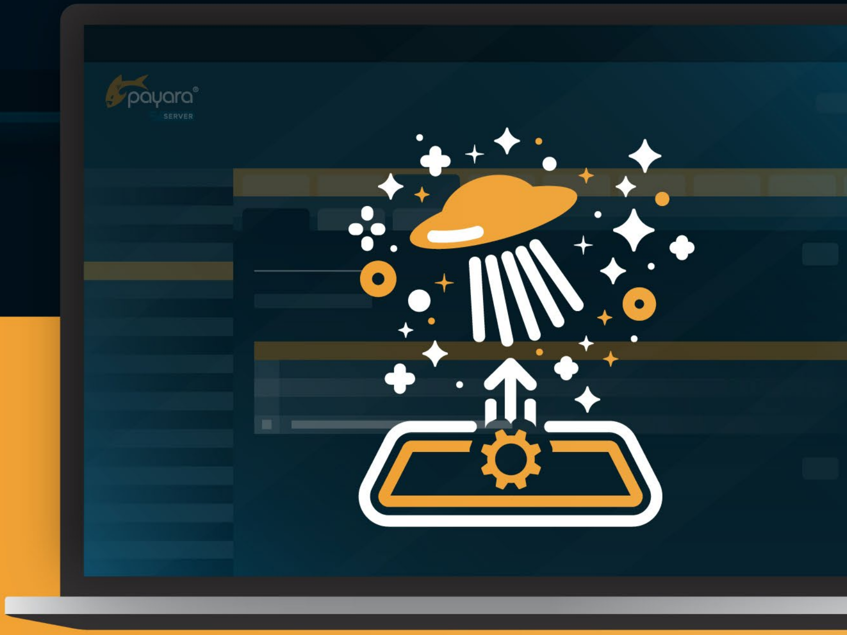
Enterprise Java Application Modernization Myths:

What Enterprises Get Wrong



Power Up Your Enterprise Java

User Guide

Contents

Guide Updated: **December 2025**

For many development teams, modernizing Java EE 8 applications is a business-critical initiative. However, the activity is often surrounded by numerous misconceptions that can hinder organizations from making effective, informed technology decisions. This guide uncovers the top myths and realities, explains exactly what companies relying on Java EE 8 workloads should consider, and demonstrates how a modern platform can offer a clear path forward, minimizing technical debt and maximizing ROI.

Myth #1: Java EE 8 is Dead

Java EE 8 is not a flashy new specification, its development stopped long ago and support from a number of technologies as well as vendors is certainly dropping. However, Java EE 8 is still widely used and powers mission-critical applications in productions environments across many sectors and industries. According to the 2024 Jakarta EE Developer Survey, 67% of enterprise Java developers still maintain Java EE workloads, with a significant 40% using Java EE 8.

While these applications are still running, the pool of runtime vendors with production support for Java EE 8 is rapidly shrinking, with limited options available for organizations that may not be able to modernize their systems just yet. In particular, with Red Hat's JBoss EAP 7 entering extended life cycle support (ELS) in July 2025, there are only two enterprise-grade application servers that currently offer full production support for the technology. The one with the longest lifecycle is Payara Platform Enterprise 5, whose maintenance support will begin in 2027 and end in 2029.

This means that while Java EE 8 is very much alive in enterprise environments, its long-term viability depends on choosing the right supporting technologies, e.g. a compatible, supported runtime. Organizations that want stability today and long-term support for a clear, not rushed migration path forward need to plan carefully. With its extended support timeline and enterprise-grade reliability, Payara Platform Enterprise 5 offers the most pragmatic choice.

Myth #2: Modernizations are Complex and Require a Complete Rebuild

A common misconception is that modernizing applications demands a total code rewrite, especially if cloud-nativeness is the ultimate goal. This misunderstanding often delays modernization initiatives, due to concerns about costs, time and excessive disruption to business operations.

In reality, modernization is not synonymous with "starting from scratch." There are several approaches, each requiring a different level of effort. In order of complexity and investment, they are replacing, retaining, retiring, rehosting, replatforming (or "lift and shift"), rewriting and refactoring. These are commonly referred to as the seven "R"s of legacy application modernization.

Therefore, enterprises can opt for a strategy that aligns with their own pace, as have to opportunity to alter their Java EE 8 applications incrementally, introducing new capabilities and infrastructure improvements while protecting existing investments. In other words, complexity can be managed, and "complete rebuild" is rarely necessary.

Myth #3: Java EE 8 Can't Be Made Cloud-Native

There's a widespread belief that Java EE 8 applications are fundamentally tied to legacy, monolithic deployments and, even when modernized to support more modern enterprise Java frameworks, they cannot be adapted for cloud-native environments. This assumption often pushes enterprises into premature, costly rewrites or refactoring projects when seeking to take advantage of microservices, containers and other cloud-related technologies.

While Java EE was certainly not initially designed with cloud applications in mind, as the popularity of cloud computing started to rise only recently, strategic modernizations of Java EE 8 projects to enhance cloud-readiness and -nativeness are possible. Even more, they have already been completed by industry leaders.

While many think that this is possible only through a complete rearchitecture or rewrite, it's important to stress that Java EE 8 workloads can be cloud-enabled in stages and starting with limited investments. Hence, enterprises don't need to abandon their existing investments to gain elasticity, observability and agility.

For instance, teams can retain/encapsulate their current Java EE 8 applications and exposing their core functions for consumption by greenfield microservices with cloud native features built in, so the original system can be maintained/hosted as is, but the application consuming the exposed core functionality from it is deployed to the cloud for better scale, performance and resilience.

Case study

A leading credit card provider partnered with Payara Services to modernize its IT infrastructure as part of a strategic shift to the cloud. The company was previously operating a large monolithic application hosted on a traditional enterprise Java application server. This setup had become increasingly difficult to scale, while the technical support received was expensive and slow, limiting the organization's ability to innovate.

Payara Services supported the customer first by migrating the applications to Payara Server Enterprise application server. Then, the company assisted the credit card provider in re-architecting its system, moving from the monolithic application to a microservices architecture using Payara Micro Enterprise. This shift enabled the credit card provider to benefit from a lightweight, cloud-native middleware solution that aligned with modern application development practices while providing greater scalability and resilience.

Payara Services' team provided hands-on guidance throughout the migration process — from refactoring to containerization and Kubernetes orchestration as well as configuration tuning and runtime support. By partnering with Payara, the credit card provider achieved improved availability, reliability, better resource utilization as well as faster deployment cycles. Post-migration, Payara's continued expert support ensured high performance and proactive issue resolution to help the customer optimize uptime and user experience.

Myth #4: Java EE 8 Applications are Locked into a Specific Runtime Vendor

Vendor lock-in has long been a concern in enterprise IT. With many companies aware of the growing obsolescence and incompatibility of Java EE 8 with newer technologies, many organizations assume that once an application has been deployed on a given Java EE 8 runtime, it is bound indefinitely to that vendor technology. This belief can foster anxiety over pricing, licensing changes and the loss of support as runtimes reach end of life (EOL).

The truth is that Java EE 8 applications are more portable than many realize. The specification was designed with portability in mind, which means that most applications can be migrated between compliant runtimes with little modification. While differences exist in terms of extensions and vendor-specific features, the core workloads are not permanently tied to a single environment.

In effect, the successor of Java EE, Jakarta EE, is designed to be vendor neutral by separating application code from server code. As such, its standard APIs help ensure consistency, compatibility and portability across different implementations, such as Payara Platform Enterprise. Therefore, migration from one Jakarta EE-certified runtime to another is simplified, so that enterprises can easily move away from one application server without undertaking full re-platforming efforts. This flexibility empowers organizations to choose a vendor that aligns with their modernization goals, cost structures and long-term support strategies as they evolve, breaking the myth that runtime choice is a one-way street.

Myth #5: Moving from Java EE 8 to Jakarta EE Locks You Out of Other Frameworks

Some organizations believe that once they commit to the namespace migration from Java EE 8 to Jakarta EE, they will be locked into that framework and cannot adopt alternative enterprise Java solutions, such as Spring/Spring Boot and Quarkus. This misconception often causes IT leaders to be concerned that modernization will limit future choices.

The reality is that modernizing with Jakarta EE does not preclude exploring or integrating other frameworks. The most common alternatives, Spring and Quarkus, share many similarities, as they are all based on Java EE/J2EE. For example, Spring integrates with several of the Jakarta EE specifications. Similarly, Quarkus implements subset of Jakarta EE and MicroProfile APIs. Therefore, the rewrite of Jakarta EE applications is possible.

In addition, modern enterprise Java solutions, such as Payara Qube, offer unified solutions to effectively manage diversity, supporting Jakarta EE, Spring and Quarkus, while minimizing manual configuration and setup. In other words, by selecting the right tools, organizations can support a heterogeneous tech stack that leverages the frameworks that are best suited to address the specific needs of each project.

The Bottom Line

Java EE 8 workloads remain a valuable asset and organization don't need to get rid of them today. In fact, while vend support is decreasing, with the right strategy, these applications can keep delivering value well into the future. What matters is not rushing into costly rewrites or believing in limiting myths, but instead choosing a pragmatic modernization path that matches your business goals.

With Payara Platform Enterprise, you can extend the life of existing Java EE 8 workloads, ensure long-term support and transition on your own terms, whether that's with incremental steps toward Jakarta EE, selective cloud enablement or by introducing modern frameworks alongside existing applications.

Interested in Payara? ***Book a Free Demo***

sales@payara.fish

UK: +44 800 538 5490
Intl: +1 888 239 8941

www.payara.fish