

The Application Server and SOA

The characteristics to look for in an app server

by Larry Cable

Over the past few years the industry has lauded, and users have increasingly adopted, a Service Oriented Architecture (SOA) (http://en.wikipedia.org/wiki/Service-oriented_architecture) approach to the development and deployment of their IT to achieve greater business agility and optimization of the associated development and operating costs.

When discussing SOA, much of that discussion normally focuses on either the role and value of a "governance" (http://en.wikipedia.org/wiki/SOA_Governance) process to manage the lifecycle of a SOA deployment and the use of a metadata repository to contain and control the artifacts associated with that governance process, or the central role that an enterprise service bus (http://en.wikipedia.org/wiki/Enterprise_service_bus) plays in the transformations between, and the routing of invocations, of services.

While a governance process and an enterprise service bus may both distinguish SOA from other architectural approaches and be crucial in SOAs success, the focus on these distinguishing and valuable characteristics of SOA seem to overshadow all else — so much so that one could be forgiven for forgetting the "services" ([http://en.wikipedia.org/wiki/Service_\(systems_architecture\)](http://en.wikipedia.org/wiki/Service_(systems_architecture))) themselves, and the crucial role and capabilities of the container(s) and runtime(s) that host these service instances in realizing a successful, enterprise-scale, mission-critical SOA deployment.

This article, in part, attempts to redress that imbalance by discussing the role and characteristics of service container(s) and runtime(s) — in particular Java application servers (http://en.wikipedia.org/wiki/Application_server) and how they can contribute to the deployment of enterprise-class, mission-critical SOA.

Perhaps it's because, unlike many of its predecessors, SOA is technology-agnostic (i.e., it's more of a meta-architecture approach), that it's growing in adoption. The fact that you can create an SOA deployment using .NET, Java, CORBA, Message Oriented Middleware (MOM), DCE, DCOM, Web Services (based on REST (<http://en.wikipedia.org/wiki/REST>) or SOAP (<http://en.wikipedia.org/wiki/SOAP>)), or a host of other technologies, is its core strength. This diversity can, however, also result in a key weakness in creating a successful deployment because of the diverse requirements imposed on the runtimes or containers that host particular services.

In beginning to consider the role and responsibilities of

service containers or runtimes, it's worth noting a key distinction between developing and deploying an application versus a service.

Applications are typically self-contained black boxes, the internal implementations of which can be changed, sometimes, radically, without impact on their consumers or users; while the internal implementation of a service can also change radically without impact on its consumers.

Where a service differs from an application is that a service can't typically anticipate its adoption, reuse, or composition. So, when developing and deploying a service, it's crucial to realize that you're in fact defining both a platform (API) and providing a service (quality of service/service level agreement) either implicitly or explicitly.

What are the implications of this? When choosing a programming model to implement a particular service the developer should consider the ability of that model to adequately describe a reusable, broadly consumable, and, most critically, an evolvable service interface definition (IDL) — one that's separately evolvable from its implementation.

To illustrate this, let's consider the CORBA IDL (http://en.wikipedia.org/wiki/Common_Object_Request_Broker_Architecture) and Web Services Definition Language (http://en.wikipedia.org/wiki/Web_Services_Description_Language).

While both these IDLs are abstracted from their target implementation languages, it can be argued that WSDL is significantly better than CORBA IDL at completely decoupling the interface from the implementation and so allows both the implementation and interface to evolve somewhat independently, and also, most significantly, to be evolved and deployed alongside previous versions of a particular definition or schema to allow clients or consumers of all versions of the interface to continue to function.

Ignore this capability at your peril — if you evolve your service implementation and/or interface without taking care of backward compatibility with your current consumers, the results could create significant difficulties.

So selecting a container or runtime that supports a broad set of programming models, their associated IDLs (if any), and the ability to host multiple versions of a particular service implementation and interface is a key.

The second distinction between an application and a service is the relative unpredictability of the demands on

a service versus an application over its lifetime. As an SOA deployment matures over time, the most critical or useful services in that deployment will be increasingly reused, often in ways the original developer or deployer never contemplated.

Therefore the ability of the container and/or runtime hosting a particular service instance (or instances) to scale to meet highly variable demands from their clients/consumers is key to creating a mission-critical SOA deployment. Because of the inherent distributed nature of SOA-style applications, a particular composite application is only as robust and scalable as its least scalable and robust component. This may seem obvious, but it's often overlooked (with disastrous results) when developing and deploying composite applications.

So, in view of the critical impact of service instance runtimes or containers on the ultimate success of the applications that may consume them, what characteristics should we look for in an application server?

1. Programming Model(s):

Containers that implement Java Enterprise Edition 5 specifications (and other emerging technologies) are excellent hosting platforms for SOA-style services since they support a variety of key APIs and technologies that can be used to develop and deploy either individual services or applications composed from services, including:

- **JAX-WS:**

Probably the key API for Java as a hosting platform for SOA services is JAX-WS; the second generation of SOAP Web Services APIs for the Java Platform JAX-WS provides the necessary abstractions for important protocols such as WS-Policy, WS-SecurityPolicy, WS-Addressing, WS-ReliableMessaging, WS-SecureConversation, SAML, WS-MEX, and other WS-I protocols to enable developers to produce WS-I Basic Profile 1.1-, 1.2-, or Reliable Secure Profile V1.0- (<http://www.ws-i.org/>) compliant (and interoperable) service endpoints, and thus create Web Services-based SOA services that can be interoperably combined with other services from either an ESB, another component, or a composite application.

- **JAX-B:**

The Java API for XML Binding complements JAX-WS to enable Java developers to bind POJOs to XML schemas and in particular XML documents that may be exchanged via Web Services by decoupling or automating the mapping from XML to Java and the reverse mapping JAX-B enables greater abstraction and independence from the XML payload exchanged over either Web Services or other protocols.

- **JMS:**

JMS is a key API for SOA deployments. While a lot of focus is put on Web Services as a fundamental communications protocol for composing services, many existing IT operations already have an "informal" SOA-

like architecture built around either point-to-point or publish-and-subscribe-style MOM deployments. So the ability to produce or consume such message-based exchanges is fundamental to extending them to new SOA deployments.

- **EJB:**

While not immediately apparent as a model for services, EJB is fundamentally valuable in encapsulating business logic around relational data, when either can be directly consumed as a service via RMI/IIOP interfaces (perhaps through an SCA assembly, or directly invoked remotely) or more likely indirectly either through Web Services or MOM via JMS and message-driven Beans.

- **JAX-RS (JSR 311, currently under development):**

Currently in the later stages of development and specification, support for JAX-RS, JSR 311, ([http://www.jcp.org/...](http://www.jcp.org/)) will be an important API standard to enable developers to expose REST-style Web Services; REST is an increasingly important distributed computing model for B2B, and B2C Web 2.0 client/server models such as AJAX and COMET Rich Internet Applications.

- **Service Component Architecture:**

The emerging set of Service Component Architecture specifications (<http://www.osoa.org/display/Main/Service+Component+Architecture+Home>) define a number of key standards to directly address some of the SOA-related assembly issues that other preceding standards do not. In particular the Java Component Specification describes how simple POJOs can be encapsulated in an SCA runtime in an application server, managed, and made available for composition by an SCA Assembly or application.

- **JavaIDL:**

While being one of the less popular APIs in the Java EE standard, the existence of a CORBA solution is still valuable when creating new SOA services out of legacy CORBA componentry.

2. Enterprise Information Systems Integration: (suggest Connectivity with Packaged Apps as a title)

Java application servers are often referred to as middle-ware, because their role in the construction of applications or services is to extend and expose existing enterprise information systems, applications, and data that is otherwise inaccessible or siloed.

Since most SOA deployments occur virally in the context of an existing mature IT infrastructure, the ability of an application server to encapsulate existing IT assets and recast them as composable services is a key enabler.

The Java Connector Architecture describes technology that prescribes connectivity to external systems through connectors and thus allows new services to leverage packaged applications and systems for implementing units of work or for pulling data these existing systems for greater reuse in a SOA deployment.

“Having a container that can deliver a rich development environment and a reliable, scalable, secure, manageable runtime environment is key to a successful enterprise-class SOA deployment, functions a superior Java application server is ready-made to do”



Larry Cable is an architect with the Java Platform Group within the Oracle Fusion Middleware organization. His focus is on Oracle WebLogic Application Server and related (Java-based) standards technologies. Prior to joining Oracle, Larry was chief architect for BEA's WebLogic Server and the WebLogic brand architecture. He was also corporate architect within the Corporate Architecture Group, as part of the Office of the CTO at BEA. Larry studied Computer Science in Scotland; in Edinburgh and St Andrews, but in the ensuing years has forgotten much of what he was taught there and often looks at his high school math textbooks in awe.

3. Developer Tools Integration:

Perhaps still in their infancy, good developer tools to support both the creation of individual service implementations, to produce and consume SOA artifacts stored in a metadata repository, and to aid in the assembly, deployment, and debugging of SOA services and applications are required to accelerate the development of such services and applications. Tool environments such as those being developed for the Eclipse platform are a strong beginning in providing such an environment that is well integrated with many Java application servers.

4. Quality of Service:

Although not formally part of any standard, key distinguishing features of a container or runtime that can provide facilities to hosting mission-critical SOA services in an enterprise deployment are the quality-of-service characteristics, in particular:

- **Reliability:**

As mentioned before, a SOA application is only as robust as its weakest component service, thus service reliability is crucial for mission-critical applications. While services can to some degree implement their own reliability mechanisms, having a hosting container that can provide facilities to maintain state over individual service failures can greatly increase the robustness and reliability of applications consuming them.

Features such as automatic state replication of service instance data and transparent failover of protocol requests between hosting containers can transparently isolate applications from individual failures of services.

- **High Availability:**

Earlier in this article we discussed the importance of quality of service for services in a successful enterprise SOA deployment. Once a particular service becomes available and, as it's increasingly consumed by other services and/or applications, the demands for that service to be constantly available will increase. As a result it's crucial that the runtime hosting environment, and even the service itself, be highly available. So the runtime and the service must be maintained and updated without interruption.

Additionally, if the container supports multiple versions of the service or application executing side-by-side, this greatly aids in migrating clients or consumers from earlier versions of the service to newer ones seamlessly.

- **Scalability:**

Another key aspect of quality of service is service scalability. As SOA deployments mature and applications increase their demands on service implementations, continued normal and exceptional operations require services to respond to dynamic changes in load/demand. While this can be achieved ad hoc at the service level, a runtime or container that can automatically scale to add or remove additional compute resources to meet invocations on service instances, to balance loads between them, and to provide transparent failover between instances when failures occur greatly improves service and application performance and reliability.

- **Security:**

One of the key issues in making enterprise SOA deployments successful is distributed security — not just securing sensitive information across networks (mes-

sage encryption, digital signatures, etc.) but also, and perhaps more importantly, distributed and interoperable authentication and authorization. Again, while this can be implemented in an ad hoc fashion at the services instance level, a container or runtime that provides integrated security services is highly desirable, not only to reduce the burden on service and application developers but to take responsibility for the overall security and integrity of a SOA deployment from the individual developer and put it in the infrastructure.

So a container or runtime that integrates well with network authorization and authentication, policy, credential, and certificate management systems is highly desirable. Support for multiple identity or credential representation mechanisms is particularly crucial for true secure interoperability between distributed heterogeneous service containers.

- **Transactions:**

While SOA doesn't require transaction processing capability, many services themselves may need to participate in either local or global transaction coordination to maintain data/process integrity. Having a container or runtime that supports both transaction models and integrates with industry-standard transaction protocols such as XA is highly desirable to host such services and enable SOA applications or assemblies to interact with transactional systems and data.

5. Operational Management and Administration:

Operational management and administration is often where commercial application server products distinguish themselves from their open source brethren and where such differentiation can make a difference as an SOA deployment evolves from its initial stages into a mature mission-critical component of a business' IT operations.

The administration and management of SOA deployments is very much in its infancy today. Little exists that enables the management, monitoring, and administration of composite SOA applications, although the SCA specifications in the areas of assembly and policy models are beginning to address the management of distributed composites. However the integration between application servers hosting Web Services endpoints, and Web Services management solutions does enable the end-to-end management, monitoring, and policy enforcement on such services.

In the interim, the facilities that many application servers provide, such as JMX, SNMP, and emerging Web Services-based management protocols can be used in an ad hoc fashion to manage and monitor individual SOA services and applications deployed therein, allowing IT operations departments to do end-to-end management of SOA deployments.

In summary, since SOA is all about creating a rich portfolio of reusable services to enable rapid construction of new applications to meet the needs of business and provide those new applications more economically, having a container that can deliver a rich development environment plus a reliable, scalable, secure, and manageable runtime environment is key to creating a successful enterprise-class deployment. A superior Java application server is ready-made to meet these needs. ☉



15TH INTERNATIONAL SOA WORLD CONFERENCE & EXPO 2009 EAST

June 22-23, 2009 New York, NY

16TH INTERNATIONAL SOA WORLD CONFERENCE & EXPO 2009 EUROPE

2009 London, England

Join Us in 2008-2009 and Gain the Insight & Knowledge to
Declare Your Company's SOA Journey a Success!

Service-oriented architectures (SOAs) have evolved over the past few years out of the original vision of loosely coupled web services replacing constrained, stovepiped applications throughout enterprise IT. Every major enterprise technology vendor today has developed its own SOA strategy, supported by innumerable mid-size companies and start-ups offering specific SOA aspects or entire solutions. This explosive growth in SOA technology is in response to a global demand — IDC estimates that spending on SOA services alone will grow from \$8.6 billion to more than \$33 billion by 2010.

SOA World Conference & Expo brings together the best minds in the business for a two-day conference that offers comprehensive coverage of SOA and what it means to enterprise IT today. As ZapThink analyst Jason Bloomberg has noted, "SOA involves rethinking how the business leverages IT in many various ways." Attend SOA World Conference & Expo 2008-2009 and learn from more than 100 speakers

about how SOA is transforming business — and the way IT and business managers think about their businesses, processes, and technology.

The collocation of SOA World Conference & Expo 2008-2009 and Virtualization Conference & Expo delivers the most relevant content to IT and business.

Who Should Attend?

- CEOs and CTOs
- Senior Architects
- Project Managers
- Web Programmers
- Web Designers
- Technology Evangelists
- User Interface Architects
- Companies and Organizations looking to stay in front of the latest Web technologies

REGISTER TODAY AND SAVE

www.soaworld2008.com