



Migrating Applications From IBM WebSphere to Apache Tomcat

www.mulesoft.com | info@mulesoft.com | 1-877-MULE-OSS

Contents

1	Overview	3
2	Examining Migration Objectives: Why Migrate a Java EE Application?	3
2.1	Reduce Complexity	4
2.2	Reduce Operating Costs	4
3	Examining Java EE Application Migration Targets	4
4	Confirming Migration Ability	5
4.1	Techniques for Determining Migration Suitability	5
5	Migrating the Application to Tomcat	7
5.1	Web Application Configuration and Code Migration	7
5.2	Web Application Deployment	9
6	Conclusion	10



1 Overview

While it has been commonplace (and well documented) for IT organizations to migrate Java EE applications initially developed on Apache Tomcat upward to commercial Java application servers, such as IBM's WebSphere, in recent years the trend has been reversing. There are a number of compelling reasons for creating new web applications using today's deployment architectures on Tomcat instead of WebSphere, but perhaps even more interesting is the trend to migrate existing Java EE applications from WebSphere to Tomcat.

In this white paper, we explore the reasons for migrating from WebSphere to Tomcat, considerations for making the decision, and techniques to be followed for a successful migration. Note that when we talk about Java EE applications, we are including applications, application components, and SOA service instances, each of which is an "application," even when the user is another service or application, rather than a human.

When we are discussing "migration," we are focusing on those applications that would transition from WebSphere to Tomcat without major re-architecture or re-write. While it is theoretically possible to move any application from WebSphere to Tomcat, those involving sophisticated transactionality (EJB or CORBA, for example) would require so much development effort that they should be considered a completely new application, not a migration.

While there is no single process for determining migration requirements and strategy, this white paper describes steps that can be used to make a migration decision and successfully implement it. The steps typically include:

- **Examine migration objectives:** determine the reasons for the migration program. These may include business needs, cost savings, technical/architectural objectives, etc.
- **Examine migration targets:** identify those Java EE applications with migration potential based on meeting the above objectives.
- **Confirm migration ability:** assure that the selected Java EE applications can be migrated from WebSphere to Tomcat. This step will typically involve code analysis, trial migration, etc. Watch in particular for the use of "hidden services"... portions of WebSphere "auto-magically" used by your Java EE application without explicit developer activity. Also important would be considering any WebSphere or third-party administrative tools used by IT staff, which might not be available for or compatible with Tomcat.
- **Identify external services:** Determine those services external to Tomcat that may be needed, factoring in costs to provide those services.
- **Convert code:** convert selected Java EE applications to Tomcat, coding as required and migrating administrative/deployment/management tooling as required.

2 Examining Migration Objectives: Why Migrate a Java EE Application?

This is the first, and probably most important, question to ask, particularly when you already have a stable running Java EE application on WebSphere and there will be both costs and risks associated with

making the changes. There may be a number of answers to this question, including the need to expand capacity, reduce IT systems complexity, retire WebSphere licenses due to vendor changes or to reduce overhead, and to reduce ongoing costs.

2.1 Reduce Complexity

Another frequent objective for migration is to reduce the complexity of the IT environments. While WebSphere is extremely powerful, it is also very complicated, both to program and to administer. Today's IT budgets require "do more with less" processes and staffing, making it difficult to sustain such server environments.

Developer productivity is also somewhat higher with Tomcat, particularly for those applications that do not require the complexity of WebSphere. Whereas a Java developer may take days to set up and configure instances of WebSphere, they can often do this in hours or even minutes for Tomcat. During development/debugging, Tomcat can be stopped and re-started ("bounced") in a few seconds, rather than the minutes required each time by WebSphere. In one study, this impacted developer productivity by more than an hour per day.

Also driving complexity reduction is the need for IT agility. Whereas business applications used to take years to develop and deploy, business requirements now demand applications in weeks/months. This requirement drives architectural and process changes, including a transition to a more horizontal, lighter-weight, more modular deployment architecture. WebSphere is both too costly and too heavyweight to meet these requirements.

2.2 Reduce Operating Costs

Maintaining WebSphere is very costly, not the least because of the vendors service contract pricing, but also because of the amount of hardware required to support a WebSphere instance and the associated space/power/cooling costs. With typical Tomcat operating costs/server coming in at 1/4 to 1/6 the cost of WebSphere, many IT organizations are investing in transitioning all possible web applications from WebSphere to Tomcat, reserving their WebSphere servers for Java EE applications that require EJB or other major Java EE facilities that Tomcat does not offer.

3 Examining Java EE Application Migration Targets

Selecting suitable Java EE applications for migration is an important step. Typically, it depends on the objectives from above, some educated guesswork regarding what applications might be suitable for migration, and some thoughts about cost savings. With multiple interacting factors in play, this is typically a somewhat iterative process. For example, migrating a specific application might achieve business objectives, but upon detailed analysis, it might prove to be too highly integrated into WebSphere services that are not available in Tomcat and too costly/complex to provide as a separate Java process or separate add-on Java component.

There are three general types of applications for which Tomcat is highly suitable. These are:

1. Any web application that has primarily been developed using Tomcat during development and is currently running on WebSphere in production.
2. User interface (GUI), including sophisticated dynamic interfaces with features like user-specific content, forms validation, etc.

3. Application components (including SOA services), particularly those requiring highly horizontal scaling and non-demanding business logic.

Evolutionary changes in application architectures have made lightweight Java containers an increasingly viable environment for today's enterprise applications, particularly those using service oriented architectures based on Web Services standards. Increasingly, IT organizations are de-composing huge vertical legacy applications running on mega-servers into collections of more horizontal application components (Web Services, etc.) running on lightweight, low-cost servers. They are also encapsulating legacy systems in Web Services to enable a 'building block' approach to future dynamic IT requirements. In both cases, migrating portions of the application from the WebSphere server to Tomcat provides operational efficiencies and improves IT agility.

Where architectural changes are the motivating driver, you must still take a close look at both the costs and the details of the migration process. In some cases, portions of the existing application code can be readily split off, but often new code has to be written to glue the results together. Targets of opportunity in a conversion would include portions of the code that have to horizontally scale but that do not require sophisticated services. User interface code is one typical area to focus on, plus application components doing large volumes of relatively simple things.

4 Confirming Migration Ability

The next step in assessing migration is to determine whether the functionality used by the application requires a WebSphere server or whether it can be supported by Tomcat, perhaps with a small number of add-on application services. Studies have shown that in most cases, only very small portions of the WebSphere functionality actually was used in any one Java EE application. While having everything you'd ever need at hand in every server instance was convenient, it was also complex and highly costly, both in acquisition cost and ongoing maintenance.

In some cases, the Java EE application from WebSphere will use one or more services not present in the Tomcat server. These may include data access/persistence, messaging, e-mail, etc. In these cases, the selected application services must be installed and configured, and typically the Java EE application code would be changed to use those specific third-party services.

4.1 Techniques for Determining Migration Suitability

With the differences between WebSphere and Tomcat in mind and suitable Java EE applications targeted for migration, we can turn to the migration process itself. There are several steps involved in the migration, starting with determining that the selected applications do not require services that will be missing in the Tomcat environment.

You may use the following checklist to decide if your application can move from WebSphere to Tomcat:

Factor	Migrate to Tomcat?	Next Steps
Application has been developed using Tomcat as runtime during development	Yes	Determine configuration and deployment changes
Application primarily uses servlets and/or JSPs	Yes	Check if app leverages any WebSphere-specific services and identify their equivalent replacements for Tomcat
Application uses Spring Framework	Yes	Determine configuration and deployment changes
Application is written to be strictly standards-compliant	Yes	Determine if other Java EE technologies are used and identify equivalent replacements for Tomcat
Application is third-party software that is also available for Tomcat	Yes	Obtain the Tomcat version of the web application and deploy it on Tomcat
Application uses EJB or other WebSphere server functionality not readily available for Tomcat	No	Need code refactoring to remove the use of such functionality. This is a “new application,” not a migration.

An easy way to do a trial migration is to move the selected web application, perhaps stubbing out those previously noted sections, to Tomcat. This effort is primarily a configuration and deployment exercise and will be described in more detail below. At this stage, it is fairly rare for the web application to function fully, unless the original developers were meticulous in assuring portability. Typically, a short debugging session either will result in a basically running web application or will surface previously unknown reasons that the web application will not readily migrate.

Once the web application is basically running, it is extremely useful to be able to use the same test suites and tooling used to maintain the original WebSphere-based application. Although these are third-party tools, they are almost always Tomcat compatible and typically ship on Tomcat by default. Where WebSphere monitoring and debugging tools have been leveraged, there are no comparable capabilities included in Tomcat, so either third-party tooling or commercialized versions of Tomcat which include monitoring/debugging/administration functions must be used. This is typically the most frustrating and time-consuming part of the migration process, because WebSphere offers a rich suite, and the open source Tomcat does not include these features.

5 Identifying the Tomcat Version Matching your Websphere Version

There are several different major versions of WebSphere in use today, each implementing different versions of the Java Servlet, JSP, and other specifications. There are at least as many versions of Tomcat, implementing matching versions of these specifications, or being compatible with them. This section identifies the WebSphere major version and its corresponding Tomcat major version, and shows which versions of the specifications were in use at the time these application servers were released.

WebSphere Version	Specifications and their Versions	Equivalent Tomcat Version
8.0	Servlet 3.0, JSP 2.2, JSF 2.0, Java 1.6	Tomcat 7.0
CE 2.1.x and 2.2.x	Servlet 2.5, JSP 2.1, JSF 1.2, Java 1.6	Tomcat 6.0 or higher
7.0	Servlet 2.5, JSP 2.1, JSF 1.2, Java 1.6	Tomcat 6.0 or higher
6.1	Servlet 2.4, JSP 2.0, JSF 1.1, Java 1.5	Tomcat 5.5 or Tomcat 6.0
6.0	Servlet 2.4, JSP 2.0, JSF 1.1, Java 1.4	Tomcat 5.5 or Tomcat 6.0
5.1	Servlet 2.3, JSP 1.2, (no JSF), Java 1.4	Tomcat 4.1

Note that Tomcat 6.0 requires Java 1.5 or higher, and Tomcat 7.0 requires Java 1.6 or higher.

6 Migrating your Application to Tomcat

At its most basic level, migrating a “clean” servlet/JSP application from WebSphere to an installed instance of Tomcat is a matter of bringing over the WAR file from WebSphere, setting up server.xml configuration parameters using your favorite text editor (including changing the hostname in server.xml, if you want to use anything other than the default localhost), deploying the application by placing the WAR file into the appropriate hostname, and starting Tomcat.

Once the software is installed, we have successfully moved clean example code from WebSphere to Tomcat and had it running in a couple of hours. While this is possible in an IT environment, it is by no means the rule.

6.1 Web Application Configuration and Code Migration

There are well known differences in the web application configuration and file layout conventions between WebSphere and those of the Java Servlet Specification standard. If your web application only uses the documented standards, it should run on Tomcat, or it will be much easier to make it run on

Tomcat. Making your web application more standards-compliant will also help you use more standard tools such as Eclipse and its plugins to ease web application development going forward.

Here are some of the important items you should check and migrate, depending on which version of WebSphere your application currently runs on:

6.1.1 On WebSphere 7.0

- From the Websphere admin console, click Environment > Shared Libraries and inspect the list of container-wide shared jars. You may need to copy each of those jars into Tomcat's lib/ directory, if there isn't already a copy of it there.
- Websphere 7's server-wide shared jars directory is the websphere/appserver/lib directory.
- If your webapp uses JDBC, the driver jar file will reside in Websphere CE's repository/ directory. Move it to Tomcat's lib/ directory. See Resources > JDBC Providers in the Websphere admin console.
- If your web application uses JavaServer Faces, download an implementation of JSF 1.2 and place the jar(s) in your web application's WEB-INF/lib directory. See the Apache MyFaces project at <http://myfaces.apache.org>.

6.1.2 On WebSphere CE 2.1.x and 2.2.x

- In server.xml, change any WebSphere-specific realm implementation to a Tomcat-supported realm class name.
- WebSphere CE runs a bundled version of Derby database. If you're using that, you'll need to download a separate release of the Derby jars and run that as part of your webapp. Place Derby's jar files in your webapp's WEB-INF/lib directory if you have just one web application that uses it, or place the jars in Tomcat's lib/ directory otherwise.
- If your web application uses the ActiveMQ JMS broker that is part of WebSphere CE, you must download a separate release of ActiveMQ and run it as a separate JVM process. Change server.xml to access this separate ActiveMQ broker.
- Configure Tomcat's connectors in server.xml by looking at the WebSphere CE property settings in `var/config/config-substitutions.properties`. See http://www.ibm.com/developerworks/websphere/library/techarticles/0808_jain/0808_jain.html#major5
- If your webapp uses JDBC, the driver jar file will reside in WebSphere CE's repository/ directory. Move it to Tomcat's lib/ directory.
- If your web application uses JavaServer Faces, download an implementation of JSF 1.2 and place the jar(s) in your web application's WEB-INF/lib directory. See the Apache MyFaces project at <http://myfaces.apache.org>.

6.1.3 On WebSphere 6.1 and 6.0

- From the WebSphere admin console, click Environment > Shared Libraries and inspect the list of container-wide shared jars. You may need to copy each of those jars into Tomcat's lib/ directory, if there isn't already a copy of it there.
- If your web application uses JavaServer Faces, download an implementation of JSF 1.1 and place the jar(s) in your web application's WEB-INF/lib directory. See the Apache MyFaces project at <http://myfaces.apache.org>.

6.1.4 On WebSphere 5.1

- Websphere 5.1 ships with a very old version of JDOM. You may want to use a newer version of the JDOM jar. Place it into your webapp's WEB-INF/lib directory if you have just one webapp that uses it, or in Tomcat's lib/ directory otherwise.
- If you are using scriptlets, you may have to unescape some double quotes. For example:
`<html:text property="<%= \"tax[\" + pageContext.getAttribute(\"i\") + \"].company\" %>"`
The above had to be changed to:
`<html:text property="<%= "tax[" + pageContext.getAttribute("i") + "].company" %>"`
in order to work properly in Tomcat.
- If your web application uses JavaServer Faces, download an implementation of JSF 1.0 and place the jar(s) in your web application's WEB-INF/lib directory if you have just one webapp that uses it, or in Tomcat's lib/ directory otherwise. See the Apache MyFaces project at <http://myfaces.apache.org>.

Once you've made these changes, your web application will be significantly more standards-compliant and can run on standard web containers such as Tomcat.

6.2 Web Application Deployment

Deployment is a significant area of difference between WebSphere and Tomcat. In WebSphere, Java EE applications are deployed through the administration console, and the scope of application management is a WebSphere cluster.

Note: Tomcat lacks even rudimentary administration, which means you must manually edit configuration files during installation. Commercialized versions of Tomcat add administration/management/monitoring capabilities to the basic Tomcat distribution, which adds significant value to Tomcat.

Tomcat offers three options for deploying a web application:

- WAR file: Manually moved into the local machine, manual or automatic local configuration
- Unpacked web application directory: Manually moved into the local machine, manual or automatic local configuration
- TCP connection to remote manager webapp: Move either the WAR or the unpacked directory to the remote manager app.

With stock Tomcat, you'll need to choose one of these deployment methodologies for each web application you want to deploy.

7 Conclusion

There are several steps to consider when migrating a Java EE application from one application server to another, but migrating to Tomcat can bring your technology stack up to date while also offering significant licensing cost savings. At the same time, you can move to a popular open source application server that is well known and heavily used in the enterprise, and make your web application more standards-compliant to allow your developers to use their choice of industry standard developer tools.

Migrating to Tomcat from WebSphere will help your bottom line, improve developer productivity, enable your operations staff to keep your web applications performing well, and allow for easier incremental application server upgrades. End to end, your server-side environment will be significantly more agile and cost-effective.

About MuleSoft

MuleSoft provides enterprise-class software, support and services for the world's most popular open source application infrastructure products, Mule ESB and Apache Tomcat. With the lightweight and powerful Mule ESB and MuleSoft Tcat Server, MuleSoft is bringing simple yet powerful infrastructure to today's dynamic Web applications. Today, MuleSoft products boast more than 1.5 million downloads and over 2,000 production deployments by leading organizations such as Walmart.com, Nestlé, Honeywell and DHL, as well as 5 of the world's top 10 banks. MuleSoft is headquartered in San Francisco with offices worldwide..

For more information: www.mulesoft.com, or email info@mulesoft.com.

Download Tcat Server: <http://www.mulesoft.com/download/>

MuleSoft and the MuleSoft logo are trademarks of MuleSoft Inc. in the United States and/or other countries. All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

All contents Copyright © 2011, MuleSoft Inc.