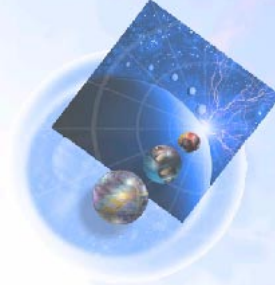


IBM WebSphere Application Server V4.0



Web Services

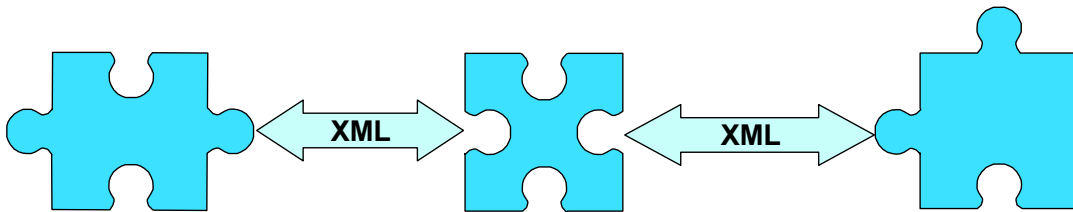
Next Generation of
e-business



What are Web Services?



- Self-contained, self-describing modular applications that can be published, located, and invoked across the Web
- Think of Web Services as Legos® with logic - interlocking blocks that create open distributed systems
- A service is basically a URL with XML-in and XML-out



- IBM®

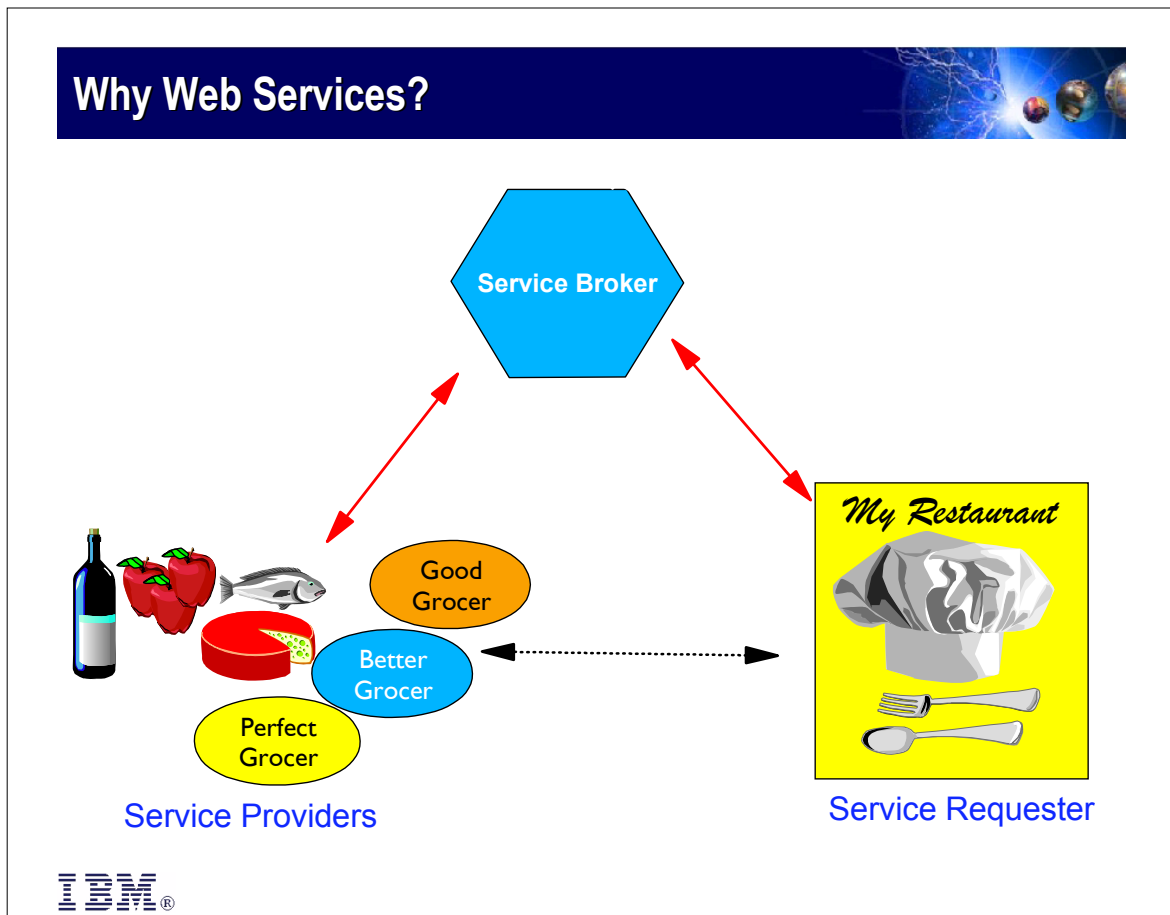
Topics



- Example - How Web Services can help
- Web Services
 - ▶ Concept
 - ▶ Architecture
 - ▶ Components
 - SOAP
 - WSDL
 - UDDI
- What's in WebSphere V4.0
- What's on AlphaWorks
 - ▶ <http://www.ibm.com/alphaworks>
- Positioning WebServices and J2EE



Why Web Services?



- ▶ The functional enhancements and the commitment to open standards that characterize the WebSphere V4.0 release go well beyond mere support of J2EE. In fact, IBM believes that the market of information technologies is mature enough to move on to the next step in terms of distributed, Internet-enabled applications. That's where Web Services play a key role.
- ▶ Let me take you through a simple scenario that highlights the technological advantages and the business opportunities that sit behind the concept of Web Services.
- ▶ Your favorite restaurant needs to periodically purchase groceries to replenish its supplies in the kitchen. For several years, the restaurant has been using a certain order management software package that has proven to be satisfactory. However, the process of selecting the best provider for the groceries, creating a purchase order, and transferring this order to the provider (into their sales order management system, which happens to be implemented by a different software package) still requires intense human intervention.
- ▶ Ideally, the restaurant would like to have access to an infrastructure that allows it to select among a number of suppliers, possibly in a programmatic way, and then to connect the purchase management application with the supplier's sales order management system with no manual effort.
- ▶ In a nutshell, this is what Web Services is all about. Rather than requiring the restaurant to interact directly with a specific provider, a process that requires manual intervention or heavy duty software customization, a broker could make available a directory of services that are accessible with standard protocols. The restaurant would be able to discover the best grocer in the directory, using certain standard APIs, then exchange information through standard interfaces with the selected provider. The providers would be interested in publishing their services with the broker, so that the requesters can discover them. The way the services are published and discovered, and the way the provider and the requester communicate is regulated by simple standards; no need of creating ad-hoc layers of software to make all this happen.
- ▶ The broker represents a new, potentially very lucrative business model.

Web Services: Benefits



- **Business value**
 - ▶ Business processes can dynamically and flexibly interact across enterprises
 - ▶ New, large business opportunities (brokers)
- **Technical value**
 - ▶ Applications can dynamically navigate, discover, and interact over the Internet
- **Developer's value**
 - ▶ Modular and agile application services that can be integrated with no regard to implementation details



- ▶ There are many aspects of the business value of Web Services. Web Services enable dynamic, flexible, potentially real-time selection of any kind of business service across the Internet. Business processes running in different enterprises will be able to bind together and cooperate as if they belonged to a seamlessly designed system - although they might be implemented by different software vendors or using different software technologies.
- ▶ The Web Services broker represents a completely new model for business opportunities, created by the Web Services technology.
- ▶ Technically, Web Services reduces to zero the effort of system integration. WebSphere provides tools to turn existing applications into Web Services--even legacy applications may become Web Services. The key element in terms of technical advantage is the loosely coupled nature of the integration. Service Providers and service requesters do not need to directly interact to tie their applications together.
- ▶ From a developer's standpoint, Web Services provide a standard way to create new modular applications or to re-engineer existing applications so that they can be published and discovered on the Internet, dramatically expanding their potential.

Web Services: What Does It Take?



- A "Lingua Franca" for exchanging information
 - ▶ XML
- Network-neutral service access protocol
 - ▶ SOAP - Simple Object Access Protocol
- A way to catalog and describe services
 - ▶ WSDL - Web Services Description Language
- A way to find out about available services
 - ▶ UDDI - Universal Discovery Description and Integration



- ▶ There are four essential elements that make Web Services possible.
- ▶ Web Services need a neutral and structured way to represent business information. XML provides an ideal framework for describing what a service can do and how to invoke it, and even for actually invoking a service, passing parameters, and getting responses back.
- ▶ Service providers and service requesters interact by exchanging XML messages. A neutral and universal mechanism for exchanging XML messages is therefore needed for Web Services to become a reality. SOAP offers a transport-neutral mechanism to carry XML messages around.
- ▶ Service providers need to be able to describe what kind of services they offer. Service requesters need to be able to understand what services are available and how to invoke them. WSDL is an XML standard to describe Web Services.
- ▶ Service providers need to advertise their services and requesters need to be able to inquire the available services. The last requirement is therefore a structured directory; a registry that providers can use to publish their services and requester can use for querying. The UDDI standard (based on a series of SOAP messages) provides this services oriented registry.

Web Services: Key Functions

Service Broker

- A searchable repository of service descriptions
- Service Providers publish their services
- Service requesters find services



Service Provider

- Provide applications as Webservice
- Publish their services

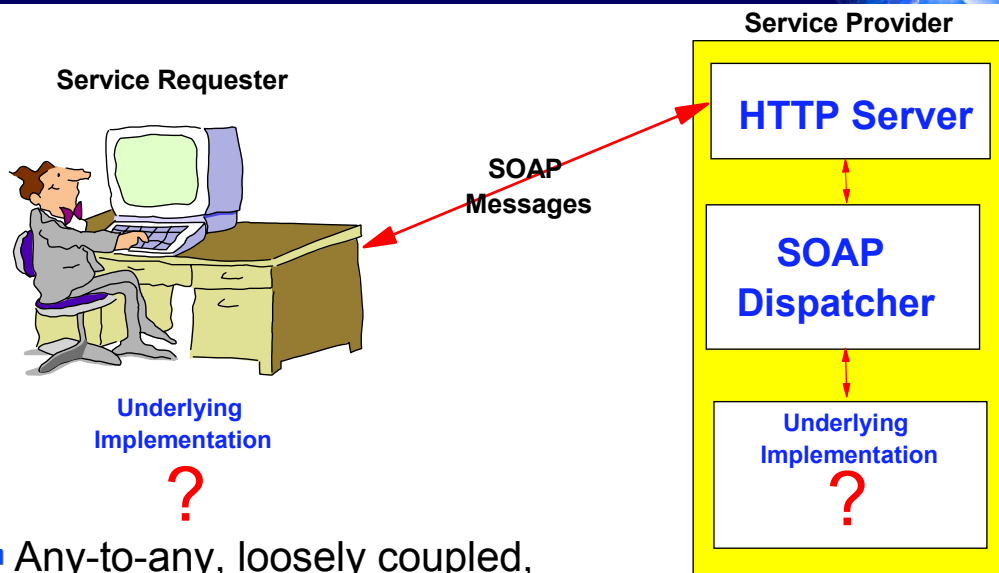
Service Requester

- A client that needs a service
- Publish their needs



- ▶ The Service Provider owns a certain group of services. Therefore, the service will ultimately "run" on the providers' platform when it is invoked. A service is invoked by a requester through an XML message. XML messages can be carried across the Internet through a network-neutral standard protocol called Simple Access Object Protocol, or SOAP. The provider has to advertise the service. This can be done by describing the service itself with a standard encoding, defined in the Web Services Description Language standard, or WSDL.
- ▶ The Service Requester is the business that requires a certain function to be fulfilled. From an architectural perspective, this is the application that is looking for and invoking a service. The requester has to find the service before invoking it - this process of discovery involves accessing a directory where the services are published. The access occurs through a set of standard APIs defined in the UDDI standard. UDDI stands for Universal Description, Discovery and Integration specification.
- ▶ The Service Broker owns the directory of all the available services. The obvious business model for the broker is to charge the providers and the requesters for the use of the directory. When providers need to advertise their services, they will do so by creating an entry in the broker's directory. When a requester needs to access a service, the broker will hand out the binding information to the requester. This information tells the requester how to invoke the service and what to expect in return. From that point onwards, the broker will not be involved any longer in the transactions that occur between the provider and the requester.

SOAP: More on the Technical Value



- Any-to-any, loosely coupled, implementation neutral application integration
- Simplifies all aspects of application integration
 - ▶ Including testing, debugging



- ▶ SOAP effectively hides the details of the implementation of the end points. The Web Service implementation could be a legacy COBOL application, a new EJB system, the interface to a commercial product written in C++, or anything else that can interpret a request message and return an appropriate response. Similarly, the service provider knows nothing about the implementation of the requester.
- ▶ The point is that neither the service requester and the service provider knows or cares about anything besides the format and content of request and response messages (loosely coupled application integration)
- ▶ Testing is also simplified. It's easy to write a dummy requester or provider to generate or handle SOAP messages in a local test environment before going live. A Service Requester application can be developed and tested before the Web Service is available. The clear line between Service Requester and Service Provider isolates the two parties, greatly simplifying debugging.
- ▶ SOAP, used as a remote procedure call where the message contains a method name and arguments, extends the power of object-oriented programming to web-based remote objects with real-world capabilities, like causing a book to arrive at your door. SOAP can also be used for exchanging documents containing any kind of XML data. It enables complete reuse of code, from systems of any type, both inside your company and among business partners. It fosters transparent integration to systems of any type.

SOAP - Overview of the Technology



- XML-based protocol supports the exchange of information in a decentralized, distributed environment
 - ▶ Transport independent: HTTP(S) will be most common
 - ▶ Information in SOAP XML
 - An envelope that describes the message and how to process it
 - Encoding rules for application-defined data types
 - Convention for representing calls and responses
 - ▶ Supports 2 types of exchanges:
 - Remote procedure calls (SOAP-RPC)
 - ◆ Marshalling/unmarshalling datatypes to/from XML
 - ◆ Packaging them up into a SOAP envelope
 - ◆ Send SOAP envelope across HTTP as a POST to the Application Server
 - Document-oriented messaging
 - ◆ Allows for the transmission of an arbitrary XML document within the body of the SOAP envelope.



- ▶ Historically, SOAP was meant to be a network- and transport-neutral protocol to carry XML messages around. A SOAP message is made of two parts, the envelope and the body, where the envelope contains information such as the recipient and the originator and the body contains the actual XML message.
- ▶ SOAP over HTTP became the premier way of implementing this protocol to the point that the latest SOAP specs mandate HTTP support. But conceptually there is no limitation as to the network protocol that could be used. For example, since HTTP is a transport that doesn't guarantee delivery and is non-transactional, one could envision that SOAP messages could be carried around by a messaging software such as MQ-Series.
- ▶ The third stage of the evolution of SOAP is represented by SOAP-RPC - the body of the message contains a call to a remote procedure (expressed in XML) and the parameters to pass in (again, expressed in XML).
- ▶ SOAP is therefore the network- and transport-neutral protocol that allows a client (no matter what technology is used to implement it, as long as it can issue an XML message) to call a remote service no matter how it has been implemented, as long as it can process an XML message.

WebSphere V4.0: SOAP support

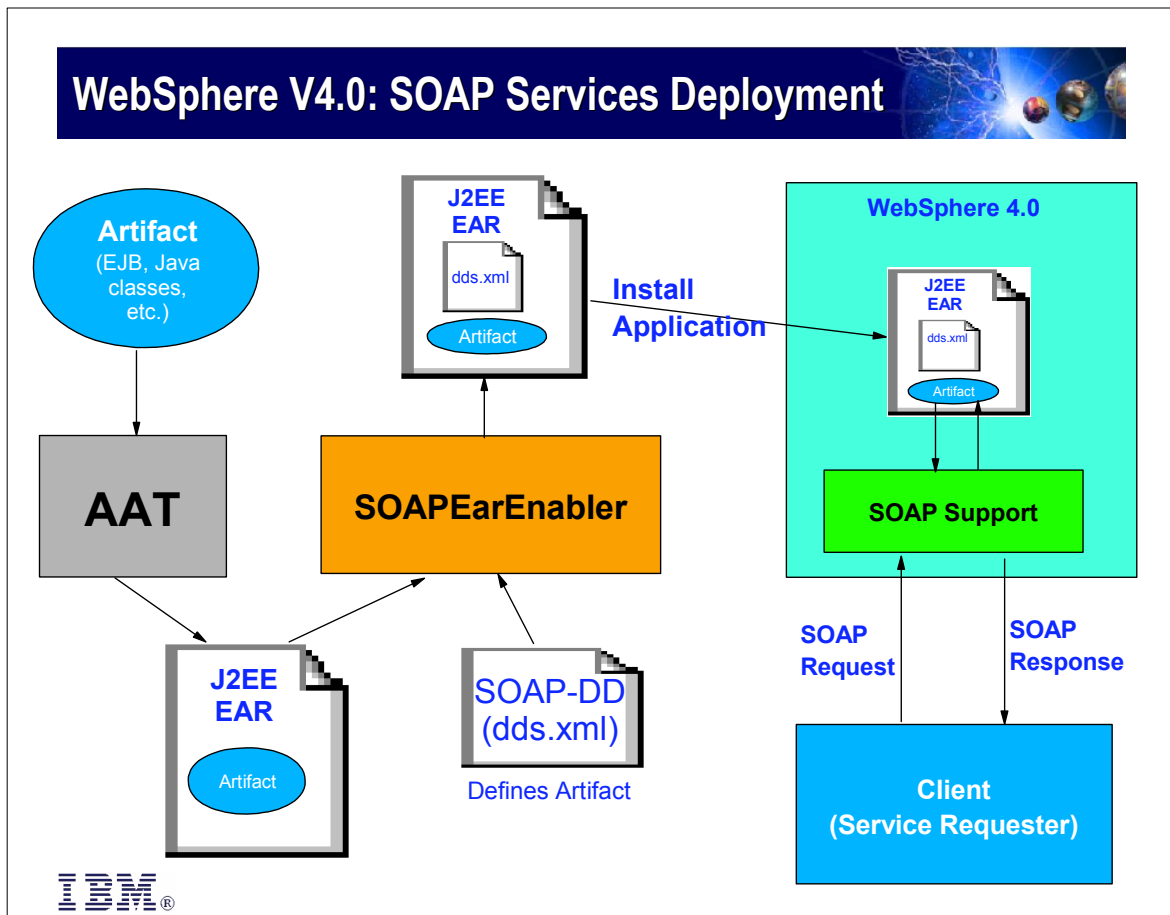


- Integrated Apache SOAP V2.1 (Java implementation of SOAP 1.1 spec)
 - ▶ soap.jar
 - ▶ soapsec.jar (Security extensions for SOAP to allow secure connections)
- Exposes following as SOAP services (artifact)
 - ▶ Standard Java classes
 - ▶ EJBs
 - ▶ Bean Scripting Framework (BSF) supported scripts
 - ▶ DB2 stored procedures
- Tools to support deployment of above SOAP services



- ▶ In WebSphere V4.0, we provide support for SOAP and secure SOAP.
- ▶ Although conceptually any implementation of a service could be turned into a SOAP service, WebSphere V4.0 supports the following technologies:
 - ▶ Java classes (any Java class and its methods can be configured in WebSphere as a SOAP service)
 - ▶ EJBs
 - ▶ BSF (procedures written in scripting languages as Microsoft JScript for instance)
 - ▶ DB2 stored procedures
- ▶ The list is likely to grow significantly in the future.

WebSphere V4.0: SOAP Services Deployment



- ▶ You can configure an artifact (Java class, EJB, BSF script, or DB2 stored procedure) as a SOAP service in WebSphere.
- ▶ You first need to create a SOAP deployment descriptor. You can use the WebServices Studio Application Developer to generate one for your Java classes, EJB, and BSF scripts - or you can create it manually (few lines of XML)
- ▶ You then need to package your artifact in a J2EE EAR file, using AAT.
- ▶ You are now ready to combine the artifact, the deployment descriptor, and the WebSphere SOAP engine in a single EAR file using the SOAPEarEnabler.
- ▶ You'll obtain an augmented EAR file that is ready to be installed in WebSphere.

Web Services Description Language (WSDL)



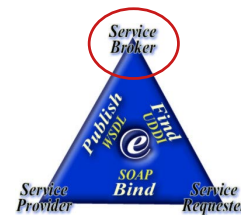
- WSDL 1.0 submitted to W3C by IBM, Microsoft, and others
- WSDL is the XML equivalent of a résumé
 - ▶ Describes what a Web Service can do, where it resides, and how to invoke it
 - ▶ Describes network services as collections of communication endpoints capable of exchanging messages
 - ▶ Serves as a recipe for automating the details involved in applications communication



Locating Web Services



- Application designers need to find service providers
- Tools need to get descriptions for services to generate client proxies/stubs
- Applications may need to find services at runtime
- UDDI: Universal Description, Discovery, and Integration Initiative defines service broker architecture architecture
 - ▶ <http://www.uddi.org/>
- IBM UDDI Test Area
 - ▶ <http://www.ibm.com/services/uddi/testregistry/>
- IBM UDDI Registry
 - ▶ <http://www.ibm.com/services/uddi>



- ▶ This chart summarizes about the roles of the service requester and of the service broker in the Web Services arena.
- ▶ The requester needs to retrieve a service definition (discovery) and to create a client for the service (binding).
- ▶ This can be greatly automated through tools like WebSphere Studio Application Developer.
- ▶ Part of the UDDI architecture is also the ability to dynamically (programmatically) finding and binding services. The same application may need to be bound to different service providers depending, for instance, on the geography where it is going to run. This dynamic Find/Bind process allows an application to be deployed internationally and to use an optimal provider without having to write specific code.

UDDI Registry Information



- Combination of White, Yellow, and Green Pages

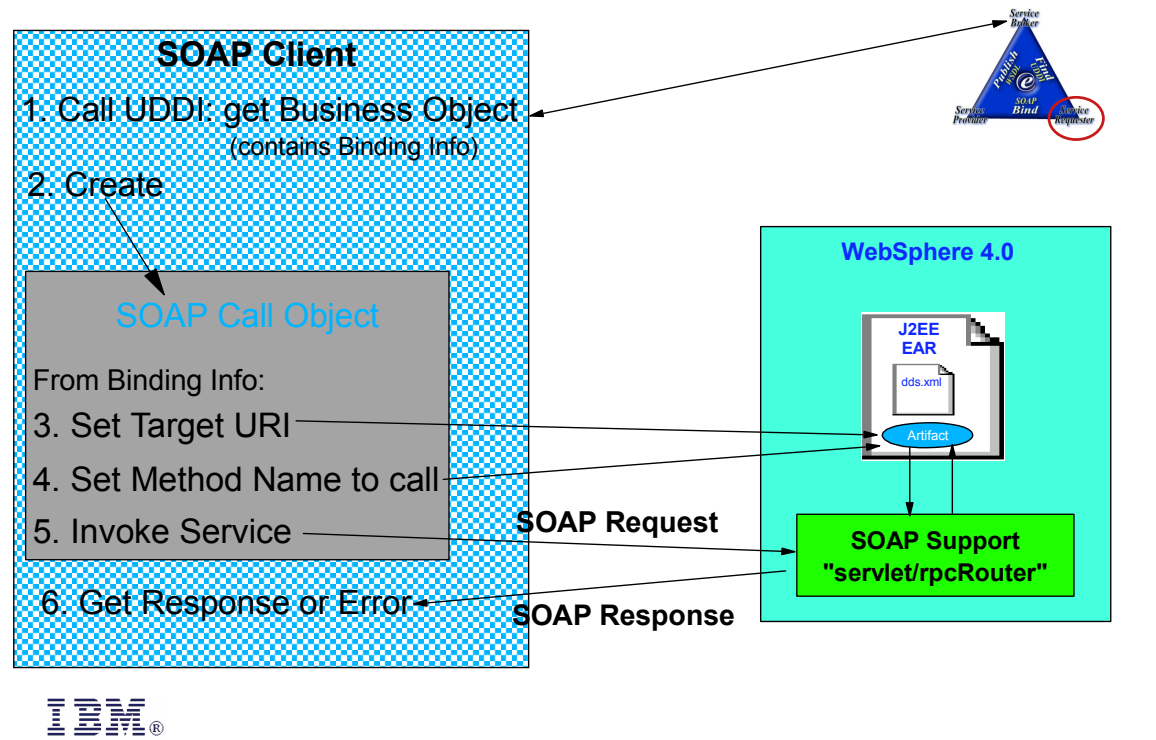
- White Pages
 - ▶ Business Name, Description, and Contact Information

- Yellow Pages
 - ▶ Organized in business categories and taxonomies

- Green Pages
 - ▶ Describe how businesses can communicate electronically
 - Multilingual process descriptions
 - Service Interfaces
 - Pointers to well defined specifications



SOAP Client



- ▶ In WebSphere V4.0, creating a Java client for a SOAP service is extremely simple. All you need to do is to generate the "client proxy". The proxy contains code that instantiate a "Call" object. The Call object takes care of creating the "right" XML message, based on the service to be called and on the parameters it takes.
- ▶ The Call object also holds the location (urn) of the service. All your code needs to do is to create an instance of the proxy and call the method that corresponds to the service to invoke. The Call object will return a SOAP response that you can then analyze (for instance through an XML parser like Xerces).

WebSphere V4.0 Support for Web Services

- Integration of SOAP Support

- ▶ SOAP server and client environments
- ▶ Enables WebSphere applications to send and receive SOAP messages, and leverage WSDL

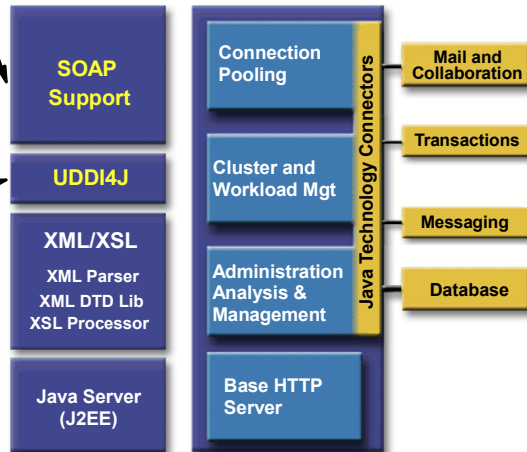
- Integration of UDDI4J

- ▶ Java interface to UDDI registries
- ▶ Enables WebSphere applications to communicate with UDDI-compliant registries to publish and find Web services

- Tooling

- ▶ WebSphere Studio Application Developer

WebSphere Application Server V4.0



- ▶ WebSphere provides support for both SOAP server and client environments.
 - ▶ This enables WebSphere applications to send and receive SOAP messages.
- ▶ WebSphere's integration of a Java interface to UDDI enables WebSphere applications to publish and find Web services.
- ▶ Integration of SOAP enables Web services deployed on the WebSphere platform to utilize platform strengths such as Security, Transaction monitoring, and Trace/Debug functions.

SOAP Security over HTTP



- HTTP basic authentication: userid/password
 - ▶ Apache SOAP has API to set the userid/password for the HTTP basic authentication

- SSL access (configured in Web Server)

- SOAP Signature
 - Applications may need non-repudiable proof of exchanged messages
 - Business partners establish some form of trust relationship based on public keys.
 - Can be done using PKI through a third party CA or through exchanging public keys via a secure channel (SSL)
 - ◆ Service deployed with a signature verification function
 - ◆ Client application sends a request that is signed and transmitted to the server.
 - ◆ At the server side, it is verified and delivered to a server application (in case of RPC, a Java object).
 - Response is processed in the same manner.



- ▶ SOAP security extension
 - ▶ Still work in progress
 - ▶ Included in WAS V4.0, based on the SOAP Security specification, and widely-accepted security technologies such as SSL.

Summary - Web Services vs. J2EE



- J2EE is about **implementation**
 - ▶ Standard for implementing business processes through robust, scalable, portable applications

- WebServices are about **interfaces**
 - ▶ Enable loosely-coupled integration of business processes
 - Irrespectively of the technology used for implementation
 - ▶ Maximize the exposure of business processes to the potential of the Internet



- ▶ Summarizing - how do Web Services fit with the J2EE standards?
- ▶ J2EE is about implementation. IBM wants to encourage customers to build their services by taking advantage of the scalability, robustness, and portability offered by IBM's J2EE platform - WebSphere V4.0
- ▶ Web Services are about interfaces. Web Services allow applications that run on different platforms, and developed with different technologies to easily interact with each other in a loosely coupled integration that doesn't require a specific, ad-hoc integration effort.