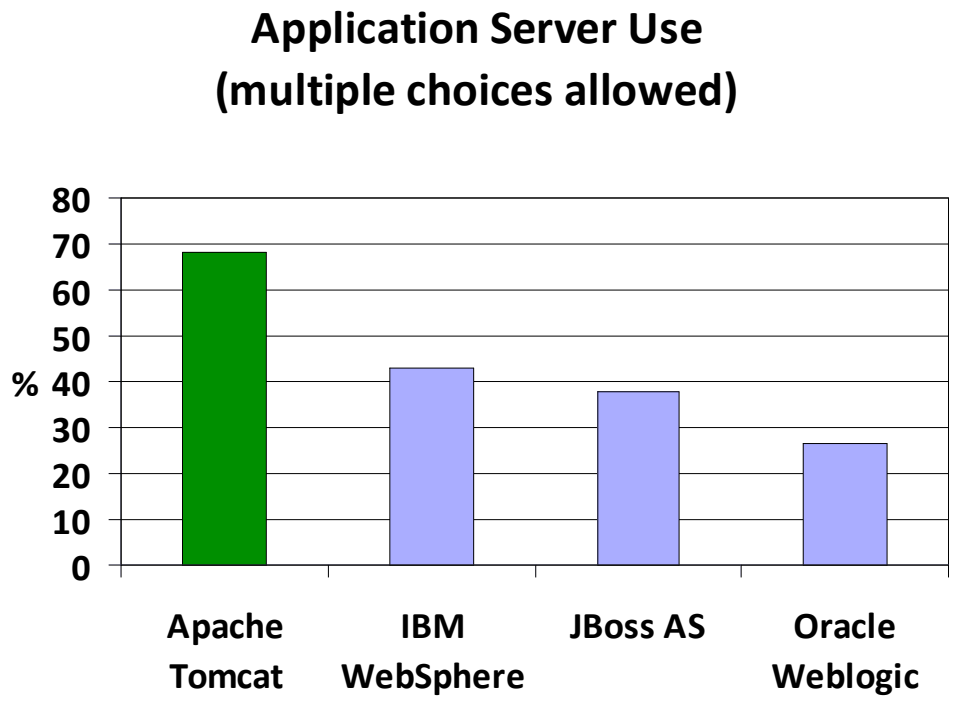
- A subscription for SpringSource tc Server is typically 33% to 75% less than what traditional Java EE application servers charge for annual maintenance. Significant additional savings occur when license costs are factored in since tc Server requires no upfront license fees.

Apache Tomcat

Apache Tomcat, the leading application server in the market today, is an open source implementation of the Java Servlet and JavaServer Pages specifications and provides HTTP session clustering support for scalability.

Tomcat is the #1 Application Server

According to a September 2008 survey conducted by Evans Data Research survey[5], **Apache Tomcat is the #1 most widely used application server** with 68% usage across organizations in their survey. IBM WebSphere Application Server placed second at 43%, JBoss Application Server was third at 38%, and Oracle Weblogic rounded out the top four at 26%. Similar BZ Research surveys focused on Java Use and Awareness[6] reinforce these results.

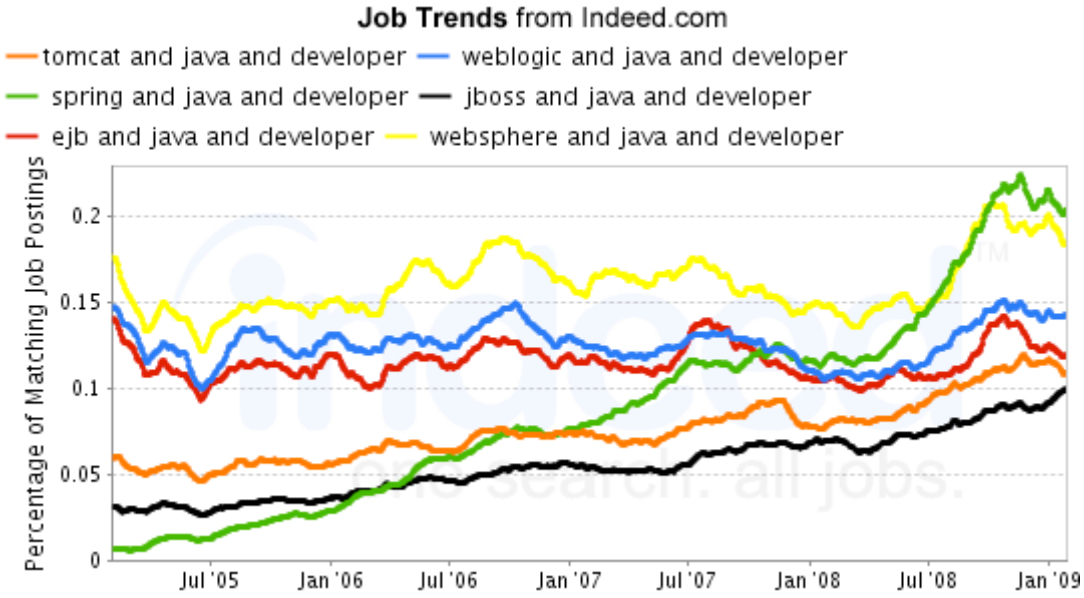


Most organizations deploy more than one application server; this explains why the above percentages add up to more than 100%. In order to better gauge true market leadership, Evans also asked organizations to specify the **primary application server** that they were using. **Apache Tomcat was #1 yet again** at 34%, while WebSphere was 22%, JBoss was 16%, and Weblogic was 11%.

Job Market for Tomcat Developer Skills

More and more companies are looking at lean application infrastructure to help them remain competitive, and the difficult economy is only accelerating this trend. Businesses not only need to feel comfortable with the technologies they embrace, but they also need to make sure they can easily find people experienced with the technologies.

The chart below was generated by using Indeed.com[7]. Indeed.com searches millions of jobs from thousands of job sites and provides a useful service that provides job trends for whatever job search criteria you may have. The search criteria for the chart below was Java Developers that have Tomcat, Spring, Weblogic, JBoss, EJB, or WebSphere skills.



The chart illustrates that **Spring skills (green line)** are in highest demand and have been on a steep incline for the past year. WebSphere skills (yellow line) right below Spring are not surprisingly high since the WebSphere brand encompasses a broad range of products. Weblogic skills (blue line) are next and have remained fairly flat over the years. EJB skills (red line) and **Tomcat skills (orange line)** are neck and neck behind that, with JBoss skills (black line) tracking behind Tomcat but on a similar path. The key takeaway of this data is that the market for Tomcat developer skills is as vibrant as the other enterprise Java alternatives which makes the choice of embracing Tomcat that much easier.

Deploying Tomcat Widely Across the Enterprise

Apache Tomcat's rise from developer workstations and departmental applications to its position at the top of the deployed application server market is remarkable. Whether it's a public website that processes millions of eCommerce transactions per month, FDA drug test result processing at a pharmaceutical company or a public site that tracks and reports roadway traffic for your drive home, you're likely to find Apache Tomcat at the core. The Tomcat project pages[8] and SpringSource's website[9] highlight a number of companies and organizations running Tomcat in production.

Other examples of large Tomcat deployments include:

- **A large financial services company** migrated from a Java EE application server to Tomcat for their central processing application handling between 2 - 20 million transactions a day across 80 high powered servers. The migration project to move to Tomcat took less than 90 days.

- **A leading online travel company** using over 10,000 Tomcat servers to power their online business. They chose Tomcat since they were able to scale out additional 2,000 Tomcat servers in a matter of hours to meet demands

during high volume periods. In contrast, it took them days to scale out a Java EE application server alternative. Since their business hinges on how quickly they can handle these peaks, this “hours vs. days” comparison made the choice of Tomcat clear.

- **A large pharmaceutical company** has 7,000 applications running on Tomcat across 1,000+ servers. They have standardized on Tomcat and only use Java EE application servers for a small percentage of packaged applications that require them to do so.

- **A leading financial services company** runs almost all of their Java applications on Apache Tomcat. This represents more than 2,000 distinct applications running on 1,200 servers. They've been using Tomcat almost exclusively for the past 5 years.

Deploying Tomcat in medium to large enterprise environments presents a challenge however, since Tomcat provides no tools at all to ease the process. As businesses deploy larger Tomcat server farms of tens, hundreds and thousands of instances, they typically need to build their own:

- Custom installation methods
- Custom centralized management, configuration and deployment solutions
- Custom diagnostic tools
- Custom monitoring tools

While other Java application server solutions may offer centralized configuration, administration, and management for their application servers, no standard solution has existed for Tomcat. This functionality has long been missing and the open source community has no plans to address these gaps since this functionality is beyond the scope of what the Tomcat community is focused on delivering.

The challenges inherent in deploying Apache Tomcat widely across the enterprise are real and have not, until today, been addressed by an enterprise solution that provides developers with the lightweight application server features they want paired with the operational capabilities that enterprises need.

Introducing SpringSource tc Server

SpringSource tc Server™ is an enterprise version of Apache Tomcat, the widely used web application server. SpringSource tc Server is hardened for enterprise use and is coupled with key operational capabilities, advanced diagnostics and is backed by mission-critical support. The key benefits of leveraging SpringSource tc Server across the application lifecycle include accelerated time to solution, reduced cost and complexity, and improved productivity and quality.

Tomcat Compatibility Enables Seamless Upgrade

SpringSource tc Server is designed to be a drop in replacement for Apache Tomcat 6, ensuring a seamless upgrade path for existing custom-built and commercial software applications already certified for Tomcat. Maintaining this level of compatibility enables our customers to add the business-critical functionality they need to more effectively run and manage their applications with the least amount of effort. This compatibility also provides an easier exit path for those who decide they no longer require the value of tc Server and want to downgrade to the base Apache Tomcat.

In contrast, existing Java EE application servers that embed Tomcat as the web container within a larger Java EE platform typically modify Tomcat (i.e. adding proprietary functionality and configuration) in ways that destroy compatibility and change the nature of the underlying Tomcat container. This means that unless a given application has already been certified to run on the Java EE server, a migration process including code and configuration changes may be required.

The key application server features of both SpringSource tc Server and Apache Tomcat include:

Application Server Features	SpringSource tc Server	Apache Tomcat
Servlet 2.5 Support (JSR 154)	✓	✓
Java Server Pages (JSP) 2.1 Support (JSR 245)	✓	✓
HTTP Session Clustering	✓	✓
Advanced I/O Features	✓	✓
Pre-built advanced non-blocking I/O components	✓	✓
Basic Windows service wrapper	✓	✓

Enterprise Ready Tomcat

SpringSource tc Server is based on Apache Tomcat 6 and therefore provides a powerful yet lightweight platform that is compatible with existing Tomcat-based applications as well as with Web applications that run on other Java EE application servers such as IBM WebSphere or Oracle Weblogic. Applications can be seamlessly moved from Apache Tomcat to SpringSource tc Server in order to gain the added benefits that SpringSource tc Server provides beyond the base Apache Tomcat.

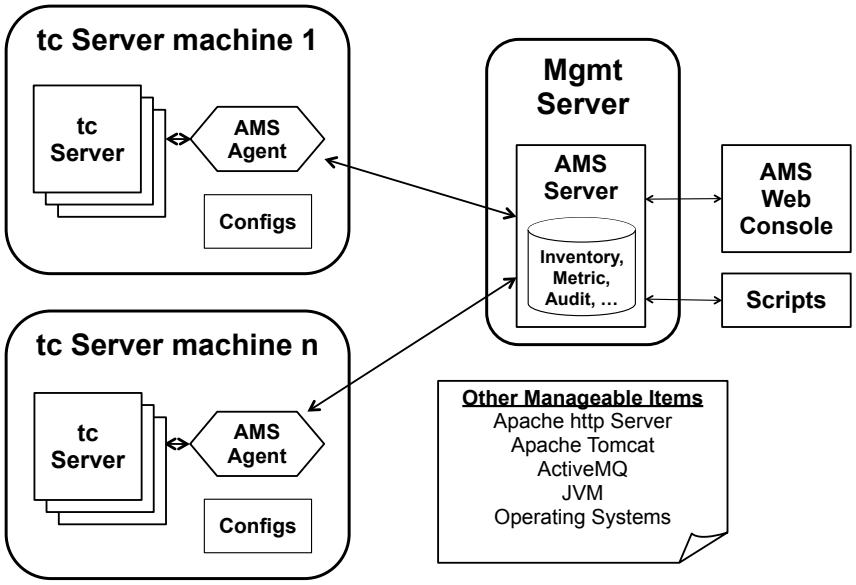
The following tables compare the application server features that tc Server provides versus Tomcat:

Additional Application Server Features	SpringSource tc Server	Apache Tomcat
Multiple runtime instances from a single binary installation	✓	
New high concurrency connection pool	✓	
Preconfigured for JMX management	✓	
Includes latest security vulnerability and bug fixes	✓	Rebuild Tomcat yourself to apply incremental fixes
Binary patch updates	✓	Binary patches are not provided by Tomcat community
Unix boot scripts	✓	
Enhanced Windows service wrapper	✓	

Advanced Configuration Features	SpringSource tc Server	Apache Tomcat
Templated production ready configurations Out Of The Box	✓	
Create Tomcat single server configuration	✓	
Modify general server configuration including JVM startup parameters	✓	
Modify context container configuration	✓	
Modify server defaults for JSPs and static content	✓	
Add, modify and delete JDBC data sources	✓	
Modify HTTP and AJP Connector settings	✓	
Create and View General Services	✓	
Modify General Engine configuration	✓	
Pre-tuned JVM Options	✓	

Business Critical Operational Capabilities

SpringSource tc Server includes advanced, distributed management and monitoring capabilities via a centralized management console referred to as AMS (Application Management Suite). The basic architecture of SpringSource tc Server is illustrated below:



SpringSource tc Server contains three major components:

- **tc Server Application Server**: The Tomcat application server at its core, tc Server is instrumented and integrated with AMS to provide an optimized management experience.
- **AMS Management Server**: This is the central controller for the management, configuration and operational capabilities of tc Server. It is a Java process that can run anywhere in your operations center.
- **AMS Agent**: This is a Java process that manages and monitors resources on an individual server or virtual machine within your operations center. One Agent can manage many resources and many resource types. The Agent accepts commands and queries from the AMS Server and interacts with managed resources such as tc Server to process those commands and queries. Interaction with tc Server occurs through a standard and secure JMX management interface.

AMS supports web based access to a graphical management dashboard supporting a full range of capabilities spanning configuration, management, monitoring and alerting. AMS also supports command line management capabilities to enable scriptable management of distributed server farms. Both the web-based graphical dashboard and the command

line scripting leverage the AMS Server as the primary interface into the system, with full role-based access control applied and an audit trail produced for all operations.

The following tables list the capabilities that SpringSource tc Server provides over and above the base Apache Tomcat and also notes the features that AMS provides for existing Apache Tomcat environments.

SpringSource tc Server provides a wide range of capabilities that enable developers, administrators, and operators to centrally diagnose, measure, and monitor the distributed application infrastructure their applications are deployed on.

Diagnostics, Metrics and Monitoring	SpringSource tc Server	Apache Tomcat
Application deadlock detection, including preconfigured deadlock trigger set to create heap and thread dumps with URL to thread association	✓	
Uncaught exception detection, including heap/thread dumps	✓	
Garbage collection metrics including throughput and count	✓	
SQL Query time monitoring metrics	✓	
Enhanced Response time monitoring metrics	✓	
Enhanced Connection pool health metrics	✓	
Enhanced Thread pool health metrics	✓	
Role-based, customizable dashboard	✓	✓ via AMS
Automated inventory of application servers and software resources	✓	✓ via AMS
Real-time metric collection and monitoring of tc Server, Tomcat, Apache Web Server, Apache ActiveMQ, underlying JVM, operating system, and other resources	✓	✓ via AMS
Historical charting and graphing performance	✓	✓ via AMS
Advanced alerting: multi-conditional, availability, event, and recovery alerts, group-based alerting, and escalation schemes.	✓	✓ via AMS

Diagnostics, Metrics and Monitoring	SpringSource tc Server	Apache Tomcat
Log file tracing, alerts on event levels	✓	✓ via AMS
Alerts based on configuration file updates	✓	✓ via AMS
Performance baselining for alert thresholds	✓	✓ via AMS

SpringSource tc Server provides a centralized, secure dashboard that enables administrators and operators to organize, operate, and control their distributed applications and infrastructure.

Centralized Operations and Management Features	SpringSource tc Server	Apache Tomcat
Secure, distributed, JMX-based server management	✓	
Create application server groups	✓	✓ via AMS
Application server start/stop/force stop from central console	✓	✓ via AMS
List deployed applications and current status	✓	
Application deploy/undeploy/reload/start/stop	✓	
Security and Access/Authorization Control	✓	✓ via AMS
Scheduled control: maintenance activities, on-demand actions, scheduled remediation actions, or scheduled responses to alert conditions	✓	✓ via AMS

SpringSource tc Server provides scripting support for administrators and operators who prefer to create and run scripts to handle distributed configuration and deployment steps.

Scripting Features	SpringSource tc Server	Apache Tomcat
List deployed servers	✓	
Create server groups	✓	
Add/delete servers to/from groups	✓	
List applications deployed including current status	✓	
Deploy application WAR file	✓	
Undeploy application	✓	
Start, stop and reload deployed applications	✓	
Get (download) configuration files and JVM parameters from a server	✓	
Modify configuration files on an individual server	✓	
Set (push) configuration files and JVM parameters to a server group	✓	
Start, stop and force stop a server or group of servers	✓	

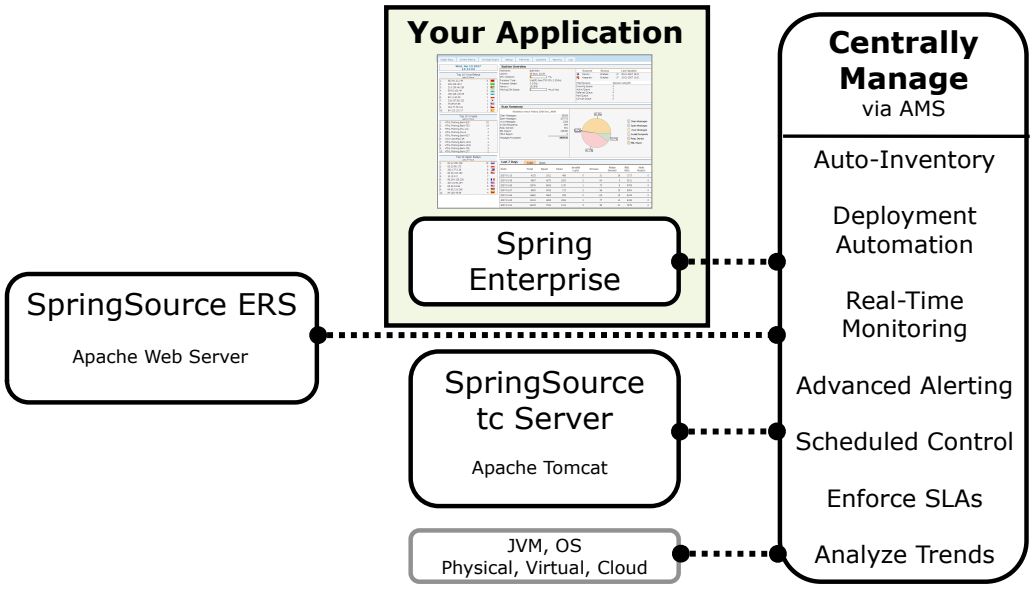
SpringSource Enterprise, ERS and tc Server

A typical enterprise Java web application deployment pattern requires a web server that front ends the application servers and a central management console that is capable of managing the end to end system. The most popular technologies used in such a deployment pattern are the Apache Web Server front ending requests to Spring-powered applications running on Apache Tomcat.

SpringSource provides supported products that offer an enhanced set of capabilities beyond Apache Web Server and Spring. These products include:

- **SpringSource ERS**: an enterprise ready version of the Apache Web Server that provides better performance and security and improves service quality, reliability and scalability of Web infrastructures. SpringSource tc Server and its AMS component includes management and monitoring of SpringSource ERS as well as existing Apache Web Servers. SpringSource ERS is an ideal front-end web-tier complement to tc Server.

- **SpringSource Enterprise**: an enterprise version of the ubiquitous Spring container that transparently instruments existing Spring applications, providing deeper insight into the health and performance of Spring-powered applications. This instrumentation plugs into the AMS management console to provide real-time monitoring, advanced alerting, and the ability to enforce application-level SLAs based on the Spring application blueprint.



Groovy and Grails

Grails is a next-generation web application framework designed for rapid application development. It is built on Spring and based on Groovy, a powerful dynamic language that provides seamless integration with Java. Grails applications follow a similar deployment pattern as Spring applications in the above diagram which means customers can comfortably run their Grails applications on SpringSource tc Server and front end those applications with the SpringSource ERS Web Server.

Conclusion

Technologies such as Spring and Tomcat have been leading the charge of a “lean software” movement that values fit-to-purpose, simplicity, and time-to-results above all. While Apache Tomcat is a perfect example of lean application infrastructure, the market-leading Java application server has lacked sufficient administration and operational capabilities required for medium and large scale enterprise deployments.

SpringSource tc Server adds these enterprise capabilities to Tomcat in a manner that maintains compatibility with applications tested and certified on Apache Tomcat. SpringSource tc Server adds a rich set of diagnostics, metrics, configuration, administration and business-critical operational capabilities that enable it to more cost effectively run web applications deployed on more complex Java EE platforms from IBM, Oracle, or Red Hat.

This paper discussed the hidden cost of complexity and provided examples of large scale enterprise deployments running on Tomcat today. An overview of SpringSource tc Server was covered as well as a detailed comparison of how its capabilities compare with the base set of capabilities provided by Apache Tomcat.

SpringSource’s focus is on delivering lean application infrastructure that enables our customers to dramatically cut cost and complexity, increase productivity, and accelerate the delivery of high-quality, business-critical applications. SpringSource tc Server provides these values to our customers by pairing the Tomcat people know with the business-critical operational capabilities needed to deploy and manage it across your enterprise.

ABOUT SPRINGSOURCE

SpringSource provides a complete suite of software products that accelerate the entire build, run, and manage enterprise Java application lifecycle. SpringSource employs the open source leaders who created and drive innovation for Spring, the de facto standard programming model for enterprise Java applications. SpringSource also employs the Java and Web thought leaders within the Apache Tomcat,

Apache HTTP Server, Groovy and Grails open source communities. Most of the Global 2000, including many world’s leading retail, financial services, manufacturing, healthcare, technology and public sector clients are SpringSource customers. For more information please visit www.springsource.com.

End Notes

[1] Forrester: Lean Software Is Agile, Fit-To-Purpose, & Efficient. December 2008. By John R. Rymer, David West, and Mike Gilpin with Jeffrey Hammond, Roy Wildeman, and David D'Silva

[2] "Occam's razor." Merriam-Webster's Collegiate Dictionary, 11th ed. 2003. (ISBN 0-87779-809-5) New York: Merriam-Webster, Inc.

[3] Albert Einstein: "Everything should be made as simple as possible, but not simpler".

[4] Gartner quote regarding Spring's use with Tomcat from Gartner Magic Quadrant for Enterprise Application Servers, 2Q08

[5] Evans Data Research: Spring Usage Study 2008 is a 30-plus page report that analyzes enterprise use of Spring and Apache Tomcat.

[6] BZ Research Java Use and Awareness Study can be found at http://www.bzmedia.com/bzresearch/index.html

[7] The Indeed.com Job Posting chart can be reproduced by going to the following link http://www.indeed.com/jobtrends?q=tomcat+and+java+and+developer%2C+weblogic+and+java+and+developer%2C+spring+and+java+and+developer%2C+jboss+and+java+and+developer%2C+ejb+and+java+and+developer&l=

[8] Sites, Applications, and Systems Powered by Apache Tomcat can be found at http://wiki.apache.org/tomcat/PoweredBy

[9] SpringSource customers using Tomcat can be found at http://www.springsource.com/customers