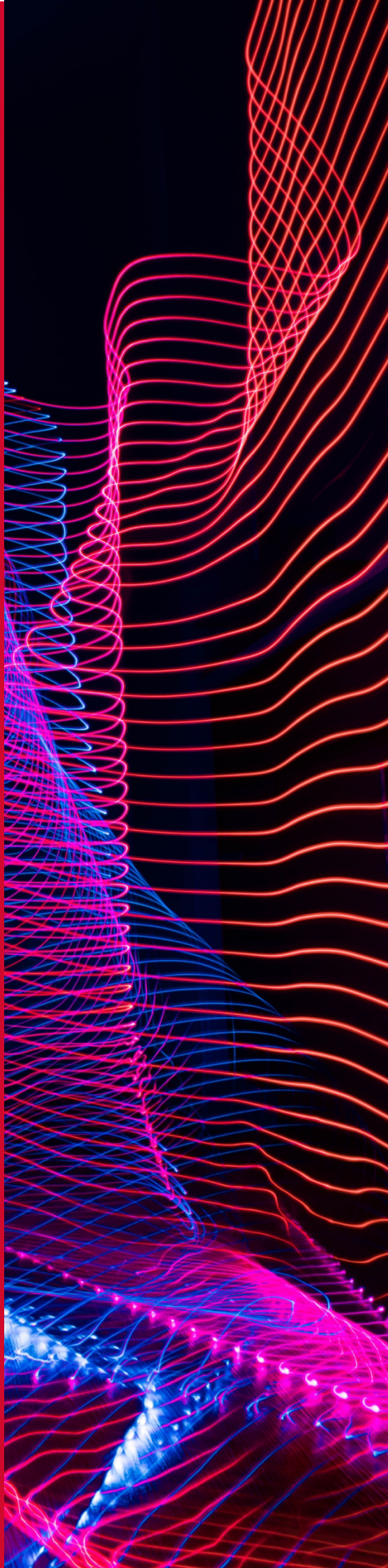
responsiv

simple · effective · distinctive

POINT OF VIEW

The WebSphere Revolution



Standing on the Shoulders of Giants

It's easy to forget in enterprise computing, how short the journey has been and where it all started. The IBM WebSphere story is a classic example of progress accelerating at warp speed as development teams absorbed the achievements of their predecessors and built new applications combining lessons learned with a rapidly evolving vision.

Timeline

- **1989** http and html invented
- **1990** Web Server and Browser
- **1993** IBM MQ Series launched
- **1994** Java invented by Sun Microsystems
- **1995** Apache http Server
- **1998 WebSphere Application Server** implements a Servlet Engine
- **1999** Tomcat leverages http and Java Servlet Container
- **2000 WebSphere** 3.5 Released with support for Java Beans and CORBA
- **2002** Richard Whyte, CEO assembles the MQ Services team at IBM including Malcolm Warwick, CTO
- **2009 WebSphere Message Broker** v7 released
- **2010 WebSphere ESB**
- **2013 WebSphere Integration Bus,** included and replaced functionality of Message Broker and ESB
- **2019** WebSphere Integration Bus is rebranded as **IBM App Connect Enterprise**

In the Beginning…

Believe it or not, there was a time before WebSphere. It's a measure of how successful IBM has been that we forget the extraordinary achievements that have brought us from IBM WebSphere Application Server (WAS) in 1998 to IBM App Connect Enterprise (ACE) in 2019.

The Internet

None of this would have happened were it not for Tim Berners Lee and his team, working at CERN , the European Particle Physics Laboratory in 1989 inventing a set of protocols to enable his team to share research data across computer systems via existing networking technology. The Protocol was **http** (hypertext transfer protocol), the key that unlocked the door was the concept of the **URI** (Universal Resource Identifier) and the language that ensured the data would be readable was **HTML** (Hypertext Markup Language).

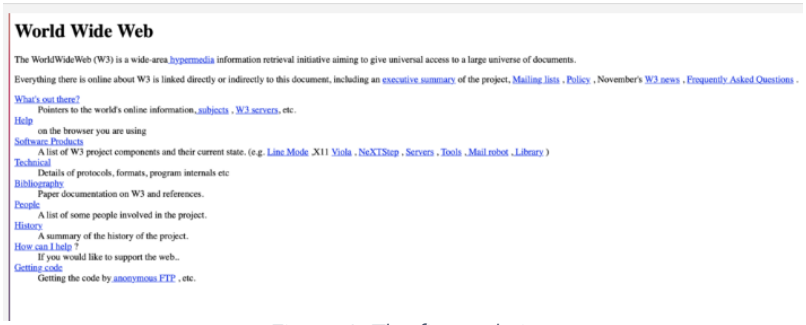


Figure 1; The first website

The first web server and primitive browser were written by Berners Lee in 1990. This, although it did not involve IBM was the moment the WebSphere revolution began. The first website was a simple text-based page with hyperlinks. It opened the door to everything we know today and literally changed the world.

Apache

The Apache project is an open-source project that delivered an **HTTP Server** in 1995.

Take a step back and consider this. Tim Berners Lee set the ball rolling in 1990. It took a further five years for a standardised reliable, open-source web server to appear. If this was the first step, only five steps later we have IBM App Connect Enterprise!

Apache is still the world's most widely used web server. And in 1998 Apache went into partnership with IBM to further develop the platform. It has since provided the core technology for IBM HTTP Server which of course is the http server bundled with IBM WebSphere.

The transfer of static text across the internet was a huge step but one that begged a few questions.

- What if we could change that content dynamically?
- What if we could have variables in html that extract data from an external application in real-time?
- What if we could write data back to the application from the web server?
- How do we do this in a platform-independent way?

Initially, languages such as Perl were utilised to leverage what was called CGI – The Common Gateway Interface. This technique became popular in the Unix community but was inaccessible to ordinary mortals. The code was impenetrable and CGI was slow. It was hard to cache session-specific variables between page loads. It was just difficult.

Java

Java was a programming language developed by Sun Microsystems in 1994 just four years after Tim Berners Lee invented http. The innovation behind Java was aimed at solving the four questions:

- How do we deal with the demand for dynamic content?
- How do we talk to external applications?
- How do we write data to external applications?
- How do we deal with platform-specific requirements?

The major innovation brought to the table by Java was the JSDK specification for a platform-independent "container". The idea was to provide a black box environment that would interpret Java code and run it regardless of platform.

Java had several advantages over Perl - it was easier to learn, read and write, it was portable and sessions were maintained separately by the container.

Tomcat

Apache Tomcat was an extension of the http server project leveraging the idea developed by Sun Microsystems of a standalone container that would interpret the Java language and maintain separate sessions per user.

Initially, in 1999 Tomcat implemented the JSDK spec for a servlet container. We now had a system that could serve static pages from the http server and dynamic pages from the servlet container. The output from the servlet container was html.

Within a few months, Tomcat provided support for Java Server Pages (JSP) which were a kind of inverted servlet - pages written in html containing snippets of code that call back to the servlet container. JSP pages and servlets could also read in data from forms on the web pages.

These developments spend the door for all the e-commerce applications we see today.

Sun on its own wanted to make the JSDK a paid-for offering but failed to get traction. They later offered it to Apache in order to leverage Apache's position as the de facto web server of the day. What followed was explosive.

WebSphere Application Server

Although WebSphere Application Server started life as a servlet container it didn't gain much traction until version 3.5 which added support for Java Beans and the Common Object Request Broker Architecture, (CORBA) in 2000.

This and the emerging J2EE standard that included Enterprise Java Beans is the gold standard by which other Application Servers are judged. The enterprise software industry was up and running. Microsoft was confined to the office, Oracle and IBM were the new normal in what became known as Enterprise Computing.

The addition of Java Beans was key. These "beans" were data structures encapsulated in code that gave programmers the ability to read data in and out of databases in near real-time, responding to an end-user sitting a thousand miles away at their computer.

The IBM WebSphere Vision

The vision IBM had for WebSphere was nothing less than revolutionary. The idea was to transform businesses with cutting-edge technology, creating a new market for IBM Services and Software Group partners. Only IBM had the structures already in place to pull this audacious manoeuvre off. Other companies could only watch.

IBM Research - Labs full of brilliant researchers working at the bleeding edge of the technology.

A **Software Development Team** working with cutting-edge technology producing IBM branded products.

Dedicated Business Partners to roll out specialist services and education at pace.

A homegrown **Consulting Organisation** to advise on business transformation.

The global economy at this point had just recovered from a serious recession and the appetite in the business community for risk and radical improvement was immense. The time was right.

Transform Business

IBM Messaging technologies had been making their impact felt in the financial sector since 1993 when MQ Series was released. The vision for business in the wider context was to combine this transaction-oriented technology with the newfound ability to operate distributed architectures using internet protocols.

In 2002 MQ series was brought into the WebSphere family and **Responsiv CEO Richard Whyte** who was working for IBM Software Group at that time put together the team that would later become Responsiv Solutions. The team quickly made a name for themselves in the financial industry and as a recognised thought leader, Richard rose to become European CTO and Principal Architect for IBM Systems Middleware before leaving to start Responsiv Solutions.

Enabling the Enterprise

WebSphere Application Server was just the beginning, the pace of development was breath-taking. Oracle acquired Sun Microsystems and started to provide real competition. IBM concentrated on developing a suite of enterprise software and mission-critical computing systems.

The Java language soon generated a large ecosystem of developers, in part because Universities had recognised both the industrial potential and the suitability of Java as a vehicle for teaching Object-Oriented Programming.

This meant that in-house development teams were quick to pick up on the possibilities of the new enterprise computing model and businesses with one eye on the global economy and the other on e-commerce began to experience the growth that the technology enabled.

Now that the platform was there, the more visionary companies began to build out their vision on top of the giant steps that had been taken between 1990 and 2000.

We would see the integration of business process modelling, the abstraction of services into an **Enterprise Service Bus** Architecture, and ultimately the introduction of **Artificial Intelligence** to turbocharge growth into the 2020s.

Connecting the Enterprise

The WebSphere Application Server enabled the enterprise to promote browser-based applications that allowed customers to transact from any place at any time. The same principles applied to booking a holiday as applied to buying and selling stocks and shares. Companies like Amazon built huge fortunes on the back of the new technology. So rapidly did the internet boom that it became possible for a company specialising simply in identifying and locating URI's to thrive. **Enter Google!**

MQ Series

MQ Series came from a different strand of research. It was not originally designed for Java, rather it focused on high volume reliable messaging between known endpoints.

MQ transports any type of data, wrapped as messages. This capability enables businesses to build flexible, reusable architectures quickly and it is core to the Enterprise Service Bus products.

Given its flexibility and emphasis on reliability and security, MQ Series was taken up quickly by financial institutions and expanded its user base from there as other industries became aware of its potential.

IBM wrapped it into its WebSphere family with the introduction of WebSphere Message Broker.

Application Infrastructure

There were still bottlenecks in the system. Java, as an interpreted language is slow, compared to compiled languages. The weight of processing in the new systems, although an improvement on CGI tended to slow the systems down. The introduction of BPM into the equation showed a possible route through the problems.

Application Integration

For 99% of e-commerce transactions, speeds of under a second are not readily discernible by human beings, and the latency of the internet itself, especially once mobile phones became the platform of choice for around 50% of the internet-based transactions, makes sub second responses unenforceable.

This meant that the focus moved away from the browser-server architecture towards application integration and business process integration.

Once business process integration had been achieved, the automation of business processes was added to the automation of the manufacturing process to take time out of the fulfilment cycle at every level.

Buying an insurance policy became a matter of uploading verifiable information through a web browser and having the policy confirmed within minutes. This is a far cry from the processes carried out by the local insurance broker of yesterday, where documentation was hand-checked, policies were laboriously compared and new insurance was only in place after several days had elapsed.

There was still an outstanding question though - all of this good stuff still relied on reliable end-to-end connectivity, connections that would be guaranteed and secure. What if this connectivity could be abstracted and automated?

Enter the **Enterprise Service Bus**

The WebSphere Enterprise Service Bus was a very clever piece of engineering indeed. Like all good movie scripts, it can be summarised in a couple of lines.

WebSphere ESB is essentially the implementation of a pattern where a software component is given the responsibility to moderate integration to back-end systems and make those integration points available as services that existing, new, and forthcoming applications can locate and plug into.

Under the covers this is complicated stuff, drawing on **Service Oriented Architecture**, requiring at minimum, translations of data models, flexible and reliable connectivity, routing, and request management. All the while presenting a simple interface that developers can readily plug into.

Where message broker provided a universal adapter, ESB extends and builds on that technology to provide an adapter for all applications on demand. The fact that this integration wizardry runs on IBM WebSphere Software is a bonus.

Service-Oriented Architecture

A Service-Oriented Architecture is one that instead of promoting point-to-point connectivity between applications, uses the concept of a broker to mediate connectivity. Connections to the broker are designed for reuse and very broadly an implementation of a service-oriented architecture acts as an adapter/hub where applications can "advertise" services internally and create connections on demand.

This style of architecture has become popular because it simplifies some of the hardest things programmers were being asked to do. Security, Routing, Message transformation to name but three. Programmers using this architecture are typically able to spend more productive time dedicated to business function and business process management.

The API Economy

Since 2020 organisations have realised and begun to act on the principle that APIs are the key to accessing and being accessed by the type of technology we are discussing here.

Where organisations used to diverge on standards and APIs in order to create exclusivity and business advantage, the introduction of standard-based architectures has tipped the balance to a point where jealously guarded secrets are inhibiting growth and opportunity for companies.

IBM App Connect Enterprise (ACE)

The latest incarnation of the WebSphere evolution is IBM App Connect Enterprise (ACE). In a nutshell, ACE combines all of the capabilities we've discussed in this article, implements a robust service-oriented architecture, and adds Cloud services to the mix.

Let's take a closer look.

IBM ACE is an integration broker that moderates the flow of business information across multiple applications and environments. Rules are dynamically applied to the data to route and transform the information as it flows.

On top of this rich set of capabilities, the latest release of IBM ACE supports cloud deployment. Hosting the runtime in the cloud compares well to hosting the runtime on-premises because it simplifies the maintenance and application of OS-level patches and allows scalable resources to be deployed in response to demand.

Another step forward for IBM is the support offered for .NET technology and built in support for a variety of web standard protocols and languages including XML, JSON, Java, PHP, JMS, and custom integration points for SAP, PeopleSoft, and JD Edwards message nodes.

Responsiv Solutions and IBM ACE

As we mentioned earlier, the Responsiv team was individually and collectively involved with IBM from early 2002 and have literally worked with this technology since the very beginning. We don't like to blow our own trumpet too loudly but there is no other company in the UK that can boast the same level of experience and success in helping enterprises improve their business performance by taking the complexity out of the development cycle and enabling development teams to focus on business benefit.

Our success in helping businesses to adapt IT infrastructure and much-loved applications to the increasingly abstract world of services, on-demand computing, and message mediation is based on three core principles. Simplicity, Effectiveness, and Distinction.

IBM ACE Discovery Workshop

If you have questions about IBM ACE or want to know how Responsiv can help your organisation, get in touch and one of our consultants will give you a call. No obligation, no frills. We make excellent coffee too!

FAQs

What is IBM ACE?

IBM App Connect Enterprise *is an Enterprise Service Bus with Messaging capability that can be deployed on-premise or in the IBM Cloud.*

What are Microservices?

Microservices extend the ideas behind service-oriented architectures to deliver sets of loosely coupled services that aggregate together provide application functionality. Where once you may have designed an application form end to end as a single monolithic entity, using a microservices architecture the functionality is broken out into reusable services.

What is a Container?

A Container is a processing environment used by WebSphere to package application code, libraries, and dependencies so that it can be run anywhere. The earliest example was the JSDK Servlet Container.

Is IBM ACE an ESB?

Yes. At heart, IBM ACE is an Enterprise Service Bus but with Messaging capability.