The Everlasting Age of "Write Once, Run Anywhere"

CELEBRATING JAVA@25

Your App Ecosystem | Driven by Java | Accelerated with Low-Code

Abstract

It takes a second to sow a seed, years to create a forest, and decades to build an ecosystem. Witnessing the Java community grow everyday is a fascinating experience in itself. Many of us have lived through the launch and wide-spread adoptionof Java and we continue to witness its evolution.

25 years ago, James Gosling and others at Sun Microsystems launched the Java programming language in 1995. The driving force behind creating Java was to build an architecture-neutral, platform-independent language for software embedded in electronic devices. Since then it has evolved into an ecosystem, from the Java Virtual Machine (JVM) to Jakarta EE (Java EE). To keep up with enterprise demands, the new focus of the Java ecosystem is on cloud computing with the aim of providing complete control and flexibility to accelerate application development in the cloud. Enriched layer upon layer, Java as a language has evolved over the decades and has created a special connection with many professional developers.

In this eBook, find out how Java is shaping the present and future technology, how it has evolved, the applications of Java and why it is popular, and how it forms the backbone of modern applications and is used to meet the modern enterprise demands.

Table of Contents

1.

Introduction

How Java is shaping present and future technology

Billions of devices today are still running on a Java platform. Since its launch in 1995, Java has become the transformational force in software development across decades. Java has stood the test of time and continues to evolve.

In this cloud-native world, mission-critical applications form the essence of business continuity, agility and digital dexterity. Shaping present and future technology, Java serves as the backbone of enterprise applications, powers the data revolution, enables applications on the cloud, connects the Internet of Things, and takes the world mobile.

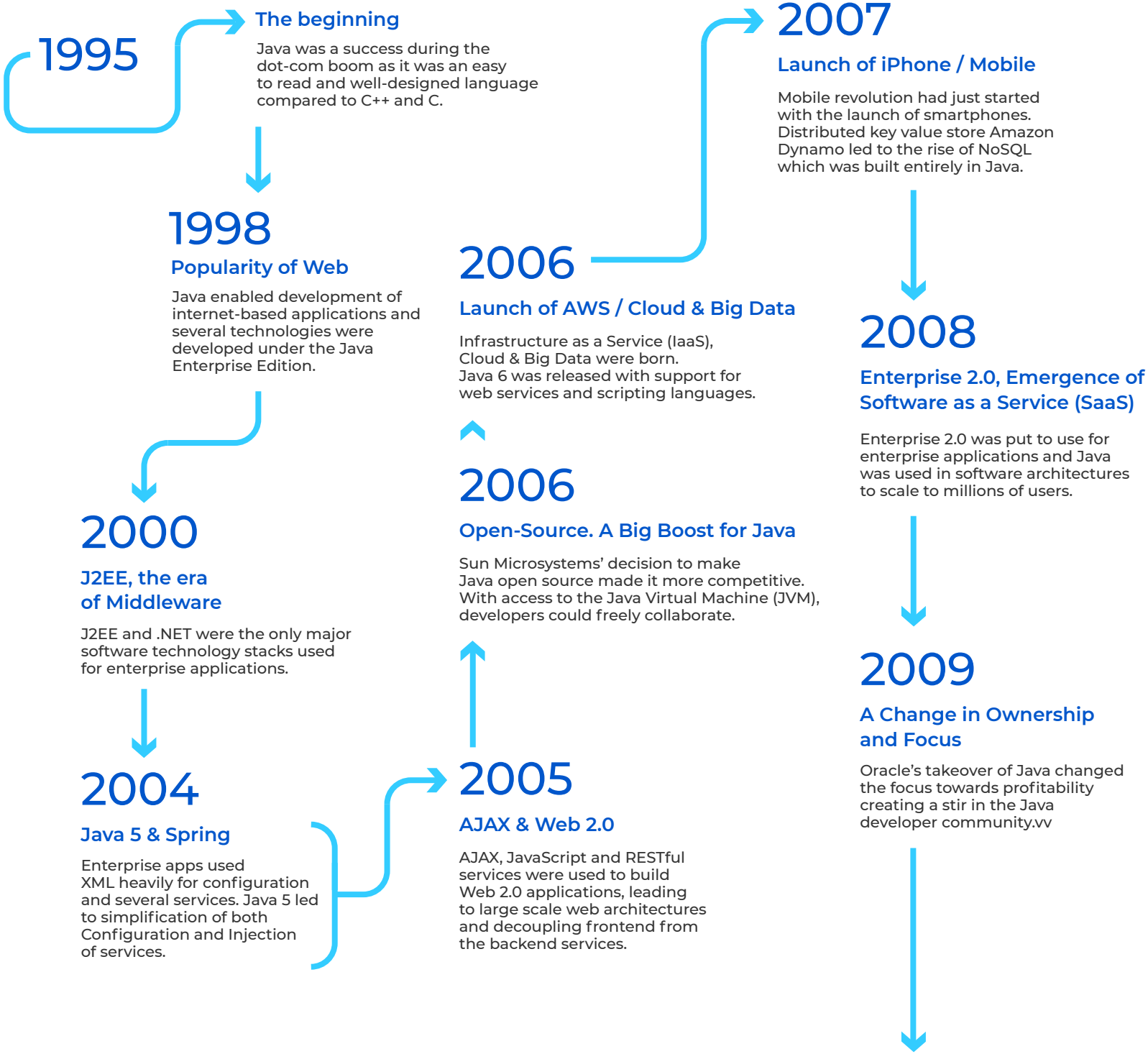
Whether you're modernizing an existing application or building a new one, low-code platforms can help Java developers upskill to become full-stack developers and supplement Java coding across the application development and delivery lifecycle.

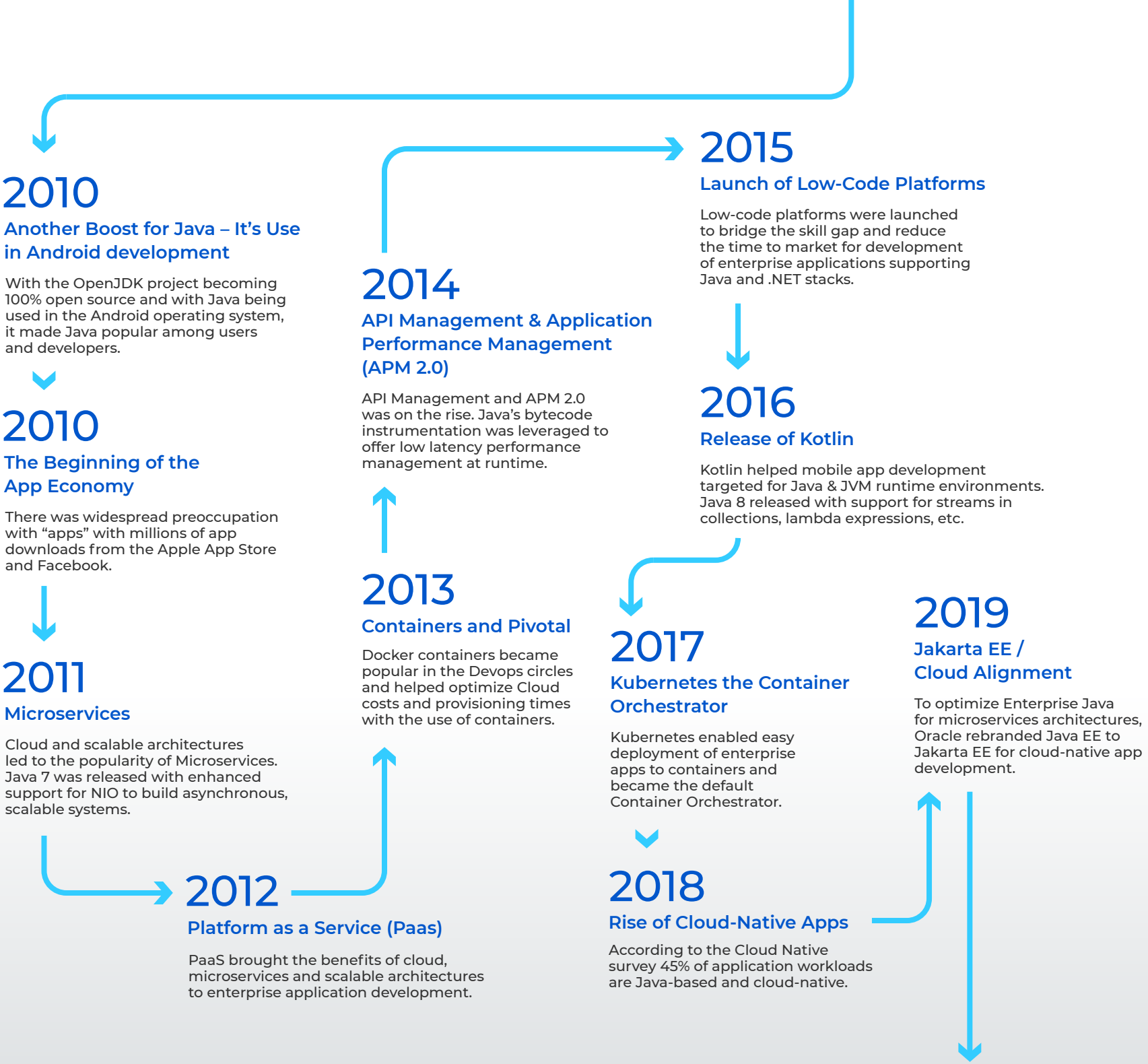
2.

Java is still young @25

The Evolution Over the Decades

Over decades, Java has shaped the way enterprise workloads and application servers are being built, meeting their needs of scalability, security, and reliability. Java also powered the mobile technology phenomenon we're experiencing today, and most Android applications are built using Java. Let's take a trip down memory lane and see how Java has evolved over the years.


1995
The beginning
Java was a success during the dot-com boom as it was an easy to read and well-designed language compared to C++ and C.
1998
Popularity of Web
Java enabled development of internet-based applications and several technologies were developed under the Java Enterprise Edition.
2000
J2EE, the era of Middleware
J2EE and .NET were the only major software technology stacks used for enterprise applications.
2004
Java 5 & Spring
Enterprise apps used XML heavily for configuration and several services. Java 5 led to simplification of both Configuration and Injection of services.
2005
AJAX & Web 2.0
AJAX, JavaScript and RESTful services were used to build Web 2.0 applications, leading to large scale web architectures and decoupling frontend from the backend services.
2006
Open-Source. A Big Boost for Java
Sun Microsystems' decision to make Java open source made it more competitive. With access to the Java Virtual Machine (JVM), developers could freely collaborate.
2006
Launch of AWS / Cloud & Big Data
Infrastructure as a Service (IaaS), Cloud & Big Data were born. Java 6 was released with support for web services and scripting languages.
2007
Launch of iPhone / Mobile
Mobile revolution had just started with the launch of smartphones. Distributed key value store Amazon Dynamo led to the rise of NoSQL which was built entirely in Java.
2008
Enterprise 2.0, Emergence of Software as a Service (SaaS)
Enterprise 2.0 was put to use for enterprise applications and Java was used in software architectures to scale to millions of users.
2009
A Change in Ownership and Focus
Oracle's takeover of Java changed the focus towards profitability creating a stir in the Java developer community.vv


2010
Another Boost for Java – It's Use in Android development
With the OpenJDK project becoming 100% open source and with Java being used in the Android operating system, it made Java popular among users and developers.
2010
The Beginning of the App Economy
There was widespread preoccupation with "apps" with millions of app downloads from the Apple App Store and Facebook.
2011
Microservices
Cloud and scalable architectures led to the popularity of Microservices. Java 7 was released with enhanced support for NIO to build asynchronous, scalable systems.
2012
Platform as a Service (Paas)
PaaS brought the benefits of cloud, microservices and scalable architectures to enterprise application development.
2013
Containers and Pivotal
Docker containers became popular in the Devops circles and helped optimize Cloud costs and provisioning times with the use of containers.
2014
API Management & Application Performance Management (APM 2.0)
API Management and APM 2.0 was on the rise. Java's bytecode instrumentation was leveraged to offer low latency performance management at runtime.
2015
Launch of Low-Code Platforms
Low-code platforms were launched to bridge the skill gap and reduce the time to market for development of enterprise applications supporting Java and .NET stacks.
2016
Release of Kotlin
Kotlin helped mobile app development targeted for Java & JVM runtime environments. Java 8 released with support for streams in collections, lambda expressions, etc.
2017
Kubernetes the Container Orchestrator
Kubernetes enabled easy deployment of enterprise apps to containers and became the default Container Orchestrator.
2018
Rise of Cloud-Native Apps
According to the Cloud Native survey 45% of application workloads are Java-based and cloud-native.
2019
Jakarta EE / Cloud Alignment
To optimize Enterprise Java for microservices architectures, Oracle rebranded Java EE to Jakarta EE for cloud-native app development.

What Strengthens the Future of **Java in 2020 and Beyond**

- **Ensures stability** – Software = short-term commodity. Java = long-term guarantee.
- **Modern solution to develop applications**
  - Using the Swing API or JavaFX, you can build desktop applications with Java.
  - With JavaFX, developed by the OpenJFX community, you can use graphical interfaces to build desktop applications.
  - Java is also the ideal language to develop Android applications.
  - Jakarta EE provides enterprise features such as web services and distributed computing. You can get a complete stack to develop your web applications.
  - The Java Card solution makes applications more portable across hardware-based and multi-cloud IoT security models.
- **Strong community** - With access to several Java-based, open-source projects, you can choose between a wide range of code libraries, and tools such as IDEs, dependency managers, and application servers.
- **Rich standard API** - With the JDK offered, you can create rich graphical interfaces for the desktop using Swing or AWT, manage inputs/outputs operations, and access databases without requiring external code libraries.
- **Easy to learn language** - With access to a large volume of free resources and training, learning Java programming language opens up the world of other programming languages that also run on the JVM. (Popular ones being Kotlin, Scala or Groovy)
- **Java is much more than a programming language** - With the Java Virtual Machine (JVM), Java is also an execution platform. You can not only run Java programs but programs written in other languages that are compiled to Java bytecode.

3. Applications of Java and Why it is Popular

Decades have passed since the launch of Java, yet the programming language is the backbone across multiple platforms used on multiple devices, embedded systems and in enterprise solutions. From Microsoft's ThinkFree Office to Google's Gmail, Google Docs, Google Earth, Google Sheets, and Google Slides, Java is used by big and small companies to develop different type of applications. The applications of Java are not limited to application and web development, it is seeping into emerging fields of artificial intelligence and big data technology.

Applications of Java	Applications of Java	Applications of Java
Desktop Applications	Supports GUI (Graphic User Interface) development. Uses APIs such as AWT (Abstract Windowing Toolkit), Swing, JavaFX.	Acrobat Reader, Antiviruses, Media Player, and ThinkFree.
Web-Based Applications	Offers Servlets, Struts or JSPs (Java Server Pages), JSFs (Java Server Faces), Hibernate, and Spring. A;so supports web servers such as Apache HTTP web-server, Apache Tomcat,	Online shopping carts, online forms, Google Slides, to Google Sheets.
Mobile / Android Applications	Offers Java Micro Edition (Java ME or J2ME), a cross-platform framework, and the Android Software Development Kit (SDK). Also compatible with Android	QR Reader app, Google Earth, Uber apps, to video gallery apps.
Distributed Applications	Offers APIs such as CORBA (Common Object Request Broker Architecture) and RMI (Remote Procedure Invocation). Also provides infrastructure support with Jini (Java Intelligent Networking Infrastructure) and JavaSpaces.	An enterprise application to forecast sales or order supplies across regions, or applications to manage distributed network of telecommunications and data

Cloud Applications	Offers cloud-based development tools such as the Oracle Java cloud service and serves as a platform using the Oracle WebLogic Server.	Offers cloud-based development tools such as the Oracle Java cloud service and serves as a platform, to create and configure cloud applications on an Oracle WebLogic Server.

Java also provides features that supports applications used in SaaS (Software-as-a-service), PaaS (Platform-as-a-service), and IaaS (Infrastructure-as-a-service) development.

4.

Combine the Old with New Technology

Meet the Demands of the Modern Enterprise

It's not an "either / or" decision. Combine traditional Java programming with low-code to meet modern enterprise demands.

Enterprise needs today have become more demanding. The new sense of urgency, to evolve, to modernize and drive mobility, scalability, and flexibility has led to enterprises embracing a 'digital first' approach and adopting new technology extensively. While this translates to reinvention in the way of working and the use of technologies, it does not mean that every aspect of the business needs to be reinvented. Old systems and skills that have stood the test of time need not be discarded altogether. Instead upgrading systems and updating skills can help to meet new enterprise demands with old tools and can result in easier implementation and better ROI on IT investment.

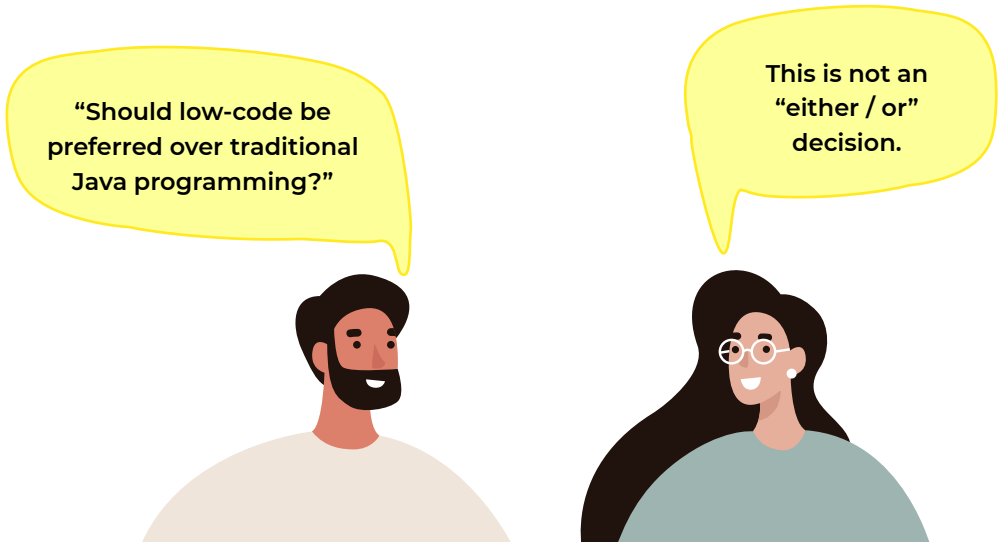
***Enterprise demands** = **A**gility, **M**obility, **S**calability, and Flexibility = Need for **developing enterprise applications faster** and better with **learner teams**.*

A relevant example of how emerging technology is combined with conventional tools is the use of low-code platforms with traditional Java programming in application development. Twenty years on and many enterprise servers are still using the Java programming language and successfully running Java-based, mission-critical applications. The fact that Java is still being used speaks of its evolution and effectiveness in addressing enterprise demands even in the age of cloud computing and container technology.

Embrace a '**digital first**' approach and adopt **new technology** to meet **enterprise demands**

How to accelerate application development?
Combine Emerging technology + Conventional tools = Easier and faster implementation + ROI on IT investment
Build leaner, agile teams with low-code

When creating application development strategies, there is often a debate about whether low-code should be preferred over traditional Java programming. However, this is not an "either / or" decision. While low-code platforms are designed to provide core functionalities, with the help of experienced developers customization and specialized functions can be added to applications. After all, a low-code platform is ultimately built on a programming language like Java.



Benefit of combining the old with new technology

Covers the micro and macro levels of application development.

By using **low-code platforms** = **Develop core functions** of apps rapidly + **Give experienced Java developers the bandwidth** to focus on critical specifications to make the application rich.

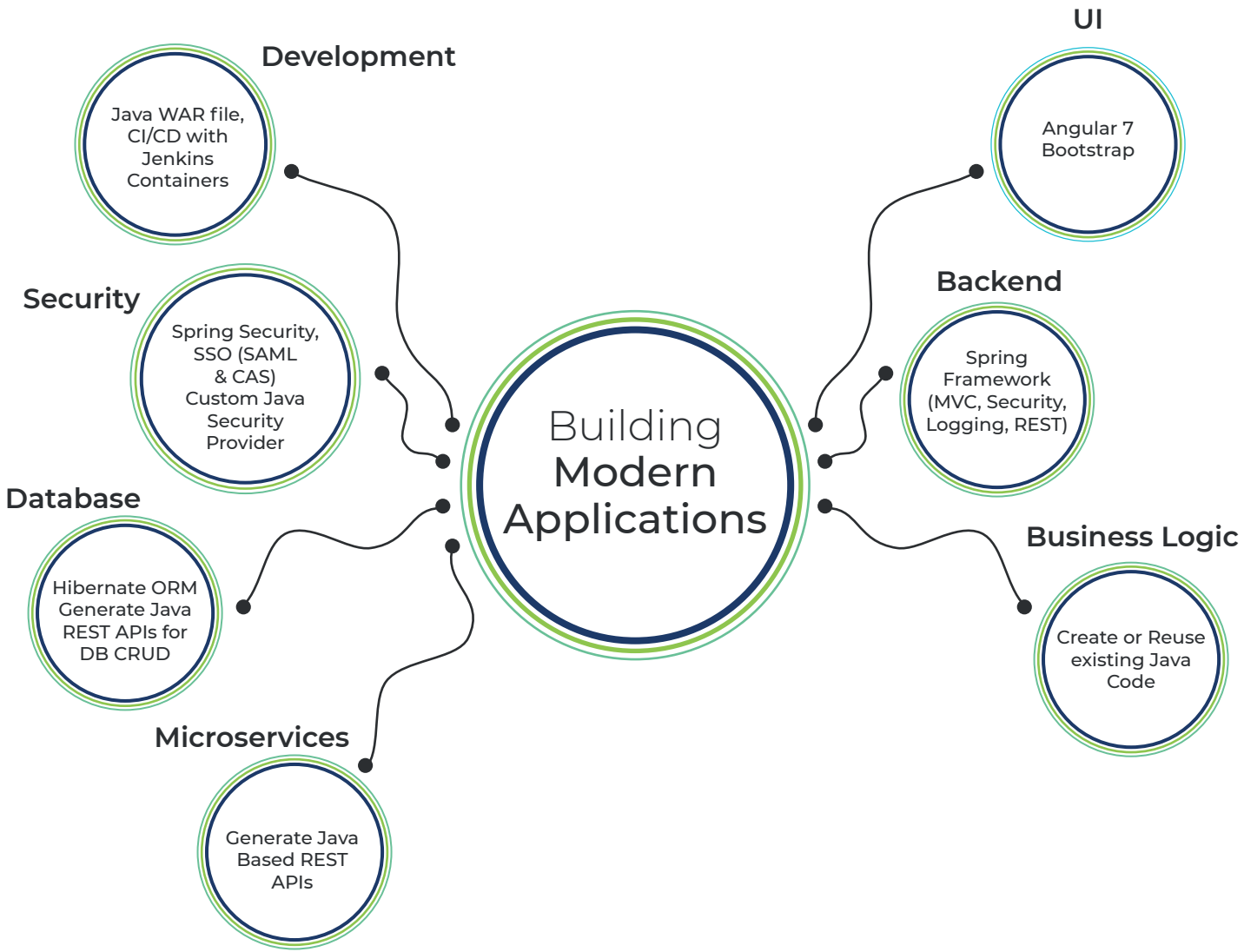
With low-code platforms gaining momentum, the role that traditional Java programming plays cannot be ignored. By combining old with new technologies, here's how the micro and macro levels of application development can be covered to meet new enterprise demands.

How combining traditional Java development with low-code can help meet new enterprise demands

Challenge	Benefit
Developing complex enterprise applications require extensive manual coding.	**Give developers the bandwidth to innovate** • Let developers focus on critical specifications to make the application rich. • Create core applications rapidly using visual development with low-code platforms. Adobe JRun, Resin.
Enterprises need applications to run on different operating systems and devices involving intensive manual coding effort by Java developers.	**Build apps that support multiple device ecosystems and operating systems** • Easily duplicate core functions and create modern applications that work in dynamic environments with low-code. • Develop applications using different languages and easily write code for specific parts of the application that have distinct functionality.
Technical requirements are specifications of IT hardware and software constantly changing, making app maintenance tedious.	**Ease the application maintenance process** • Make app maintenance easier for Java develop ment teams • Using visual development interfaces and modular components, easily maintain applications in the backend and focus on critical functions with low-code.
Before deployment, every line of code needs to be tested in a lab environment which requires intensive manual programming, making the process complex and time-consuming.	**Streamline the application deployment process** • With pre-built modules in low-code platforms, test app functionality before their release. • With actual development and production environ ments accessible in cloud ecosystems, low-code makes app testing and deployment simpler and faster.

How the Java Ecosystem can be adopted across the entire app development lifecycle

Many Java-based enterprise applications lack modern UI. Enterprise application development teams use a combination of technologies to modernize legacy Java-based systems



To optimally utilize the skills and experience of development resources, enterprises need to combine the technical strength of traditional systems with the speed, agility and scalability that modern technologies offer. By combining the old with new technologies, not only can applications be developed, maintained and deployed faster, rich applications can also be created by allowing traditional development teams to give the required attention to detail.

In a market environment where change is constant and the future is uncertain, future-proofing seems to be the safest way forward. Stay relevant to stay ahead by upgrading your legacy systems with a modern low-code platform.

An ideal low-code platform...

...Promises accelerated application development

...Offers a visual development environment

...Enables auto-generation of code

...Supports code-customization with 2-way workplace sync with IDEs

...Ensures extensibility and re-use with prefabs

...Allows for full integration with CI/CD pipelines

Future-Proof the Developer Workforce by Upskilling Java Development Teams

Mobile and web application development have become the protagonists of digital transformation stories. To accelerate modernization initiatives and upskill existing development teams, companies are increasingly embracing low-code platforms.

Full stack developers are involved in end-to-end development of an application. Upskilling to transform your Java teams to become full stack developers is one of the ways to meet the demands of the growing market and reduce the skills gap. Here's how low-code platforms can make full stack app development simpler and faster:

Devlopment Aspect	Traditional Approach	WaveMaker low-code Approach
To create responsive and rich UI/UX	Requires highly skilled frontend technologist (HTML5, Bootstrap, Angular, UI Design) JavaFX.	Simple drag and drop out-of-box Angular-based responsive UI with minimal coding.
To create business logic and APIs	Requires skilled Java and API developersAdobe JRun, Resin.	Auto-create microservices with database logic, reuse existing Java code, or create new Java code in IDE such as Eclipse.
To create database middleware logic	Create database logic using ORM tools and map them to application logic. Studio and Kotlin.	Auto-create ORM layer using ORM Tools such as Hibernate and create logical data models that can be used within application UI.
Deployment options	Create deployment scripts to deploy apps to Java application server and create CI/CD hooks for continuous delivery.	Application artifact, a WAR file, can be deployed to any Java application server. Supports container as well as VM-based deployments for OnPrem and Public Cloud environments.

How low-code helps Java development teams **go the extra mile**

- **Developers can work faster and better** – By provide access to custom JavaScript and Java integration, low-code is a a tool that seasoned developers can use to increase their bandwidth and focus on tasks that can deliver more value.

- **Low-code runs on stacks that developers are familiar with** - Low-code platforms run on top of familiar cloud computing services (such as AWS). Developers can compile visual code to native Java or .NET. To execute drag-and-drop code deployments they don't need to rely on proprietary visual runtime environments. This allows developers to append their existing server scripts to their low-code deployments.

- **Supports full stack development** - With low-code you can access graphical user interfaces, open source code libraries, inbuilt templates, layouts, themes and widgets; granular role-based access control, rich standard APIs, and one-click testing and deployment to develop and deploy modern, enterprise applications.

5.

The Future of Java and How it is Evolving

Companies today are looking to deploy applications rapidly, easily and with agility. Using cloud-native technologies empowers enterprises to optimize their application development and lifecycle management, scale applications to meet demands and utilize resources optimally. The new focus of the Java ecosystem is cloud computing.

Java Enterprise Edition (Java EE) was introduced with the aim of providing complete flexibility and control, and accelerating application development in the cloud. Over the decades Java EE has been the platform choice of enterprise applications as the demand for scalable, cloud-ready SaaS applications increases. With the evolution of technology there are changes in architectural styles and development teams are faced with the challenge of finding core Java developers. Low-code platforms have emerged as a tech-driven solution and empower Java development teams by:

- **Offering advanced feature**
  - The platform can be customized by adding plugins.
  - JavaScript can be manipulated using auto-generated scripts.
  - The frontend can be customised with JS plugins .
  - Backend logic is based on Java Services.
  - Real-time processes and data can be integrated and made accessible.
- **Providing a lightweight, resource-optimized environment -** The platform supports application scaling extensively by using cloud-native architecture based on Docker containerization models.
- **Empowering development teams -** Build portable, cloud-native applications rapidly with minimal coding effort. Developers can focus more on the business logic and address the high-speed demands of the enterprise. scaling extensively by using cloud-native architecture based on Docker containerization models.

Java EE evolved to Jakarta EE

In September 2019, a massive endeavour was undertaken to rebrand Java EE. With the effort

- 18 member organizations
- 43 projects
- 61+ million lines of code
- 126 git repositories
- 160+ committers

By providing a lightweight, resource-optimized environment, low-code platforms using Jakarta EE are evolving rapidly. They support application scaling extensively by using cloud-native architecture based on Docker containerization models. Most importantly, they empower developers to build portablecloud-native applications rapidly, enabling them to keep up with the high-speed demands of the enterprise.

Conclusion

Java has much more to offer to empower IT systems. Its robust features support real-world applications across many other areas, from enterprise applications, embedded systems, gaming applications, big data technology, robotics to Artificial Intelligence applications. With the increasing need to develop scalable and secure applications, Java is still a popular choice mainly because it has security features that are tried and tested.

In a future that is uncertain, the new normal is constant change. Where future-proofing seems to be the ideal way forward, staying ahead by upgrading your legacy systems must take precedence. While this may translate to reinvention in the way of working and the use of emerging technologies, it need not mean reinvention in every aspect of the business. Instead of discarding old systems and skills that have stood the test of time, upgrading systems and updating skills can help to meet new enterprise demands with old tools and can result in easier implementation and better ROI on IT investment. The best way to build a future-proof business is to create an IT ecosystem that has a robust foundation, one that drives mobility, scalability, and flexibility to enable enterprises to embrace a ‘digital first’ approach.