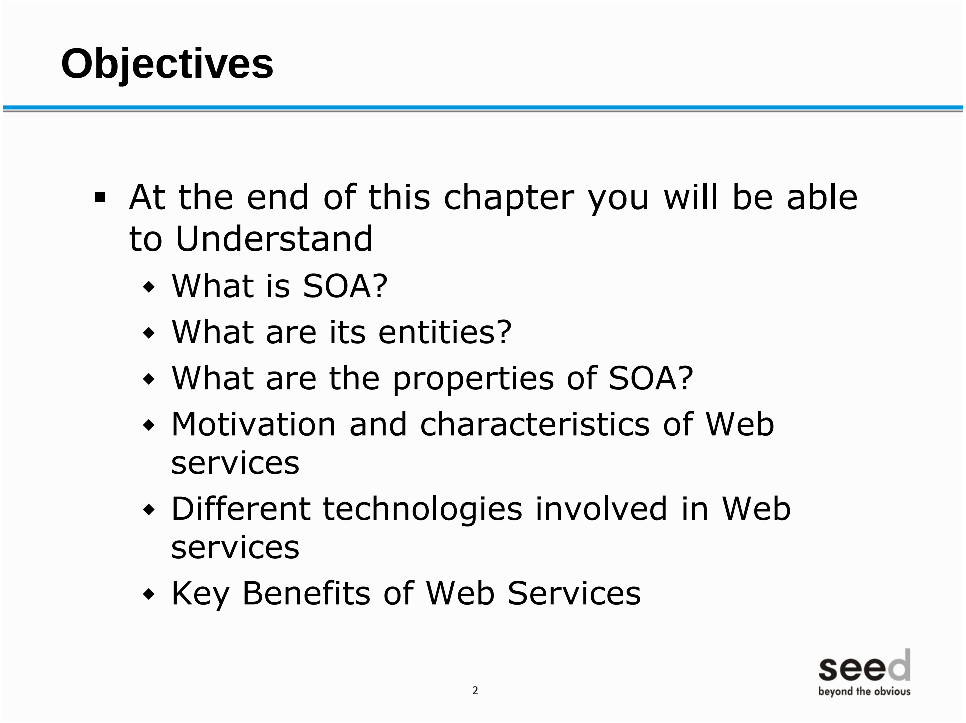
SOA And Web Services Introduction



Objectives

- At the end of this chapter you will be able to Understand
  - What is SOA?
  - What are its entities?
  - What are the properties of SOA?
  - Motivation and characteristics of Web services
  - Different technologies involved in Web services
  - Key Benefits of Web Services

Service Oriented Architecture (SOA)

- Web services promote an environment for systems that is loosely coupled and interoperable. Many of the concepts for Web services come from a conceptual architecture called service-oriented architecture (SOA).

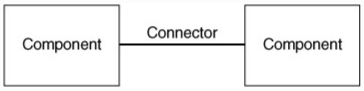
- SOA configures entities (services, registries, contracts, and proxies) to maximize loose coupling and reuse.

- ***Software architecture*** describes the system's ***components*** and the way they ***interact*** at a high level.

Service Oriented Architecture (SOA)

- The interactions between components are called ***connectors****.* The configuration of components and connectors describes the way a system is structured and behaves

- ***Software architecture*** describes a system's components and connectors.



Component

Connector

Component

Service Oriented Architecture (SOA)

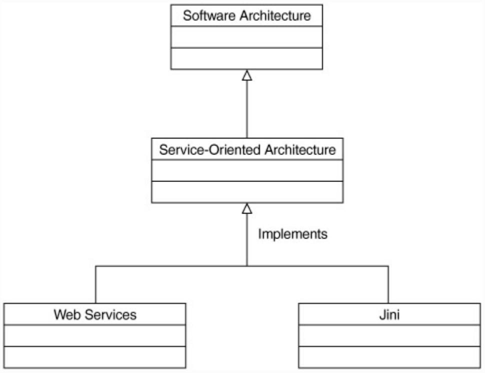
- The ***software architecture*** of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them
- ***Service-oriented architecture*** is a special kind of software architecture that has several unique characteristics
- ***Sun*** defined SOA more rigorously in the late 1990s to describe ***Jini***, a lightweight environment for dynamically discovering and using services on a network
- The ***goal*** in developing ***Jini*** was to create a dynamically networked environment for devices, services, and applications. In this environment, services and devices could be added to and removed from the network dynamically*

- * Ref, Sun Microsystems, *Jini Network Technology*, www.sun.com/jini.

Service Oriented Architecture (SOA)

- ***Web services*** are simply one set of technologies that can be used to implement SOA successfully.

- The most important aspect of service-oriented architecture is that it separates the service's implementation from its interface.

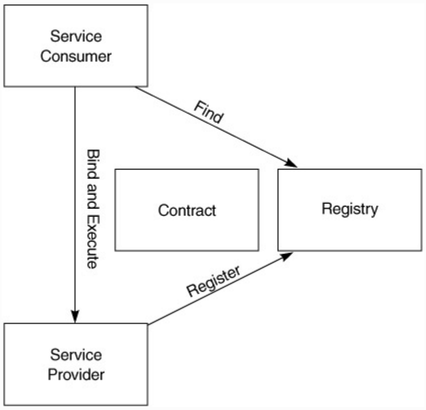


Service Oriented Architecture (SOA)

- Consumers expect that their interaction with the service will follow a contract, an agreed-upon interaction between two parties.
- The way the service executes tasks given to it by service consumers is irrelevant. The service might fulfill the request by executing a ***servlet***, a ***mainframe application***, a ***Visual Basic application***, or an ***EJB application***. The only requirement is that the service send the response back to the consumer in the agreed-upon format.

SOA Entities

- The "***find, bind, and execute***" ***paradigm*** as shown allows the ***consumer*** of a ***service*** to ask a third-party ***registry*** for the service that matches its criteria. If the registry has such a service, it gives the consumer a ***contract*** and an ***endpoint address*** for the service.
- SOA consists of the following six entities configured together to support the find, bind, and execute paradigm.

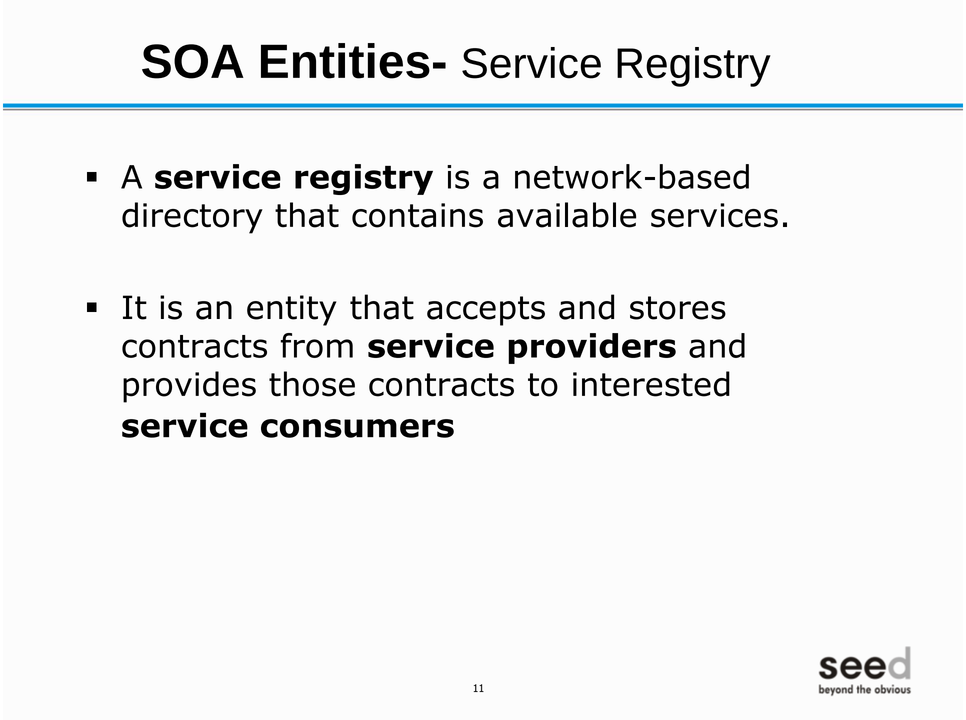


SOA Entities- Service Consumer

- **Service Consumer** is an application, service, or some other type of software module that requires a service.
- **Service Consumer** is the entity that initiates the locating of the service in the registry, binding to the service over a transport, and executing the service function.
- The **service consumer** executes the service by sending it a request formatted according to the contract.

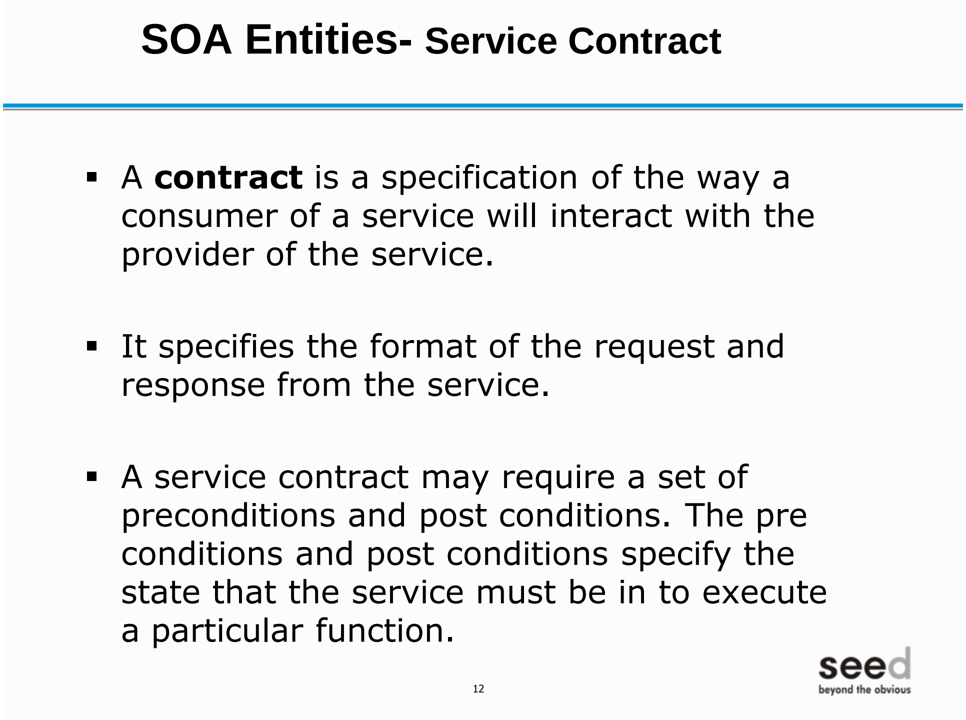
SOA Entities- Service Provider

- The **service provider** is the service, the network-addressable entity that accepts and executes requests from consumers.
- **Service provider** can be a mainframe system, a component, or some other type of software system that executes the service request.
- The **service provider** publishes its contract in the registry for access by service consumers.



SOA Entities- Service Registry

- A **service registry** is a network-based directory that contains available services.
- It is an entity that accepts and stores contracts from **service providers** and provides those contracts to interested **service consumers**

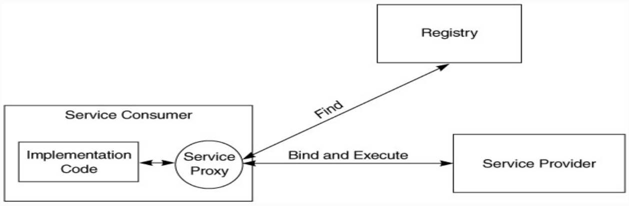


SOA Entities- Service Contract

- A **contract** is a specification of the way a consumer of a service will interact with the provider of the service.
- It specifies the format of the request and response from the service.
- A service contract may require a set of preconditions and post conditions. The pre conditions and post conditions specify the state that the service must be in to execute a particular function.

SOA Entities- Service Proxy

- The service provider supplies a ***service proxy*** to the service consumer.
  - The service consumer executes the request by calling an API function on the proxy.
  - The service proxy finds a contract and a reference to the service provider in the registry.
  - It then formats the request message and executes the request on behalf of the consumer.
- The service proxy is a convenience entity for the service consumer.



SOA Characteristics

- Service-oriented software architecture has these characteristics*
  - Services are discoverable and dynamically bound.
  - Services are self-contained and modular.
  - Services stress interoperability.
  - Services are loosely coupled.
  - Services have a network-addressable interface.
  - Services have coarse-grained interfaces.
  - Services are location-transparent.
  - Services are composable.
  - Service-oriented architecture supports self-healing
  - * Stevens, *Service-Oriented*, 2002, Sun Microsystems, *Jini Technology Architectural Overview* 2001

What Are Web Services?

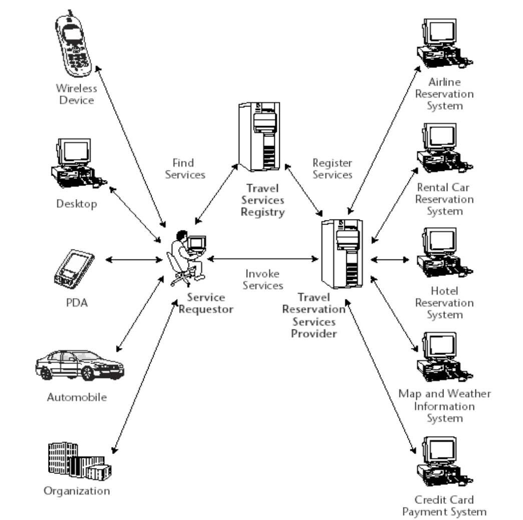
- Web services are based on the concept of service-oriented architecture (SOA).
- SOA is the latest evolution of distributed computing, which enables software components, including application functions, objects, and processes from different systems, to be exposed as services.
- According to ***Gartner research (June 15, 2001),*** *"Web services are loosely coupled software components delivered over Internet standard technologies."*
- Web services are ***self-describing*** and ***modular business applications*** that ***expose*** the ***business logic as services*** over the Internet through programmable interfaces and using Internet protocols for the purpose of providing ways to find, subscribe, and invoke those services.

What Are Web Services?

- Based on XML standards, Web services can be developed as ***loosely coupled*** **application components** using any programming language, any protocol, or any platform.
- This facilitates delivering business applications as a service accessible to anyone, anytime, at any location, and using any platform.

What Are Web Services?

- A simple scenario of how an organization’s business functionalities can be exposed as Web services and invoked by its customers using a wide range of application clients.



What Are Web Services?

- Web services are typically implemented based on open standards and technologies specifically leveraging XML.

- The XML-based standards and technologies, such as ***Simple Object Access Protocol*** (SOAP); ***Universal Description, Discovery, and Integration*** (UDDI); ***Web Services Definition Language*** (WSDL) are commonly used as building blocks for Web services.

Web Services Motivation

- Web-based B2B solutions are usually based on custom and proprietary technologies and are meant for exchanging data and doing transactions over the Web.

- B2B has its own ***challenges***. For example, in B2B communication, connecting new or existing applications and adding new business partners have always been a challenge. Due to this fact, in some cases the scalability of the underlying business applications is affected.

- **Ideally**, the business applications and information from a partner organization should be able to interact with the application of the potential partners seamlessly without redefining the system or its resources.

Web Services Motivation

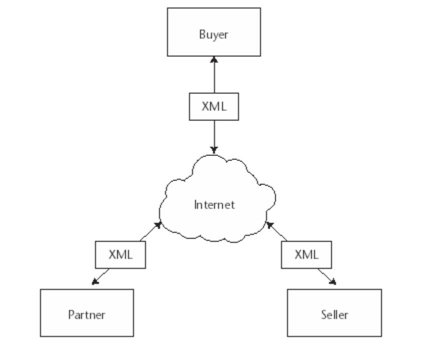
- To meet these challenges, it is clearly evident that there is a need for standard protocols and data formatting for enabling seamless and scalable B2B applications and services.

- Web services provide the solution to resolve these issues by adopting open standards.

Web Services Motivation

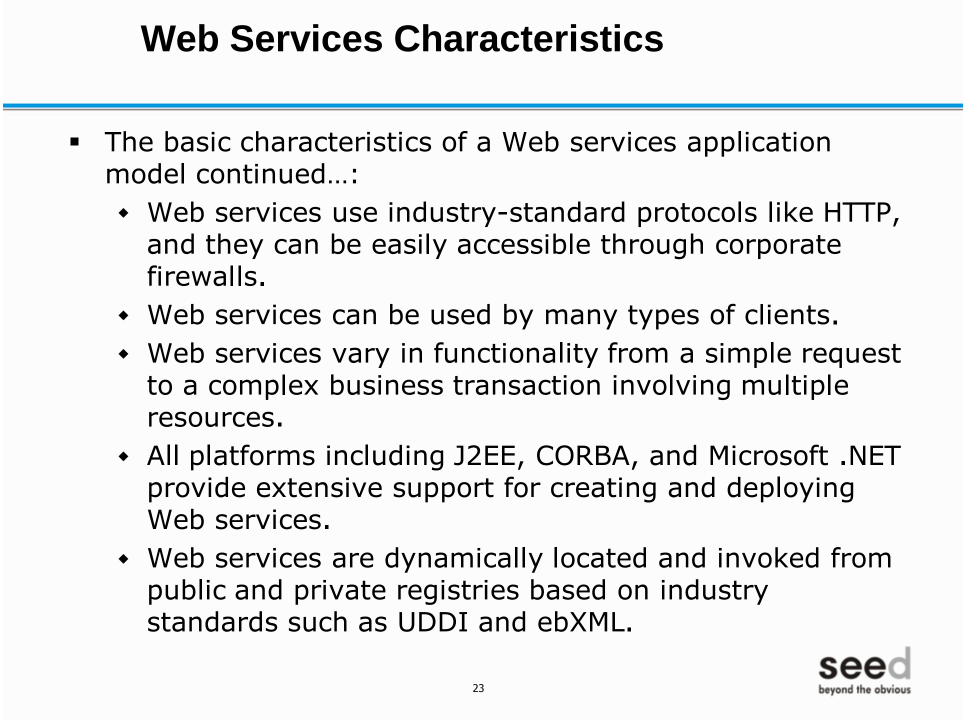
- Figure shows a typical B2B infrastructure using XML for encoding data between applications across the Internet.

- Web services enable businesses to communicate, collaborate, and conduct business transactions using a lightweight infrastructure by adopting an XML-based data exchange format and industry standard delivery protocols.



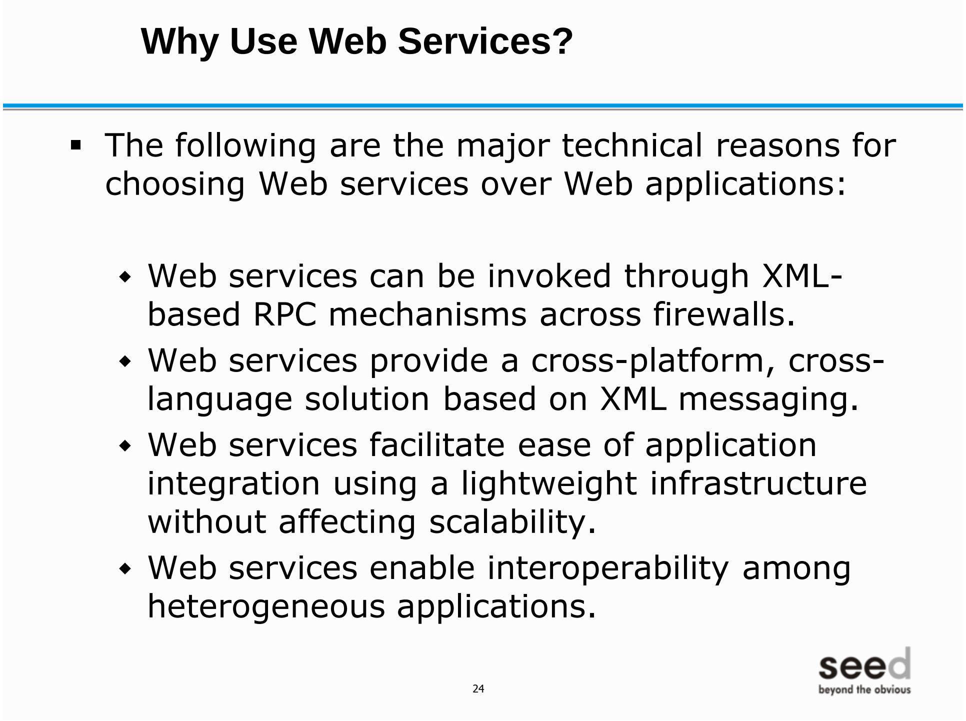
Web Services Characteristics

- The basic characteristics of a Web services application model are as follows:
  - Web services are based on XML messaging
  - Web services provide a cross-platform integration of business applications over the Internet.
  - To build Web services, developers can use any common programming language, such as Java, C, C++, Perl, Python, C#, and/or Visual Basic
  - Web services are not meant for handling presentations like HTML
  - Because Web services are based on loosely coupled application components, each component is exposed as a service with its unique functionality.



Web Services Characteristics

- The basic characteristics of a Web services application model continued…:
  - Web services use industry-standard protocols like HTTP, and they can be easily accessible through corporate firewalls.
  - Web services can be used by many types of clients.
  - Web services vary in functionality from a simple request to a complex business transaction involving multiple resources.
  - All platforms including J2EE, CORBA, and Microsoft .NET provide extensive support for creating and deploying Web services.
  - Web services are dynamically located and invoked from public and private registries based on industry standards such as UDDI and ebXML.

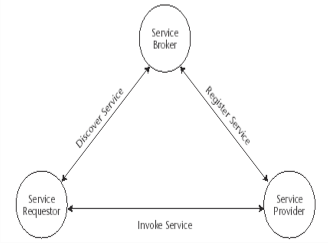


Why Use Web Services?

- The following are the major technical reasons for choosing Web services over Web applications:
  - Web services can be invoked through XML-based RPC mechanisms across firewalls.
  - Web services provide a cross-platform, cross-language solution based on XML messaging.
  - Web services facilitate ease of application integration using a lightweight infrastructure without affecting scalability.
  - Web services enable interoperability among heterogeneous applications.

Basic Operational Model of Web Services

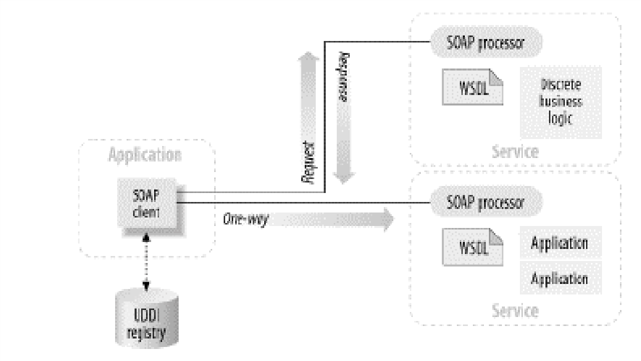
- **Service provider.** The service provider is responsible for developing and deploying the Web services. The provider also defines the services and publishes them with the service broker.
- **Service broker.** The service broker (also commonly referred to as a service registry) is responsible for service registration and discovery of the Web services.
- **Service requestor.** The service requestor is responsible for the service invocation. The requestor locates the Web service using the service broker, invokes the required services, and executes it from the service provider.



Core Web Services Standards

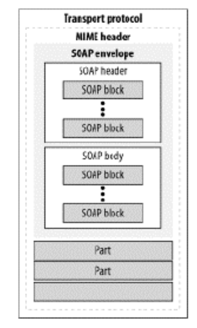
- **Extensible Markup Language (XML)** plays a vital role as the common wire format in all forms of communication.
- **Simple Object Access Protocol (SOAP)** is a standard for a lightweight XML-based messaging protocol. It enables an exchange of information between two or more peers and enables them to communicate with each other in a decentralized, distributed application environment.
- **Web Services Definition Language (WSDL)** standard is an XML format for describing the network services and its access information.
- **Universal Description, Discovery, and Integration (UDDI)** defines the standard interfaces and mechanisms for registries intended for publishing and storing descriptions of network services in terms of XML messages.

Discrete components in a web services architecture



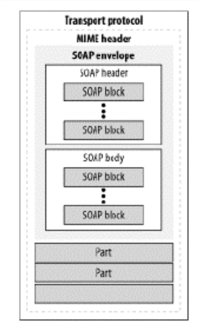
SOAP

- SOAP is an XML based protocol used to exchange information throughout a distributed environment.
- The ***SOAP envelope*** declaration is simply the outermost XML tag that delineates the boundaries of the SOAP document.
- The ***SOAP header and body*** are syntactically similar. SOAP 1.1 and SOAP 1.2 have no conventions for what is supposed to be in the header; it is simply a place to put directives to the SOAP processor that receives the message. The sending and receiving parties need to agree on which elements go there and what they mean.



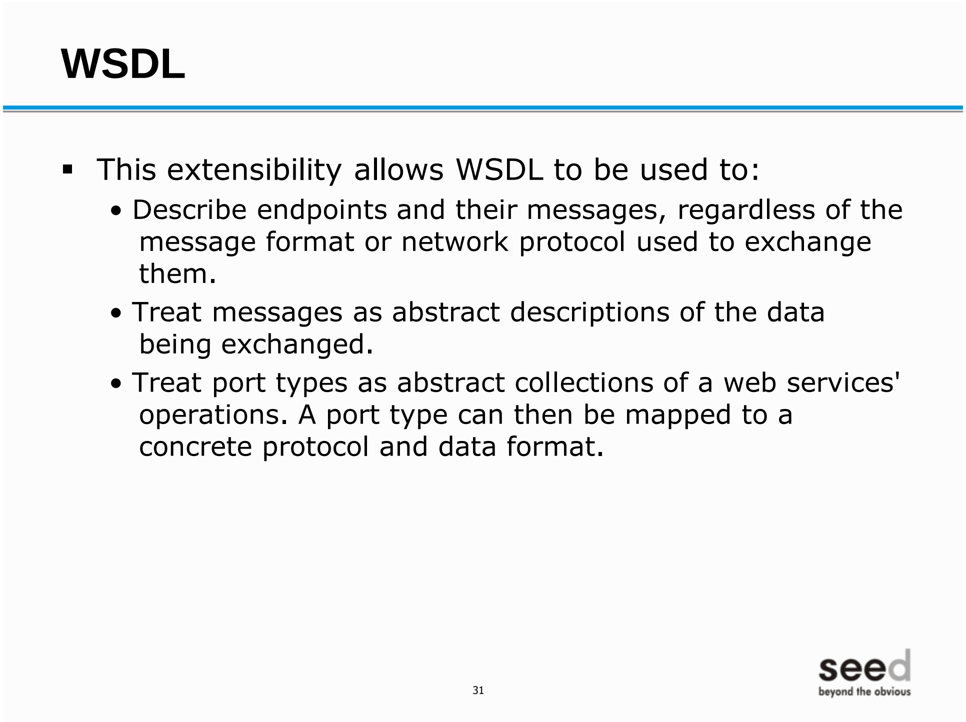
SOAP

- While XML and SOAP are very good at describing data, many kinds of application data aren't well-suited for XML—for example, a piece of binary data such as an image, or a CAD file that contains schematic diagrams of parts being ordered electronically.
- SOAP with Attachments (SwA) was born in recognition of this limitation.
- SwA combines the SOAP protocol with the MIME format to allow any arbitrary data to be included as part of a SOAP message.



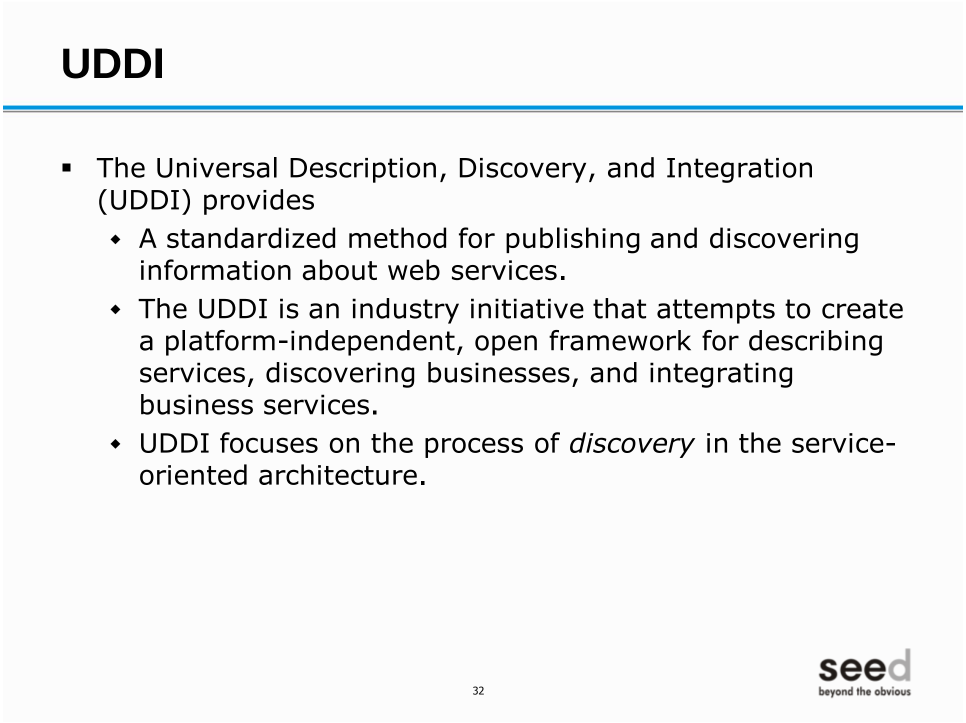
WSDL

- WSDL is an XML grammar for describing a web service as a collection of access ***endpoints*** (URLs to which service requests are sent) capable of exchanging messages in a procedure- or document-oriented fashion.
- A WSDL document is a recipe used to ***automate*** the details involved in application-to-application communication.
- On one level, WSDL is not that different from CORBA IDL or Microsoft IDL. They are all used to define the interfaces (method signatures) and data types for a discreet piece of programming logic
- On another level, WSDL is an altogether different beast, offering a degree of extensibility that has no parallel in the IDL specification.



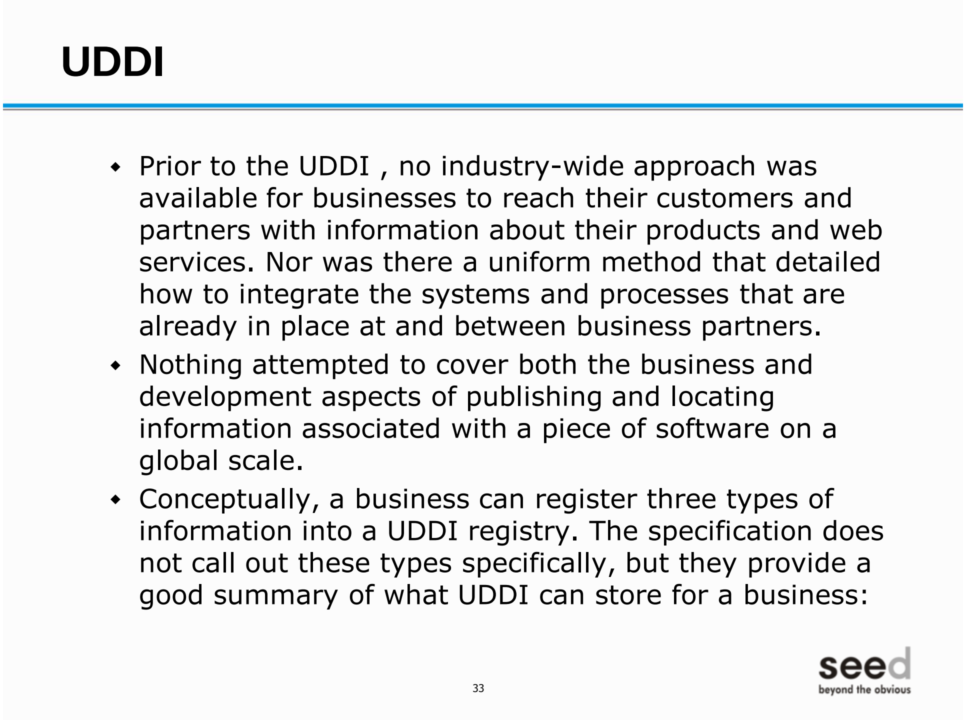
WSDL

- This extensibility allows WSDL to be used to:
  - Describe endpoints and their messages, regardless of the message format or network protocol used to exchange them.
  - Treat messages as abstract descriptions of the data being exchanged.
  - Treat port types as abstract collections of a web services' operations. A port type can then be mapped to a concrete protocol and data format.



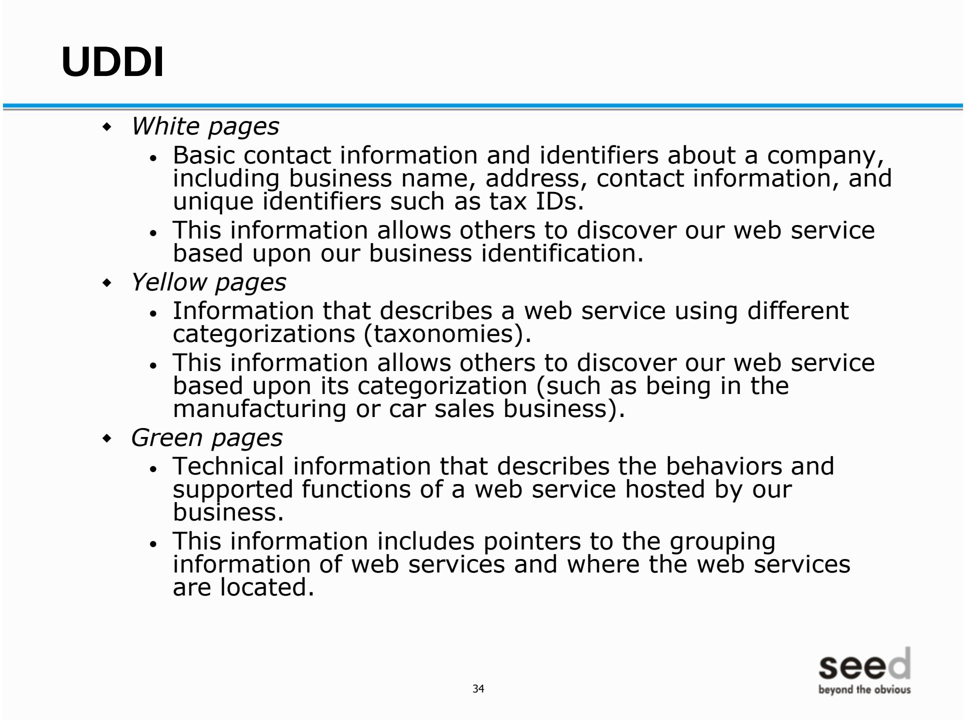
UDDI

- The Universal Description, Discovery, and Integration (UDDI) provides
  - A standardized method for publishing and discovering information about web services.
  - The UDDI is an industry initiative that attempts to create a platform-independent, open framework for describing services, discovering businesses, and integrating business services.
  - UDDI focuses on the process of *discovery* in the service-oriented architecture.



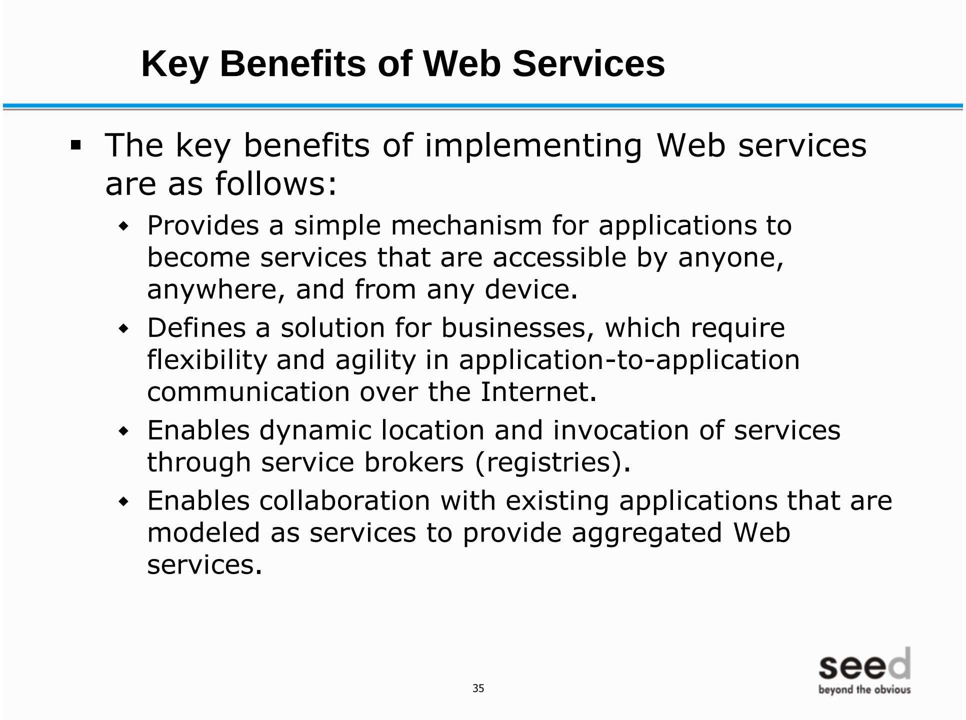
UDDI

- Prior to the UDDI , no industry-wide approach was available for businesses to reach their customers and partners with information about their products and web services. Nor was there a uniform method that detailed how to integrate the systems and processes that are already in place at and between business partners.
- Nothing attempted to cover both the business and development aspects of publishing and locating information associated with a piece of software on a global scale.
- Conceptually, a business can register three types of information into a UDDI registry. The specification does not call out these types specifically, but they provide a good summary of what UDDI can store for a business:



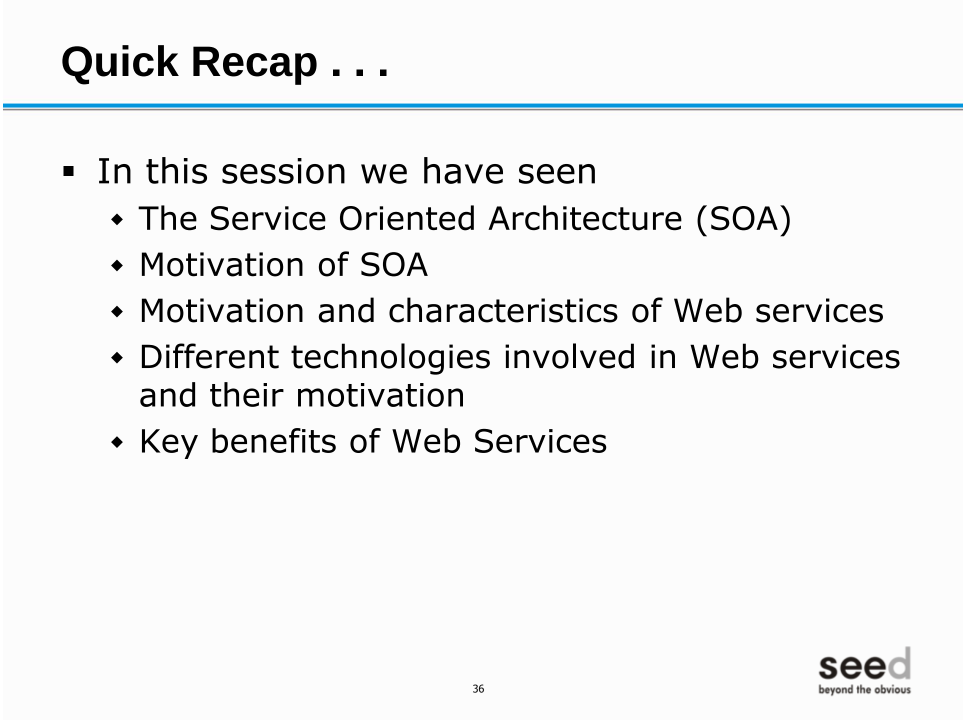
UDDI

- *White pages*
  - Basic contact information and identifiers about a company, including business name, address, contact information, and unique identifiers such as tax IDs.
  - This information allows others to discover our web service based upon our business identification.
- *Yellow pages*
  - Information that describes a web service using different categorizations (taxonomies).
  - This information allows others to discover our web service based upon its categorization (such as being in the manufacturing or car sales business).
- *Green pages*
  - Technical information that describes the behaviors and supported functions of a web service hosted by our business.
  - This information includes pointers to the grouping information of web services and where the web services are located.



Key Benefits of Web Services

- The key benefits of implementing Web services are as follows:
  - Provides a simple mechanism for applications to become services that are accessible by anyone, anywhere, and from any device.
  - Defines a solution for businesses, which require flexibility and agility in application-to-application communication over the Internet.
  - Enables dynamic location and invocation of services through service brokers (registries).
  - Enables collaboration with existing applications that are modeled as services to provide aggregated Web services.



Quick Recap . . .

- In this session we have seen
  - The Service Oriented Architecture (SOA)
  - Motivation of SOA
  - Motivation and characteristics of Web services
  - Different technologies involved in Web services and their motivation
  - Key benefits of Web Services