WebLogic Problems? Try This.

The Payara® Platform - Production-Ready, Cloud Native and Aggressively Compatible.

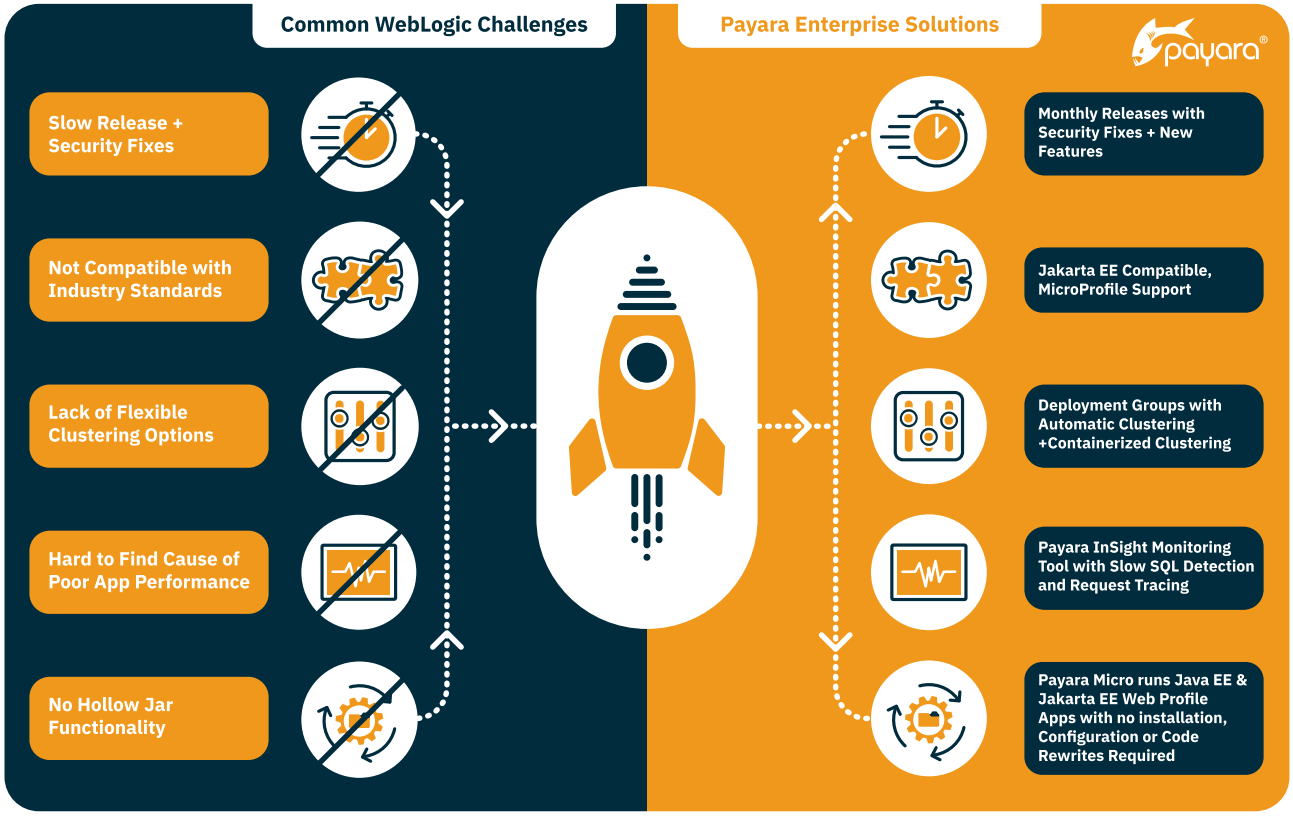
Contents

WebLogic Problems? Try This.

Running Oracle WebLogic in production? It's likely you are facing challenges due to irregular releases and component upgrades, lack of MicroProfile Support, lack of flexible clustering options - and wishing you had better monitoring capabilities.

Read below and see if you recognize the problems covered. It’s possible you didn’t think there was an alternative and that these issues were unavoidable. However, if you see yourself and your business in these issues, there is another way!

There is a solution to these common WebLogic problems that you maybe haven’t heard of yet.



WebLogic is Slow to Release New Features and Security Fixes

There is always a large gap between the releases of the WebLogic server containing new features. This also means upgrades of components like CDI and JAX-RS take a while and thus vulnerabilities aren't fixed as soon as possible.

The slow-release cadence is demonstrated below:

- WebLogic Server 12cR1 (12.2.1.3) in 2017
- WebLogic Server 12cR2 (12.2.1.4) in September 2019
- WebLogic Server 14c on March 30, 2020

Choosing an application server with a monthly release strategy with fixes, security updates, and new features ensures any vulnerabilities are patched as soon as possible while giving developers access to the latest features quickly.

Try This Fix

Slow or Not Compatible with Latest Standards

A prime example of WebLogic's slow compatibility with the latest standards is when WebLogic waited three years to support Java EE 8. WebLogic 14c in March 2020 was the first version to support Java EE 8.

WebLogic is not compatible with MicroProfile, specifications that are not only useful in a microservices environment, but also deliver application portability across multiple MicroProfile runtimes and are extremely useful in other applications, including:

- Configuration - to define application configuration outside your application
- OpenAPI support - to document your REST services
- Request Tracing - to follow your user requests through various parts of your application or across services
- Rest Client - allows you to call endpoints in a very compact way

Other examples of WebLogic lagging behind when it comes to supporting common standards include support for the JVM, as WebLogic Server was not supported on JDK 11 for six months after it was released. If you want to use the GraalVM runtime, you can only do so with the enterprise version of GraalVM (for an additional cost), and only on the Red Hat Enterprise Linux Operating System.

Using an application server that is not compatible with the latest standards, or is slow to implement support for them, means developers are working inefficiently and miss out on improvements to the development experience.

You can overcome these challenges by using an application server that's based on standards like Jakarta EE and MicroProfile, and one that maintains a regular release cadence to ensure you can take advantage of improvements to the development process as soon as they are available.

Try This Fix

Lack of Flexible Clustering Options

In many cases, you need to deploy your application in a clustered way. The application is available on different machines so that the requests can be distributed and thus higher throughput can be achieved. Hosting your application across multiple, redundant hosts guarantees reliability and allows for future scaling.

While WebLogic Server supports classic clustering that allows you to have different instances joined in a cluster, all running the same application or applications, it doesn't offer the ability to run different applications in the instances or extend clustering to containerized environments.

If you need flexible clustering options and automatic clustering capabilities or the ability to cluster within containerized environments, you'll want to look at an application server that include a feature like Hazelcast to assign your application to a cluster as well as all other resources it requires, such as a database connection pool setup, and perform the required configuration on the new instances as they are run.

Try This Fix

Hard to Determine Causes of Slow Application Performance

The WebLogic Diagnostics Framework (WLDF) is a monitoring and diagnostic framework. It provides WebLogic administrators with statistics for subsystems like the number of Threads active and the number of Sessions. Information around this monitoring can be written to various notifications channels like Log files, SMTP, SNMP, JMX, and JMS.

Oracle WebLogic has also support for more advanced monitoring features like Self-Health Monitoring. It detects if a Managed server needs to be restarted due to some failures or SQL Leak Connection detection. But WebLogic does not offer Slow SQL Detection or Request Tracing monitoring, which can make it difficult to know when a call is taking too much time or figure out the cause of slow application performance.

You can get better insight into your application's performance problems with an application server that offers more advanced monitoring features, preferably one that provides built-in monitoring over the use of an external monitoring system. Even better, choose an application server that also offers a wider variety of notification channels to receive monitoring alerts and the ability to write your own notifiers.

Try This Fix

No Hollow Jar Type Functionality

The classic application server requires you to install the software, perform the required configuration like defining the JDBC connection pools, and install the applications.

In the modern paradigm, a lot of companies switch to a Server Runtime. You just start up the server that is packaged in an executable JAR file, and during boot time, the configuration and application deployment is performed.

Especially in a Cloud environment, this flexible 'installation' is beneficial as instances are much more volatile.

WebLogic does not offer hollow jar functionality.

Try This Fix

How Payara Platform Overcomes Common WebLogic Challenges

If you're using WebLogic and struggling with the above challenges – you may find the solution is as easy as migrating to the Payara Platform. Payara Server Enterprise, an open source product with a 10-year software lifecycle, provides several benefits for running your Enterprise applications over Oracle WebLogic Server.

Payara Enterprise Has Monthly Releases with New Features and Security Fixes

To overcome your security vulnerabilities, slow WebLogic release schedule, and lack of compatibility for standards - Payara Server Enterprise is on a monthly release schedule with each release containing new features, fixes, and security updates. That way, you can be assured vulnerabilities are patched as soon as possible, and as a developer, you can benefit from all the latest features quickly, such as the latest JDK or compatibility with Jakarta EE and MicroProfile.

Payara Platform is Jakarta EE Certified and Supports MicroProfile

Java EE was donated to the Eclipse Foundation. It's first release as Jakarta EE 8 has been available since September 2019. Not long after this date, a Jakarta EE 8 certified compatible version of Payara Server was released (5.193.1 in October 2019).

Payara Server contained implementations for all the core MicroProfile specifications right from the beginning, and they are always updated to the latest version within a few months after new specifications are released.

Payara Server has Flexible Clustering Options with Deployment Groups

To overcome the lack of flexible clustering options, Payara Server has Deployment Groups. With Deployment Groups, you can also define that the application is associated with a Deployment Group, just as you can define a cluster for your application with WebLogic, but since an instance can be joined to multiple Deployment Groups, not every instance is running the same set of applications.

The setup of such a Deployment Group is also very easy and simple. You not only assign the application but also define the other resources it requires, like the database connection pool setup. All the required configuration, such as the connection pool, is performed on the new instance the moment a Payara Server instance joins an existing Deployment Group.

This flexibility is extended into containerized environments. The Payara instance can be in a Docker Container and when this container is started, it automatically joins an existing Deployment Group, and all the required configuration and deployments of applications are performed, and it can participate without any intervention into the handling of user requests.

Payara Server Offers Extensive Monitoring Features and Notifiers

For improved monitoring and insight into your applications performance issues, Payara Server offers Slow SQL detection and the Request Tracing feature whereas WebLogic does not. Both features give you an indication when a call takes too much time, which can be used to research the slow performance of your application.

The channels you can use to receive notifications for this monitoring subsystem are also more extensive on Payara Server than on WebLogic. Besides the classic channels like Log files or ELK stack, Mail messages, JMS messages, JMX Beans, Payara Server has a few additional channels supported by default:

- CDI events
- Slack
- New Relic
- DataDog

This gives you the greatest possible choice to monitor your system in production.

You also do not need to rely on an external system for the monitoring of the application in Payara Server like you do with WebLogic. Starting from version 5.194 you also have the Monitoring Console available within Payara. This gives you all the monitoring information from within the server itself and doesn't require any external system. The features of the Monitoring Console (Payara Community Edition) or Payara InSight (Payara Enterprise Edition) are improved which each release.

Payara Micro Offers Hollow Jar Functionality

For hollow jar functionality, Payara Micro, the microservices-ready version of Payara Server, offers this methodology without any special development adjustments as is the case for many other frameworks. Payara Micro is a packaging capable of running Java EE and Jakarta EE Web Profile-based applications. It is an executable JAR file that when supplied with an application (WAR or EAR), and a script file containing the asadmin commands for the configuration, gives you this runtime functionality.

Since it is based on the Web Profile, with the addition of the Concurrency and JMS client specification, you can just use your application as a microservice with the help of Payara Micro.

You can run Payara Micro with no need for using specific dependency, frameworks, or development idioms. The Jakarta EE and MicroProfile specification give you all you need to run the application as a microservice.

Learn More

Visit the “Learn” section of our website for links to everything you’ll need to know about using the Payara Platform (https://www.payara.fish/learn/)

Payara Server Enterprise – the best application platform for Jakarta EE (Java EE) apps. https://www.payara.fish/products/payara-server/

Payara Micro Enterprise – the platform of choice for containerized Jakarta EE (Java EE) microservices deployments. https://www.payara.fish/products/payara-micro/

sales@payara.fish **+44 207 754 0481** **www.payara.fish**