

technical paper



# Building Web Services on the J2EE platform

Introduction

Why J2EE for Web Services

J2EE platform support for Web Services

Choosing a J2EE Vendor for Web Services Implementation



## Building Web Services on the J2EE platform

This technical paper is aimed at providing IT Managers, CTOs, and Architects an insight into implementing Web Services using the J2EE™ platform. It explains the suitability of J2EE for adopting Web Services, elucidates the support for Web Services within J2EE, and points to the desirable features in the J2EE tools required for building and running Web Services.

### Introduction

Web Services technology is the next generation architectural model for developing and deploying business applications. Web Services are functional software components that are published, discovered, and invoked across any public network including the Internet. Web Services can communicate with one another over the Internet using a loosely coupled communication model.

Do refer to the white paper “Web Services, An executive Overview” for a briefing on the Web Services.

### Why J2EE for Web Services

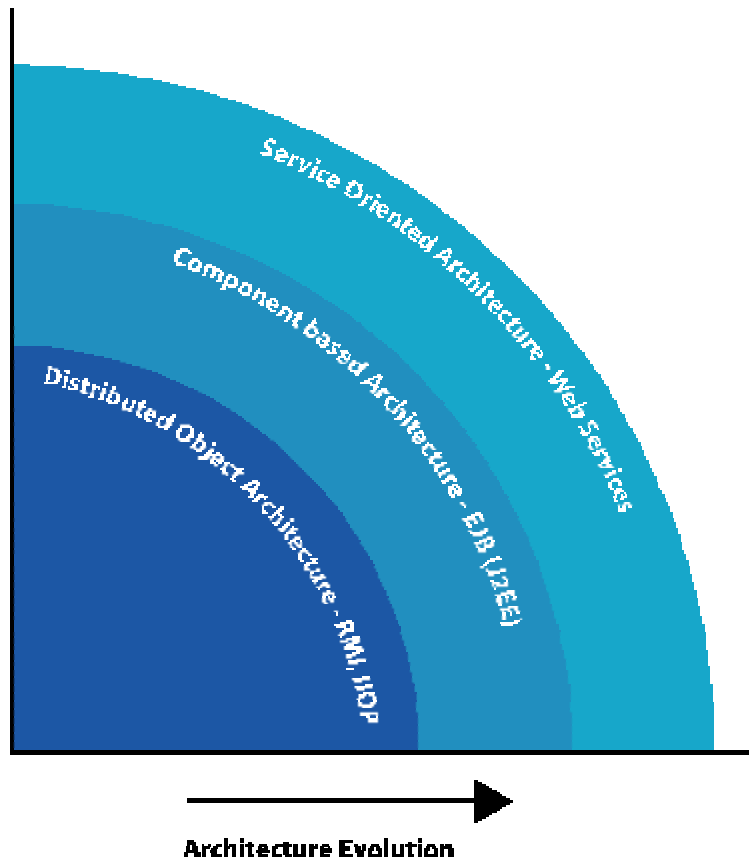
J2EE, the foundation of the modern enterprise-computing environment, is the standard technology and architecture framework enterprises use to build business applications. It enables developers to create simple, reusable components that encapsulate the domain-specific business logic and rely on a standard run-time environment for enterprise-class security, transaction and resource management services. The key characteristics of this revolutionary technology include

- Extending Java’s benefit of ‘Write Once Run Anywhere’
- Platform Neutrality with no vendor lock-in
- Widespread tool support from large vendors
- Availability of skills and strong momentum in the developer community
- Reduced maintenance and development cost through a proven component model
- Ease of development and faster time to market

The white paper ‘J2EE™ 1.3 - The Next Step for Enterprise Architecture’ presents more information on the relevance of J2EE technology for enterprises.

**Web Services – an extension of J2EE**

Web Services is based upon Service Oriented Architecture (SOA) model. Service Oriented Architecture closely follows the Component Architecture Model evolution.



Component Architecture Model provides for functional components to be independently developed, and subsequently used as units of a large system. SOA extends the concept by allowing for the components to be exported onto the network and enabling them to be published, discovered and invoked dynamically by each other across the Internet. SOA enables integration of applications over Internet and provides for loose coupling. The Web Services framework implements the SOA model by being independent of the development platform.

J2EE platform is based upon the component-based architecture. It provides a scalable runtime environment, mechanism to integrate with backend legacy systems, along with declarative configurability of runtime behavior etc. Powered with these features, J2EE forms the ideal development and deployment platform of the core functional components. These functional components can be encapsulated later, to be exposed as Web Services.

#### **Direct support for Web Services**

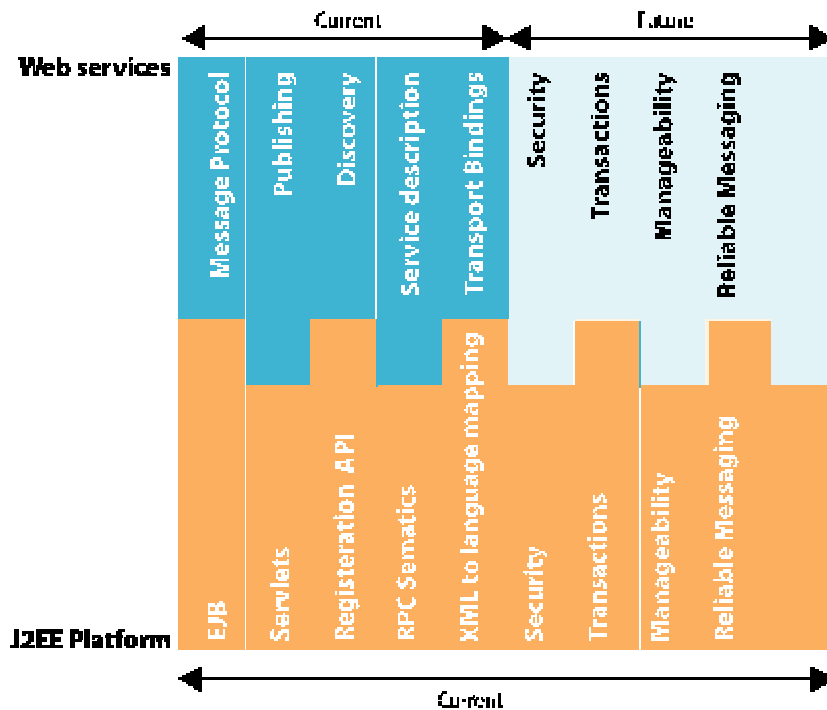
J2EE has rich support for handling XML, which forms the core building block for Web Services. J2EE defines a host of APIs to handle XML Parsing, XML Transformation, XML based communication, XML data formats and XML

registration. This wide ranging technology support along with availability of tools for handling XML provides developers with enhanced flexibility and efficiencies in building real world Web Services. The J2EE standardization process also includes a number of specifications enhancing the support for Web Services. All the specifications enabling Web Services have already been defined, or in advanced stages of evolution. The details of the actual standards supported within J2EE are listed later in this document.

**An evolved platform**

The ubiquitous acceptance of the J2EE platform among enterprises makes it a mature platform for developing mission critical systems. The wide acceptance also provides for a wider range tools for publishing and managing Web Services. All the existing J2EE tool and services vendors have announced support for Web Services along with multitude of new software vendors providing offerings to enable development teams build robust Web Services in a fast and efficient manner.

In addition J2EE has aids for accessing the Legacy application in the form of “Java Connector Architecture”. Hence, for enterprises looking at integration with legacy systems or extending these systems as Web service, J2EE provides the complete infrastructure.



In addition to providing data and code portability, Web Services need to be scalable, secure, and efficient, especially as they evolve. Also, the first generation of Web Services are not likely to provide several “Quality of Service” features including secure and transactional access, in an interoperable fashion. The J2EE platform being

transactional, secure, scalable, reliable and manageable is rightly designed to meet the requirements of developers working with an evolving Web Services technology.

It won't be long before these "Quality of Service" features become an integral part of Web Services, accelerating their adoption while J2EE platform forms an ideal technology framework catalyzing this adoption.

#### **Standards based and Portable**

Similar to the Web Services specification development, J2EE platform specifications are promoted by collaborating organizations with wide industry support. Web Services enable enterprise system using different computing platforms to communicate with each other. This feature makes the Java platform, which makes code portable, the natural choice for developing Web Services. This portability available with J2EE provides a big advantage over the Microsoft's proprietary .NET platform. The industry-accepted standards also result in wide choice in the tool base available for developing and deploying Web Services.

### **J2EE platform support for Web Services**

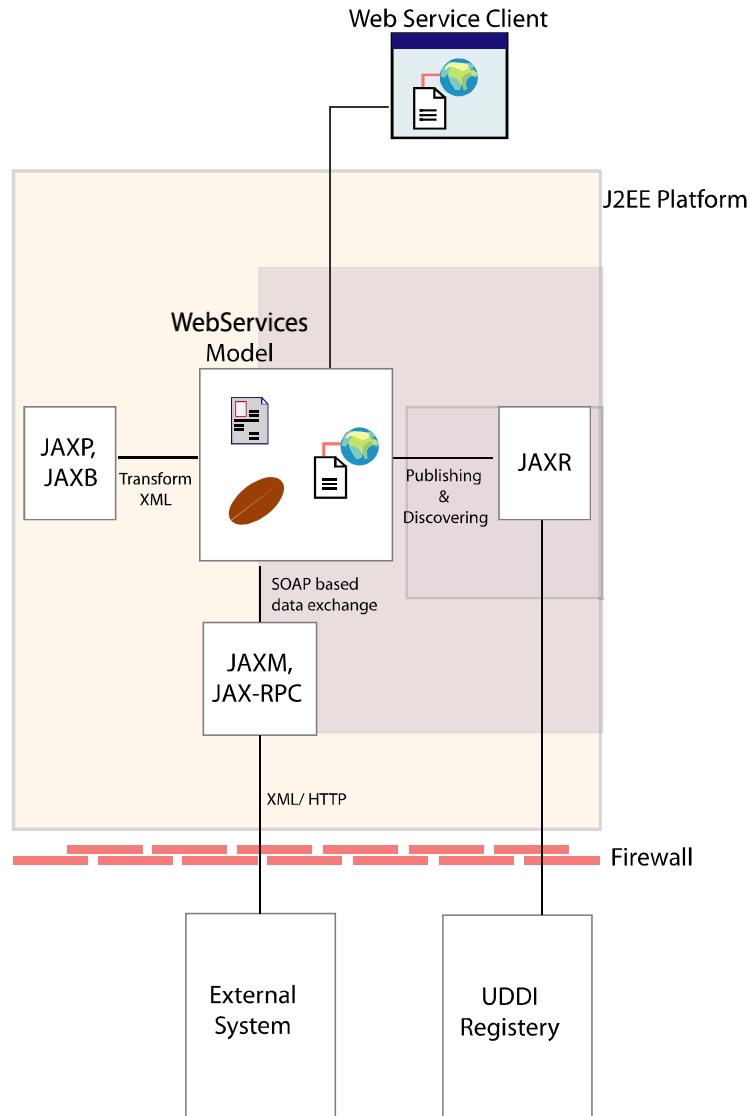
As a part of its initiative to integrate Web Services into the J2EE platform, a large number of specifications supporting the technology are currently evolving. The next release of the J2EE platform specification, J2EE 1.4 provides wide spread support for Web Services. The following section lists standards related to Web services that would become a part of J2EE. Most of these would be a part J2EE 1.4 version, expected to be released at the end CY' 2002.

#### **Web Services Standards**

J2EE platform support for Web Services can be broadly classified into two categories of standards.

- Standards defining Java APIs for handling XML. These standards are analogous to XML Toolkit.
- Standards that defining J2EE as a well-defined programming model for Web Services.

The following figure depicts the utilization of various XML toolkit APIs to provide the Web Services support. The container itself supports the programming model for Web Services.



The various Standards and their benefits along with a J2EE Application server are given below:

### Standards defining Java APIs for handling XML

#### XML parsing

Java API for XML Processing (JAXP) supports parsing of XML documents and transformation. JAXP standard has been in existence for a long time and its latest version (JAXP 1.2) provides support for XML Schema.

This Standard is considered as the most basic standard and a building block for the other Java-XML standards. Even while the J2EE framework handles the XML related activity, JAXP provides flexibility for XML processing to be handled at the application level.

#### SOAP Message handling

The SOAP with Attachments API for Java (SAAJ) provides a mechanism to construct and decipher messages conforming to the SOAP 1.1 specification and SOAP with Attachments note. This is a basic API which could be used by other JAX\* Standard implementations to provide more specific services. This API can be used by the application to construct SOAP messages.

#### XML based RPC

Java API for XML based RPC (JAX-RPC) enables building of web applications and Web Services incorporating RPC functionality as per SOAP 1.1 and WSDL 1.1 specifications, the core specifications forming a part of basic profile of the Web Services.

JAXRPC provides for

- API and conventions for compiling Java classes and interfaces from the WSDL and XML Schema definitions.
- API and conventions for mapping Java classes and interfaces into WSDL and XML Schema definitions.
- API and runtime support for marshalling and unmarshalling arguments inline with SOAP encoding. It also provides runtime support for transmitting and receiving calls in line with SOAP 1.1 specification

The application components in J2EE platform may not directly use this JAX-RPC API. The application server would use JAX-RPC API that would in turn result into calls on the application components. The Web Services clients may use JAX-RPC API directly for communication with the application server.

#### XML based Messaging

The Java API for XML Messaging (JAXM) provides support for writing business applications that perform messaging based on the SOAP1.1 and the specification for SOAP with Attachments. Message exchange scenarios based on JAXM involve exchange of XML documents in synchronous as well as asynchronous fashion. This model of programming may be preferable in some scenarios as compared to XML based communication.

This standard is not mandatory with J2EE 1.4 specification, the specification is final and vendors can be expected to provide JAXM based tools as a part of their offerings.

#### Web Service Registration

The Java API for XML Registries (JAXR) provides a standard Java API for accessing different kinds of Web Services Registries providing XML access. A Web Services registry is an enabling infrastructure for building, deploying, and discovering Web Services. This API enables use of a single abstraction API to access a variety of XML Registries. JAXR details bindings between JAXR information model and both UDDI V. 2.0 and ebXML registries.

The Application developer may not be directly using the JAXR API but would register Web Services transparently with the facilities provided by the application Server.

#### Java - XML translation

Java API for XML Binding (JAXB) makes XML easy to use by compiling an XML Schema into Java classes. The generated classes provide runtime support to the user by translating from Java to XML and vice versa. This makes communication with XML as simple as communicating using Java. This can also be used by applications for easier and direct XML handling.

Currently in its early stages, this specification is not a part of the J2EE 1.4 platform.

#### Enterprise Web Services Specification

This standard outlines the requirements and conceptual architecture for Web Services support in J2EE environments. This specification deals with the Web Services client programming model, Web Services server programming model, the packaging and deployment model, WSDL bindings and security support. In essence, this standard defines the Web Services programming model in J2EE, making use of the Java-XML specifications detailed above.

#### J2EE 1.4 Specification

This specification defines the functionality support from any certified platform provider, mandates well-defined support for Web Services. A single point standardization of all the services to be provided by any J2EE platform compliant vendor enables the J2EE application developers to port their applications across various vendors.

## Choosing a J2EE Vendor for Web Services Implementation

The standardization of J2EE platform has led to the emergence of variety of compliant environments. These environments can broadly be classified into

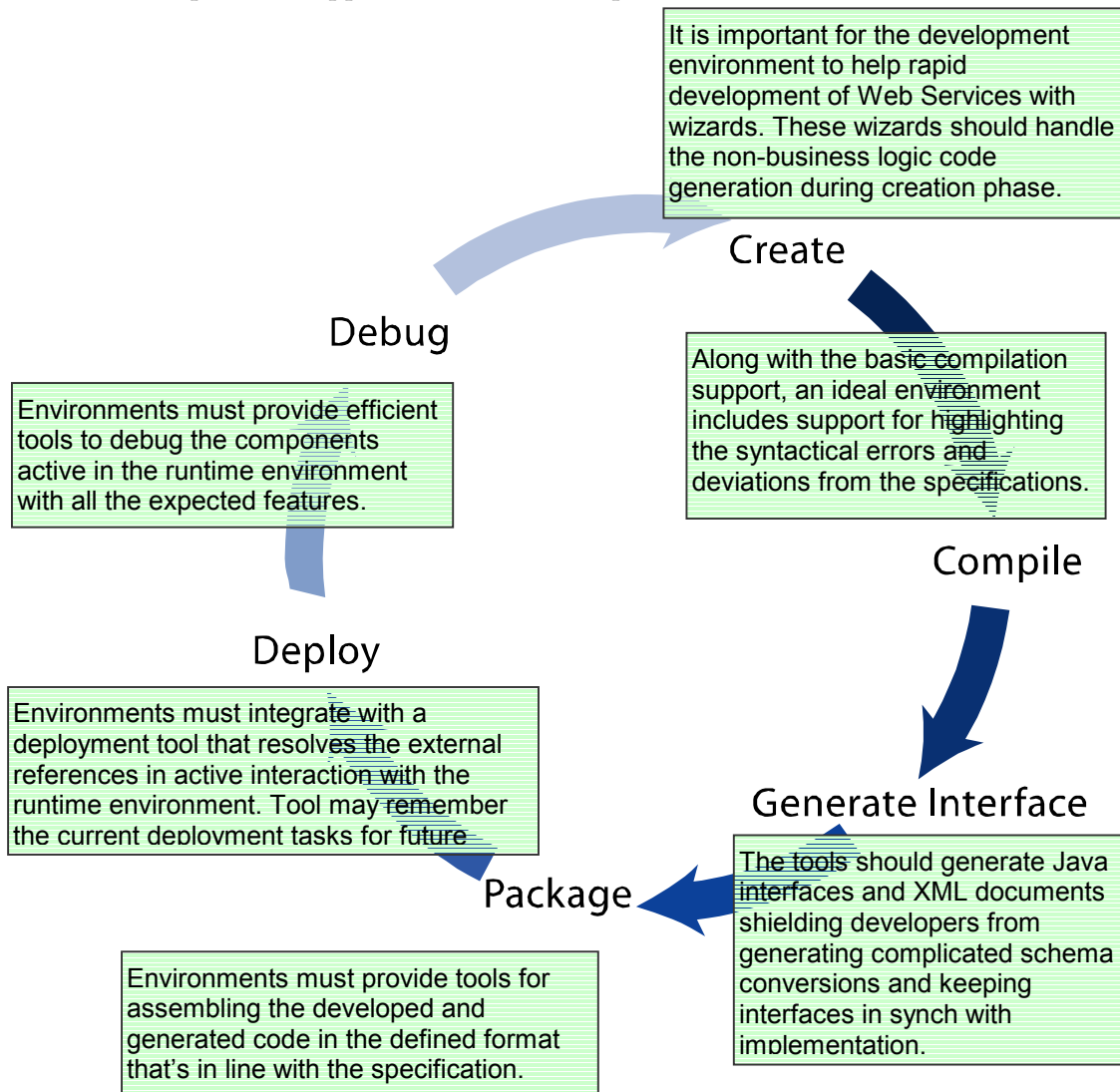
- Development and assembly environments
- Runtime environments

The following section elucidates the essential components and desirable features of the two environments. It also highlights the factors to consider while choosing the right technology environment for the enterprise.

#### Development Environments

The continuous evolution of the J2EE framework to support advanced features has also produced a corresponding rise in complexity. Effective Web Services development makes the case for tools that simplify and enhance productivity of the development teams by a transparent handling of the framework complexity.

An integrated development environment should handle all the tasks running through the development life cycle. The various processes involved as a part of the development life cycle are given below. It is essential that the development environment provides support for all the developer tasks listed below



Listed below are the additional features that may be supported by the Development Environment

#### Integration with multiple runtime environments

Integration with multiple runtime environments provides the enterprise with the option to choose the development environment independent of runtime environment. Some of the advanced development environments also provide for seamless porting of Web Services from one runtime environment to another.

#### Providing a complete environment

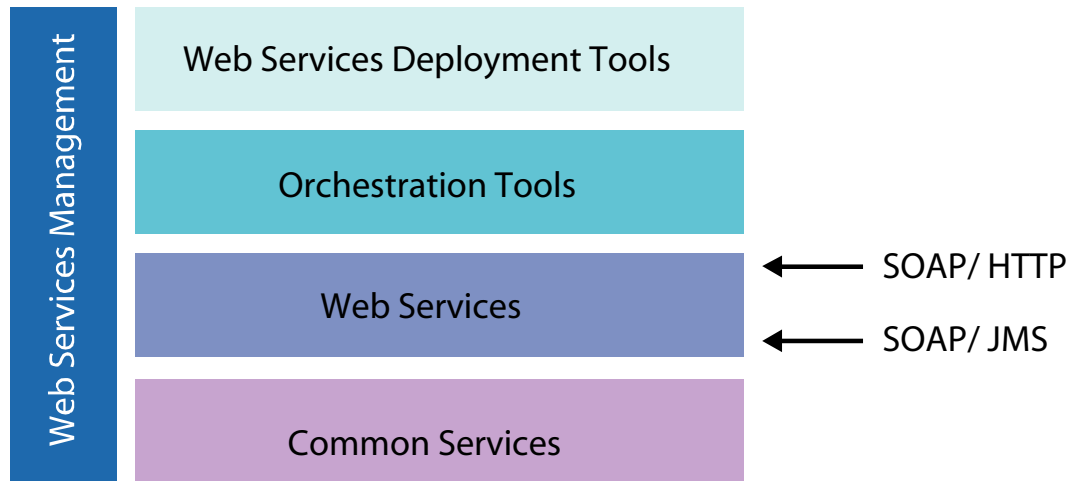
It is very useful for the development environment to provide a real life runtime environment that is normally not available for access to the developer. These include facilities like bundled Web Services registry and testing tools.



## Runtime Environments

The runtime environment for J2EE Web Services support should include the common service support mandated by the specification along with certain additional services that lessen the complexity of deploying and managing Web Services.

The figure below gives an insight the building blocks that provide an enterprise standard J2EE Runtime environment.



### Standard Web Services Support

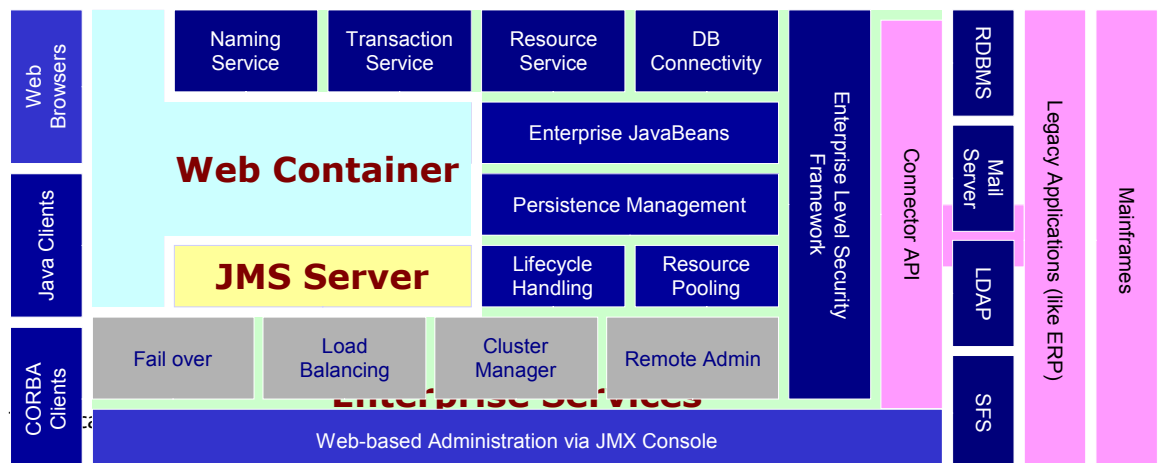
This is the essential service for hosting a web service. This service handles the responsibility of

- Translating the SOAP messages into Java calls
- Dynamic registration and enquiry of Web Services from the registry
- Interactions with the existing common services.

### Existing Common services

These services are a mandatory part of the current J2EE standards and have existed even prior to web service support. While some services (say load-balancing & failover support) shown below are not a part of the Standards, they are still viewed upon as the core services, which are essential for the enterprise adoption.

Along with the support for all the services these features like robustness, scalability and performance will be crucial in determining the total cost of managing Web

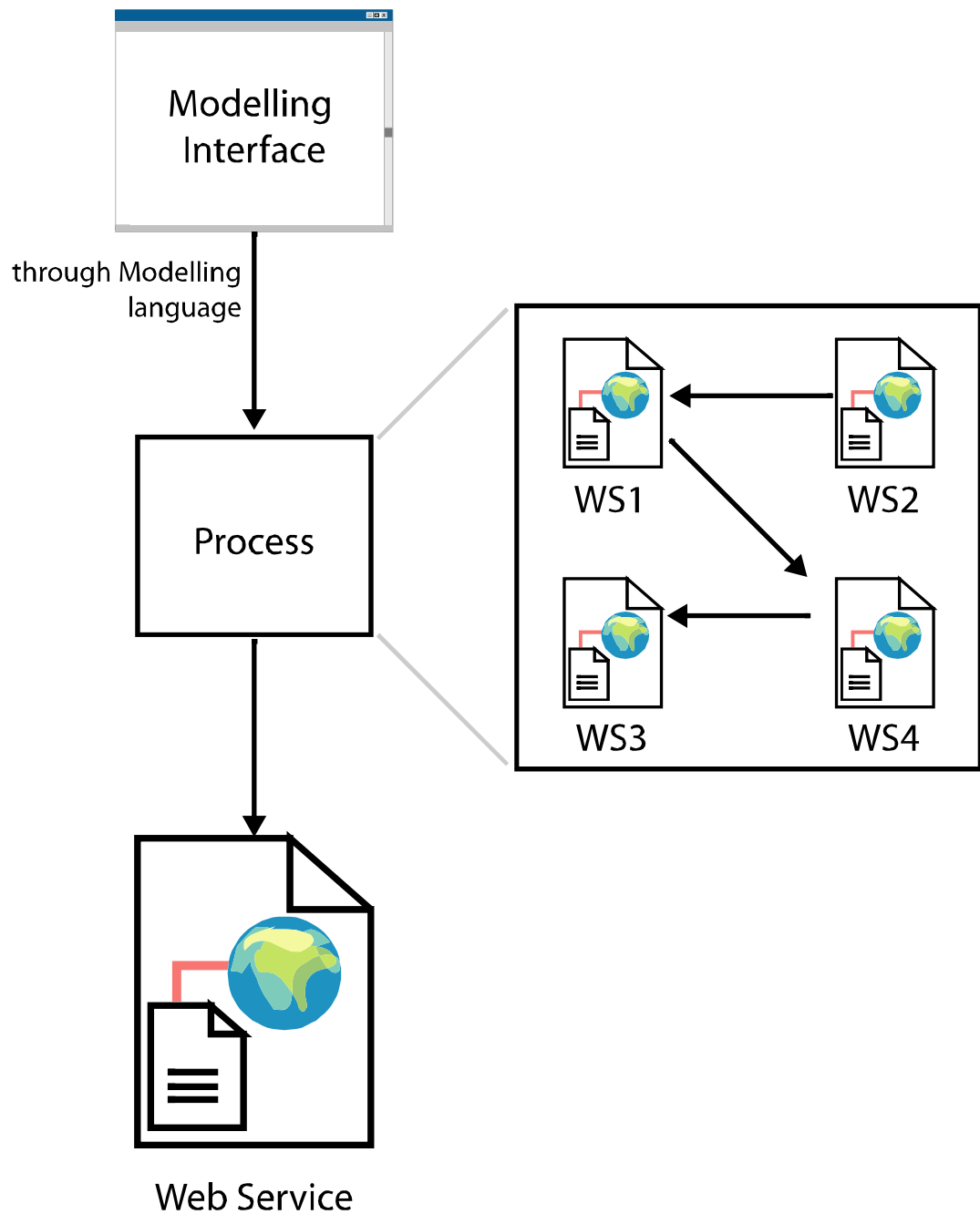


Services.

**Orchestration Service**

The basic support for Web Services opens the window for interaction with external world but still falls short of providing enterprises with flexibility of coping with constantly evolving and dynamically changing business processes. However, it is possible to provide a mechanism for modeling the business processes in a way that the business analysts are provided the leeway to define and modify the business processes, using Web Services. Though an additional feature, this would provide tremendous flexibility to enterprises in managing their IT infrastructure.

Orchestration service is an extension to the basic Web Services support and is to be provided on top of the essential technologies like SOAP, WSDL and UDDI. Having an orchestration services as a part of the runtime environment provides access to the complete power offered by Web Services.

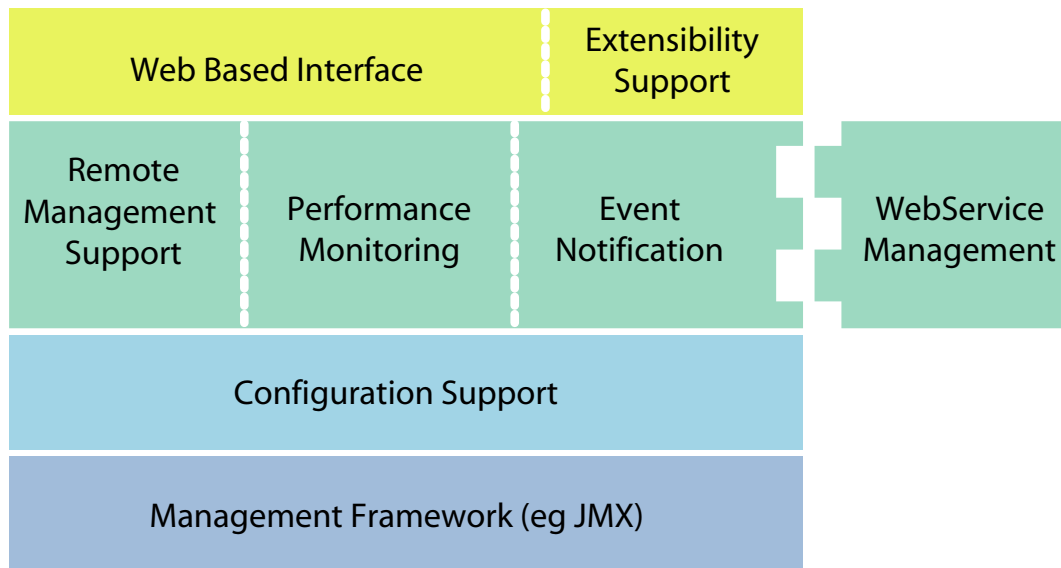


### Web Services Management

Management of the application servers has always been important, and with Web Services that provide for interaction with the external world, this aspect acquires further significance.

It would be desirable to choose an environment that provides a way of managing the application server as well as allows for integration of server management with the management of the Web Services in a generic fashion. It would be also be helpful for

the Web Services hosting organization to monitor the activity and performance of Web Services, along with access to an event notification facility.



## Conclusion

One of the essential reasons for the wide following for Web Services is because of its ability to communicate in an interoperable manner. This interoperability liberates the technical managers from inter-firm and intra-firm application integration thus opening a huge window of opportunity for streamlining existing IT infrastructure as well as for newer business models. Adoption of an evolving technology like Web Services demands and assumes technical maturity from enterprises. J2EE platform is an established and accepted platform that provides a robust, scalable and high-performing environment for enterprise applications. Hence J2EE is expected to be the platform of choice for providing Web Services.

