

Building and Running Spring Applications on SpringSource tc Server[™]:

A Technical Perspective

WHITE PAPER

Table of Contents

1. Building/Running Spring Applications on SpringSource tc Server™	3
2. Widespread Adoption of Spring and Tomcat	3
3. What is tc Server?	4
3.1 tc Server Runtime	4
3.2 tc Server Management Server	4
3.3 tc Server Agent	4
4. Spring and tc Server Share Common Philosophy	5
4.1 Both Spring and Tomcat Are Widely Used by Developers	5
4.2 Both Spring and tc Server Are Light Weight	6
4.3 Both Spring and tc Server Are Highly Modular	6
4.5 Both Spring and tc Server Offer Professional Support Subscriptions . . .	6
5. Combining Spring Framework and tc Server	7
5.1 Spring Framework Instrumentation	7
5.2 Spring Insight Console	8
5.2.1 Agile Development	9
5.2.2 QA Rear View Mirror	9
5.2.3 Load and Performance Testing	10
5.3 Spring Framework Application Deployment	10
6. Conclusion.	11

1. Building/Running Spring Applications on SpringSource tc Server™

Many of us have had the frustrating experience of developing, testing, and tuning our Spring Applications, only to be then faced with what can amount to a “port” to a different production environment, typically a commercial JEE Application Server. To add to this frustration, we know that our application doesn’t require 90+% of the functionality (not to mention complexity and cost) of that JEE Application Server in the first place, but we are aware that IT Operations management and monitoring capabilities are not found in our lightweight desktop development environments. It is also increasingly common that the production application will be hosted in a virtualized environment, which makes building, testing, and deploying on a common light weight platform even more valuable.

We know that we will be faced with doing this over, and over, and over, again during the application life cycle due to today’s business pressures to deliver incremental functionality (value) in the application on a frequent basis. This cross environment switching- which at its core is primarily administrative- can add hours to an otherwise simple deployment task. One large newspaper in the UK found that each cycle was taking them multiple developer hours, mostly porting the complex configuration settings and deployment metadata, for each update. Using Spring on tc Server reduced their update time to minutes, a dramatic savings, particularly considering their aggressive agile application development methodology.

The advantages of Spring are well known, as are the advantages of Tomcat, so this white paper focuses primarily on the advantages of combining the Spring Framework with SpringSource’s tc Server™ in order to create a highly productive, reliable, enterprise ready, production environment for deploying today’s mission critical JAVA applications. While it is definitely not required to use Spring and tc Server together, because each operates just fine on their own, the combination does offer many additional benefits due to their natural synergy and due to specific enhancements made to both the Spring Framework and to tc Server.

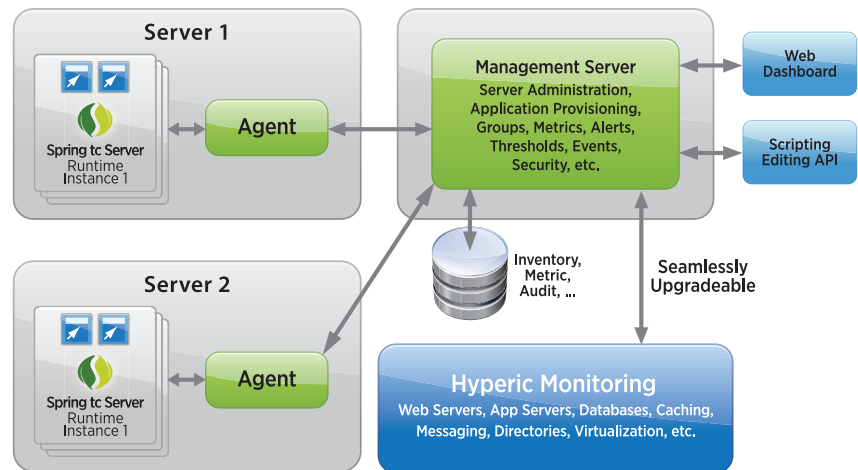
2. Widespread Adoption of Spring and Tomcat

The probability is that your organization has been using the Spring Framework for JAVA application development for many of your critical IT projects. In the seven years since its introduction, Spring has become very widely adopted, with over 3 million JAVA developers using Spring and 83% of the organizations with large IT development teams leveraging the Spring Framework to streamline their development processes. According to Evans Data, Spring is also used in over 50% of all applications running on the major JEE Application Servers and on Apache Tomcat.

It is also highly likely that you are using Apache Tomcat, at least in the development phase. Tomcat is by far the most widely utilized JAVA web applications server, in use at over 64 % of all corporate enterprises. While Tomcat is an excellent web applications server and widely used in developing applications that are later deployed on JEE Application Servers, it has lacked some of the administrative, management, and monitoring capabilities that IT organizations require, thus inhibiting its adoption in many production environments. To resolve this problem, SpringSource has introduced tc Server, a commercially supported, production ready, web applications server based on Apache Tomcat.

The combination of the Spring Framework with tc Server in enterprise development, test, and production environments provides developers with important advantages in several areas:

- A common modular, lightweight, extensible JAVA development/production infrastructure that is well suited to meeting both today’s agile business demands and horizontally scalable/virtualized deployment architectures.
- Sophisticated Spring application and server monitoring/control, not available in any other Spring deployment environment.
- Significantly improved handoff of Spring applications to both test and production.
- Powerful script based automation of repetitive functions, greatly simplifying test and deployment



3. What is tc Server?

Spring Source's tc Server is a Web Application Server, based on 100% Apache Tomcat, enhanced by adding enterprise scale server/applications management and monitoring capabilities, plus professional support. By maintaining 100% Tomcat compatibility, tc Server allows IT organizations to seamlessly transfer applications that are already being written on Tomcat in development to high volume production environments. Being light weight and modular like Tomcat, tc Server also provides development environments with automation and in depth monitoring that speeds development and greatly improves debugging and tuning of applications.

3.1 tc Server Runtime

The tc Server runtime consists of 100% Apache Tomcat 6.0.20 (or most current production version) and Tomcat plug-ins to allow managing and monitoring the Tomcat environment. Multiple runtimes can be installed and managed on a single physical server, and multiple physical servers can be running the same application code in a highly parallel horizontally scaled deployment architecture. When Spring is used on tc Server, the Spring Framework has it's own management and monitoring plug-in, extending tc Servers capabilities down into the framework and application itself.

3.2 tc Server Management Server

The tc Server Management Server is based on SpringSource's Hyperic HQ technology, specifically enhanced to provide server and applications management. The Management Server can control large numbers of tc Server runtimes and applications and can be installed anywhere from an individual developers desktop to the administrative servers in a large data center. tc Server's Management Server can also be expanded to support hundreds of other infrastructure environments by upgrading to Hyperic HQ. The Management Server includes a powerful web based GUI to provision, control, and monitor tc Server instances. Particularly important for production management of applications, the tc Server Scripting Engine allows IT staff to create automated processes for all tc Server management functionality. Either the GUI or the scripts can be utilize to manage servers, applications, and even monitoring.

3.3 tc Server Agent

A light weight server agent is installed on each physical server and communicates with the tc Server Management Server. The tc Server agent, plug-ins, and monitoring extensions can be installed at any time, without affecting development or the application in any way. The tc

Server Agent is responsible for providing provisioning services for the application, management and control to the Runtime environment, and a monitoring interface for the Tomcat plug-in, as well as for the Spring Framework if utilized.

4. Spring and tc Server Share Common Philosophy

In order to fully understand the synergy between the Spring Framework and tc Server, it is useful to explore the similarities and synergies of these two technologies. The Spring Framework and the Apache Tomcat project were developed by completely separate Open Source communities and at differing points in time, but it is interesting to observe that both teams shared a common architectural vision in many ways.

Both are completely modular, carefully maintaining a lean core that can be extended as necessary. Both projects focus on developer productivity and application performance. Both projects have maintained their focus over a period of years, keeping the vision consistent, even while the communities have evolved.

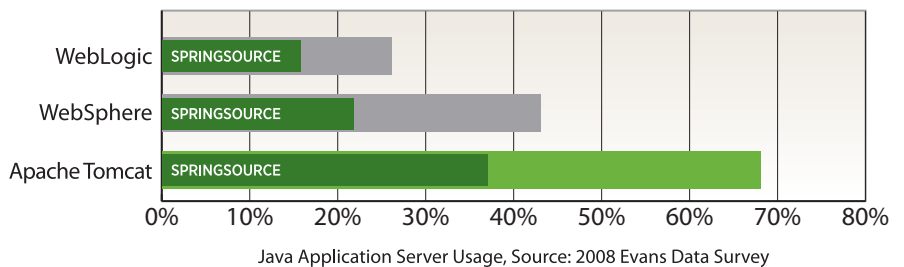
4.1 Both Spring and Tomcat Are Widely Used by Developers

Developers like Spring and Tomcat for similar reasons. Both of these simplify the developer's job and make them much more productive. Spring reduces (essentially eliminates) much of the JAVA "grunt work" and Tomcat eliminates most of the JEE Server time wasting steps to test, deploy, and run applications.

The Spring story is about improving JAVA Developer productivity, during design, coding, test, and support. As noted above, Spring is the most widely adopted JAVA Framework, in use by over 3 million developers. According to Evans Data, Spring accounts for over 50% of all deployed applications on the major JAVA Application Servers. Spring's adoption is a testimony to its productivity and power. Key architectural concepts, first introduced in Spring, have also made their way back into the JAVA Specification, replacing the time consuming, overly complex, and cumbersome constructs of EJB2.

Spring Leadership

- Used by 3 million Java developers
- Powers 50% of apps on IBM WebSphere, Oracle WebLogic, and Apache Tomcat
- Used by 83% of organizations with large development teams



Tomcat is without question the de-facto choice of JAVA developers for their desktop environments, again because of its simplicity, performance, and productivity. Tomcat installations take minutes, verses hours for the commercial JEE Application Servers. "Bouncing the Server", done dozens of times a day during portions (Debugging, QA, etc) of the development process, take 5-7 seconds for Tomcat, 5- 10 minutes for JEE Servers. That alone can be a savings in developer time of 3-5 HOURS per day. Configuring a JEE Application Server can take hours and requires significant training to get it right. Configuring Tomcat for development can take anything from no time at all to tens of minutes.

4.2 Both Spring and tc Server Are Light Weight

Spring, in particular, is well known for its ability to run anywhere there is a JVM, but some of these environments can weigh down Spring with unnecessary complexity and unused functionality. The very same Spring Application can be hosted on a tc Server instance whose footprint is tens of megabytes and a JEE application server that approaches a gigabyte.

Light weight infrastructures offer many operational and cost benefits, including significantly lower update and maintenance costs, reduced cost of servers, reduced facilities costs, and improving virtualization (frequently used in today's test environments) through better consolidation ratios.

4.3 Both Spring and tc Server Are Highly Modular

One of the biggest problems with the commercial JEE Application Servers is that they grew over time as highly integrated monoliths. This process was driven by an ever more complex JEE specification to some extent, but was also driven by the demands of the biggest/best customers who wanted more and more capability for the same cost. On the vendor side, it's easier to develop and test a non-modular product. The result became "everything for anyone", effectively making these environments too fat and too complex for 99% of the market needs.

Spring and tc Server have both steadfastly maintained a light weight core plus pluggable add-on architecture. This means that if your application requires messaging, for example, you add that module. If you do not need messaging, you aren't forced to carry the weight, complexity, and cost. One or two un-necessary functions doesn't seem like a huge burden, but when you look at the dozens of such functions in a typical JEE Application Server, they really add up, costing tens of thousands of dollars a year for absolutely no benefit.

Both Spring and tc Server Share a Common Management Architecture

The Spring Framework comes in both an instrumented version and a non-instrumented version. For development, debugging, and application tuning, it is useful to use the instrumented version. For production, the non-instrumented version reduces overheads and offers higher performance. Administrative access uses the JMX standard, but Spring does not force you to use any particular monitoring console, or any console what-so ever, unless you choose to do so.

tc Server's runtime also provides administrative interfaces based on JMX, but once again doesn't force any particular use of those interfaces. The tc Server Management Console uses the interfaces in both Spring and the tc Server Runtime to bring together all of the important operating parameters of Spring and Tomcat.

4.5 Both Spring and tc Server Offer Professional Support Subscriptions

Spring has enjoyed a high level of support since the founding of Interface21, Rod Johnson's company founded in 2004 to develop and support the Spring Framework. All of the major contributors to the Spring Framework worked at Interface21 and the business model was based entirely on providing excellent support and training services. In 2007, the company became SpringSource ("Spring from the "Source") and continued/expanded their service offerings, building a reputation for high quality professional support. In 2009, SpringSource became a division of virtualization pioneer, VMware, enabling Spring to further expand its service offerings and to add additional application infrastructure components to the portfolio.

One major barrier to adoption of Tomcat in mission critical applications had been the lack of professional support. While Tomcat does offer the IT organization multiple support choices, including self support and community support, only recently has enterprise level production support been available from a major vendor, the Spring Source division of VMware. There are some major Tomcat deployments using Self Support, but many of those companies have also found that Self Support is somewhat more costly to do than was expected and it can be very

difficult to maintain synchronization with the Apache community project. SpringSource has continued their philosophy of commitment to open source by investing heavily in the Apache project, and maintains multiple Tomcat committers within SpringSource.

The combination of world class support for Spring and Tomcat provides a valuable service to IT organizations seeking to leverage the Spring Framework on the Tomcat platform.

5. Combining Spring Framework and tc Server

Development organizations benefit in several other ways from explicit features included in the combination of Spring and tc Server, as well as from the inherent synergies discussed above. SpringSource has enhanced both tc Server and the Spring Framework in to make the combination even more productive for development organizations and to smooth the transition of applications from development to test to enterprise scale deployment. Note that these Spring Framework monitoring capabilities are not supported by commercial JEE Application Server management consoles and are unique to the combination of the Spring Framework and tc Server. These enhancements include:

- **Spring Framework Instrumentation-** When applications based on the Spring Framework are hosted on tc Server, we gain the ability to monitor the performance and resource utilization of the Spring Framework, including over 150 run time parameters. Additionally, alerts can be set for any of these, allowing developers, test organizations, and IT operations to get advance warning of impending issues. When the Spring Framework Instrumentation surfaces a potential problem, tc Server supports Spring Insight to do more detailed deep dive analysis, right into the internals of the application.
- **Spring Insight-** Spring Insight answers the question “what just happened”. It provides both a high level view into your application and deep dive visibility into your application’s internal activity on a request-by-request basis. Insight will show you which Spring MVC controllers are unhealthy and allow you to drill down into specific problem requests. With a few clicks you can navigate from the 10k foot view to a specific remote web-service call. For any request you can see all the JDBC queries it made, how much time it took to render, or timings for any of your major Spring beans.
- **Spring Framework Application Deployment-** tc Server directly supports provisioning, management, and monitoring of Spring Framework based applications, including managing deployment to distributed groups of tc Server instances. This allows developers to create applications in a light weight modular environment (Tomcat + required plug-ins), then transfer the application to test and on to production without having to create completely different deployment and configuration files.

By enhancing Spring to make it more manageable and by enhancing tc Server to enable it to provide sophisticated Spring monitoring, application management, and server management, SpringSource provides capabilities that rival, and in some cases exceed, the monitoring and debugging capabilities of the best JEE Application Servers. This is all provided in an efficient, lightweight, modular, form and at costs an order of magnitude less than JEE Application Servers.

5.1 Spring Framework Instrumentation

Both development and IT operations organizations benefit from the ability to monitor what’s happening in the Spring Framework during application operation. Spring offers a “production ready” monitoring capability that has minimal impact on application performance, while providing metrics for the application’s behavior. These can be used to generate alerts for errant application behavior and even to trigger corrective controls for automatic problem resolution.

Exactly what kinds of metrics and operations does the production-ready Spring Instrumentation provide? More details are in the Spring Framework Instrumentation documentation, but highlights include:

- Monitoring execution times for code that uses @Controller, @Service, @Component, @Transactional, and @Repository stereotyped annotations

Metrics

- AverageElapsedExecutionTime(ms), ExecutionsPerSecond, InvocationCount, MaximumExecutionTime(ms), MinimumExecutionTime(ms), ThrownExceptionCount
- Monitor execution times and enable operations to be executed across a range of standard Spring components such as

Platform Transaction Manager

- Metrics: CommitsPerSecond, FailedCommits, FailedResumes, FailedRollbacks, FailedSuspends, ResumesPerSecond, RollbacksPerSecond, SuspendsPerSecond

Hibernate Session Factory has over 40 metrics exposed

- Metrics: QueryExecutionMaxTime(ms), QueryExecutionCount, QueryCacheHitCount, QueryCacheMissCount, EntityLoadCount, EntityInsertCount, EntityUpdateCount, EntityDeleteCount, ...

Default Message Listener Container

- Metrics: ActiveConsumers, AverageElapsedTimePerMessage(ms), FailedMessages, MessagesPerSecond, MessagesReceived, ScheduledConsumers
- Operations: start, stop, set Concurrent Consumers, set Max Concurrent Consumers, scale Max Concurrent Consumers, set Max Messages Per Task, set Idle Task Execution Limit

JMS Template

- Metrics: AverageElapsedTimePerMessageSent(ms), FailedMessageSends, MessagesSent, MessagesSentPerSecond
- Operations: set Receive Timeout

Java Mail Sender

- Metrics: AverageElapsedTimePerMessage(ms), FailedMessages, MessagesSent, MessagesPerSecond
- Operations: setHost, setPort

Thread Pool Task Executor

- Metrics: ActiveTasks, LargestPoolSize, PoolSize, QueueSize
- Operations: set Core Pool Size, set Max Pool Size, set Keep Alive Seconds

Spring Framework Instrumentation, used with tc Server, provides far superior monitoring and operational control for Spring based applications than is available in any other deployment environment. Test organizations, in particular, will find Spring Framework instrumentation invaluable when stress testing and performance tuning applications, while both developers and production operations staff gain invaluable information from within the container.

5.2 Spring Insight Console

Another key component of the tc Server Developer Edition is the Spring Insight Console, a dashboard view of real-time Spring application performance metrics. Without changing their code, developers can use Spring Insight to detect, analyze and diagnose application performance issues right from their desktops. Insight is not targeted to production deployments at this time, but it does provide a “deep dive” into Spring Framework based application behavior for development and test organizations.

In development and testing stages, developers can use Spring Insight to verify immediately whether their newly-written code is behaving as designed. QA engineers can pinpoint specific causes for “what just happened” and relay detailed information to developers. Stress testing an application typically tells you which URL areas are slow. By combining Spring Insight with your existing tools (such as JMeter), you can see not only which URLs are slow, but far more importantly “why”, thus accelerating your time to production.

Spring Insight uses AspectJ to load-time weave your web application. This means that you are not required to make any changes to your application to use Spring Insight. Zero! Spring Insight collects its data in memory and does not require a database or disk access — this makes it trivial to try out! When deploying an application to Spring Insight, you will need to give it more memory to accommodate the storage of traces, response times, etc. As the internal limits are reached, Spring Insight will discard traces to keep the memory footprint low. It provides configuration options to tune the memory footprint.

There are a large number of uses for Spring Insight, but we will briefly discuss three of those:

- Agile Development
- QA Rear View Mirror
- Performance Testing

5.2.1 Agile Development

Web application developers realize a massive increase in productivity when they can make changes and see the effect immediately. Typically, a developer makes changes to HTML or JSP and reloads the browser to verify that the modified application performs as desired. However, developers often lack a centralized tool that shows how their changes affect:

- JDBC queries
- Spring bean interaction
- Calls to external services

Large, popular frameworks such as Hibernate and Spring Web push much of the code that developers formerly wrote manually into a convenient library. This process saves time and improves maintainability. The downside is relinquishing control, which means that the developer may not know exactly what is going on behind the scenes:

- How many database transactions did a web request create?
- How expensive is it to use complex web parameter binding?
- What are the HTTP headers being sent to the developer’s REST application?

The Spring Insight Trace view solves these problems. It allows developers to make changes and verify their effectiveness immediately.

5.2.2 QA Rear View Mirror

Spring Insight gives QA a richer picture of an application’s performance, eliminating much of the work required to diagnose problems. As QA tests an application, typical problems include:

- Slow-loading pages
- Database grinding
- Stack traces

As these problems arise, QA engineers can browse to the Spring Insight dashboard and review all recent operations. They can access in-depth information that helps them track down bugs:

- A list of all database queries and their performance
- A detailed description of the web request, its parameters, and headers
- A list of component method calls and their parameters
- A list of all Spring components that were used and their performance

QA forwards this information to the developer, thus improving the turnaround time for identifying and resolving root causes.

5.2.3 Load and Performance Testing

Web applications must be loaded and stressed before being deployed in a production setting. Spring Insight works with your existing load-testing tools to answer two main questions:

- What was slow?
- Why was it slow?

After running a load test, Spring Insight displays a breakdown of all requests to Spring Web. It shows you:

- The response time trend over a designated period
- A histogram that identifies response time patterns and outliers
- Detailed statistics, such as 95th percentile response time

Using this information, you can drill down to specific information about why a request was slow:

- Did the request execute an extremely slow database query?
- Did it make a call to a remote system that locked up?
- Did it spend a long time rendering the result?

The request trace information that you access in the Trace view is also available when you analyze a performance test.

5.3 Spring Framework Application Deployment

The issues associated with developing on one environment and deploying on a different production environment are well known. This gets even worse when test is using yet a third environment. If there are differences between different “standard” JAVA environments, and there always are, then each handoff from development to production becomes a “port”. Customer experience is that this can take from hours to days to do. This wasn’t so bad when applications remained un-touched for months/years, but today’s demanding business environments are driving changes far more frequently, sometimes even within hours.

So, since there is a “porting problem”, why would we use one environment for development and different one for production, particularly when “developer licenses” are generally free? The simple answer is that most JAVA production environments are far too large and far too sluggish for efficient development. Further, since there are not development versions and production versions (it’s only a license difference), every developer has to deal with all the overhead of the full production environment.

This situation has led to developers using the Spring Framework, either stand alone or on Tomcat, for creating the applications on their desktops, then porting those applications to commercial JEE Application Servers for test and production. The ability to use the Spring Framework on tc Server changes all that. Both are lightweight and modular, and the tc Server production environment includes exactly the same runtime code as the developer is using on their desktop. Even tc Server’s Management Server is light weight and modular, particularly when compared to the commercial JEE Application Servers, allowing developers and test organizations to leverage tc Servers management and monitoring if desired.

Generally, application transfer from a development environment to a production environment is complicated more by configuration and deployment meta-data than by the application code itself, particularly when developing using the Spring Framework. tc Server’s Application Management capabilities, and in particular it’s powerful scripting capability, allow setting up critical application and environment configurations once, and then transferring them almost verbatim thru the handoff process. The addition of template driven deployment configurations makes this even easier, by allowing common configuration meta-data to be re-used across multiple applications.

Customers report that the seamless handoff of Spring Framework applications from development, to test, and to production on tc Server now takes them only minutes, critical when they are engaging in rapid paced agile development of new business applications and features for existing applications. This compares to multiple hours when they were using a commercial JEE Application Server.

6. Conclusion

The combination of the Spring Framework and tc Server provide both development organizations and IT Operations with a number of important advantages, including a lighter weight, more agile, deployment environment, more seamless transition from application development to production, significantly lower support costs (both internal and external), much better control and monitoring of Spring applications throughout the processes, and simplified production tuning and debugging. Some of these benefits are due to Spring itself, some are due to tc Server itself, and many are due to the optimized combination.

In addition, the availability of multiple support options, including 24 x 7x 365 production support, from one highly regarded vendor provides IT management with the assurance that when assistance is required it is available and that the support is backed by key designers and architects from each OpenSource community.

About SpringSource

SpringSource, a division of VMware, Inc., (NYSE: VMW) and the leader in Java application infrastructure and management, provides a complete suite of software products that accelerate the entire build, run, 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, Hyperic, Groovy and Grails open source communities. Nearly half 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 visit: www.springsource.com.



North & South America
+1 877-486-9273

Europe/Middle East/Africa
+44 1276 414300

Asia Pacific
+61 284040150