Distributed Systems – Techs
4. Web Services

W3C Definitions

- WS is a software system designed to support *interoperable machine to machine interaction* over a network
- software system
  - identified by a URI (accessible on the Web through a URL)
  - whose public interfaces and bindings are defined and described using XML artifacts (such as Web Services Definition Language - WSDL)
- other software systems
  - can discover its definition & may then interact with the Web service in a manner prescribed by its definition
  - access it through XML-based protocols such as Simple Object Access Protocol (SOAP) sent over accepted Internet protocols, such as HTTP

WS Arch. (WSA) as Distrib.Comput.Arch. (DCA)

- purpose of a DCA: enable programs in one environment to communicate and share data/content with programs in another environment.
- Classic: programmers have had to
  - tell one application program where to go to find another cooperative program - known as "tightly coupling" applications.
  - maintain these programmatic links over the useful life of the applications that they have written.
  - Creating these "hard-wired" links is complicated and cumbersome
- WSA - A new DCA ?
  - the applications themselves could automatically find cooperative programs to work with
  - program-to-program communications process – called "loosely coupled."

Definition by Gartner research - 2001

- Def: WSs are loosely coupled software components delivered over Internet standard technologies.
- WSs are self-describing and modular business appls…
  - … 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.
- WSs can be developed using any programming language, any protocol, or any platform

Historical evolution: the beginning

- late 1990s, Microsoft & a couple of other companies were thinking about an XML-based RPC that could work over HTTP.
  - Term *SOAP* was coined in 1998.
  - First versions of SOAP 1.0 published in Dec 1999.
  - Support from both the commercial and Open Source community -> 2001, SOAP 1.2.

- Microsoft, IBM, and Sun Microsystems are pushing WSs as the next great technology to allow developers to create remote objects easily.
  - Earlier remote object technologies, such as COM+ and CORBA, were difficult to implement and had high maintenance costs.

Historical evolution: WS as result of Web evolution

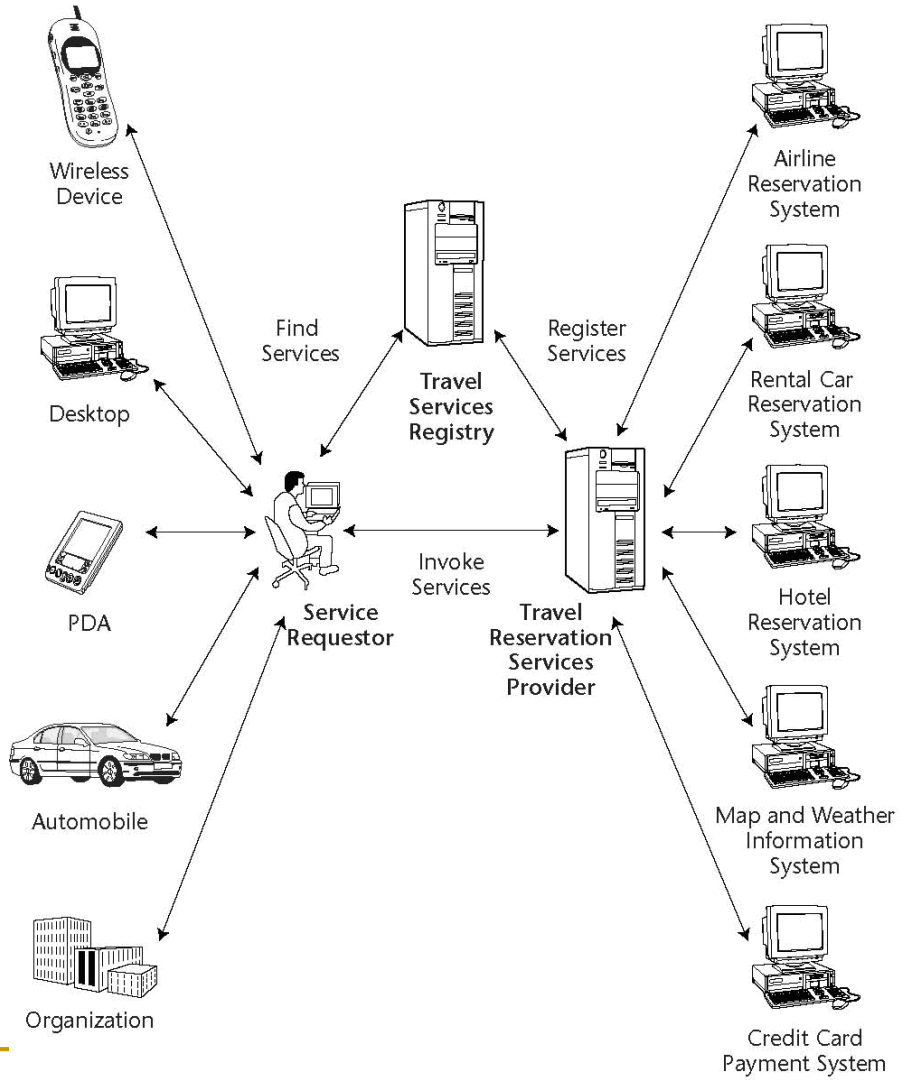
1. Initially, the Web consisted of sites that were plain HTML pages.
2. Later, Web appls. dynamically generated these HTML pages.
   - Web appls. are still limited to the GUI capabilities of their HTML pages
   - Enable interaction between and end user and a Web site
3. WSs go beyond this limitation, since they separate the Web site or application (the service) from its HTML GUI.
   - the service is represented in XML and available via the Web as XML.
   - Enable the application-to-application communication over the Internet
- Map Web site example:
  1. Web site: only static links to maps of various cities and locales
  2. Web appl: provide driving directions, customized maps, etc
  3. Extend functionality to provide a Web service: other enterprises can use to provide directions to their own office locations integrate with global position systems

Technical reasons for choosing Web services over Web appls

- WSs can be invoked through XML-based RPC mechanisms across firewalls.
- WSs provide a cross-platform, cross-language solution based on XML messaging.
- WSs facilitate ease of application integration using a light-weight infrastructure without affecting scalability.
- WSs enable interoperability among heterogeneous applications.

Example: the travel reservation services

- Expose business appls as Web services
- Supporting a variety of customers and application clients
- Business appls are provided by different travel organizations residing at different networks and geographical locations



Reusable objects as a fundamental concept in WSs

Example:

- a programmer write ONCE a calculator program as a WS
- Available for (bolted-onto, coupled or reused):
  - a spreadsheet program,
  - a customized transaction program,
  - a mortgage amortization program, or
  - any other program that could logically make use of a calculator.

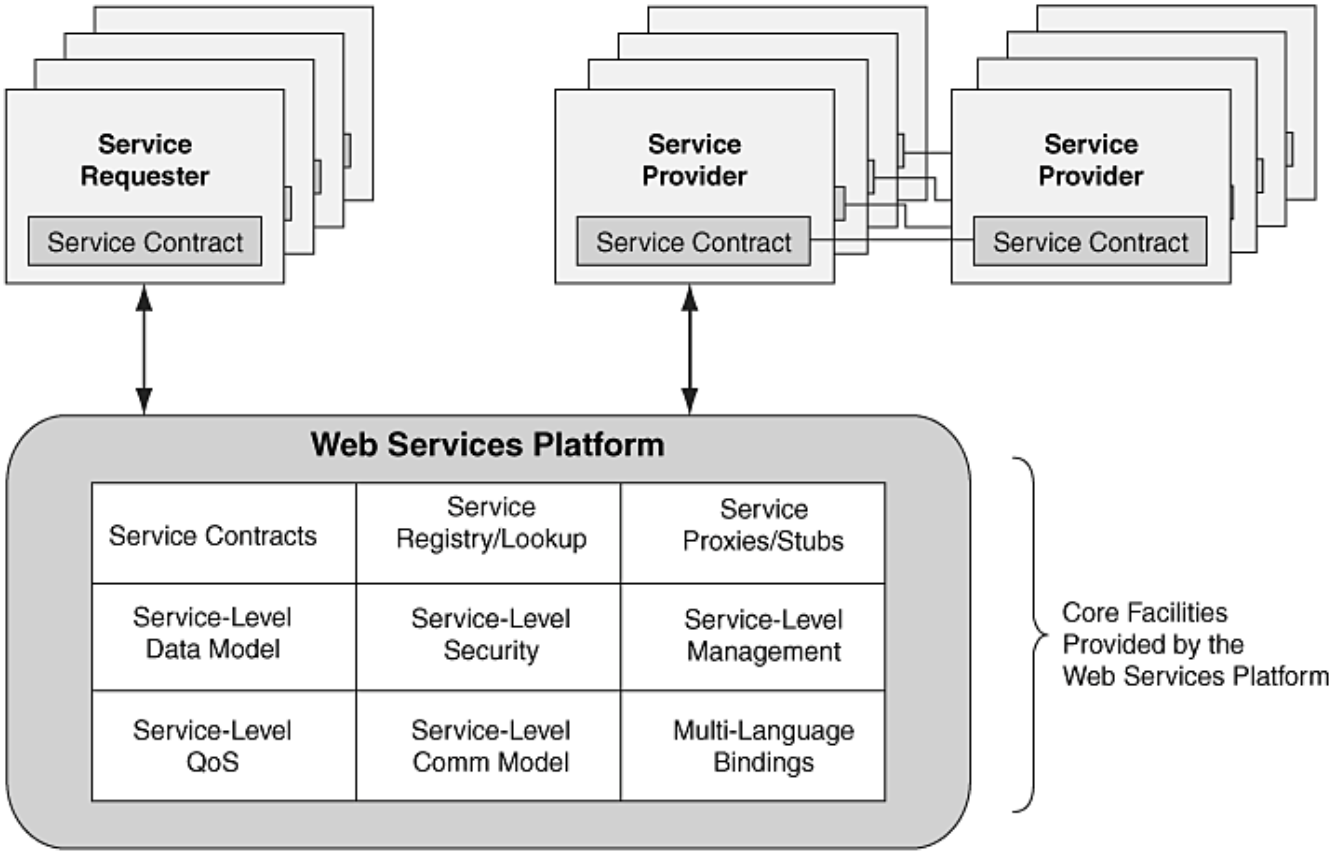
Realm of WS

- ! software components that are programmatically accessible over standard Internet protocols
  - “The Internet is the OS”!
- WSs expose a standard interface that is platform and technology independent.
- Context: the growing need for application-to-application communication and interoperability.
  - WSs provide a means of communication among software applications
    - running on different platforms and
    - written in different application development languages
    - present dynamic context-driven information to the user.

Key benefits of WS

- Interoperability in a heterogeneous environment
- Business services through the Web
- Integration with existing systems
- Freedom of choice
- Support more client types
- Programming productivity

Elements of a WS platform (1/4)



Elements of a WS Platform (2/4)

1. ***Service contract***
   - unambiguous, well-defined service interface using WSDL.
   - should be human-readable and machine-readable.
2. ***Service contract repository***
   - Might include taxonomies in UDDI or another registry to categorize services and search Should be highly available and replicated.
3. ***Service registration and lookup***
   - A naming service for locating service instances and run-time resources
   - Whereas the service contract repository is used to look up service contracts, service registration & lookup is used for finding run-time instances of the services.
4. ***Service-level security***
   - authenticating service requesters,
   - role-based access control,
   - single-sign-on, privacy, integrity, non-repudiation.

Elements of a WS Platform (3/4)

5. ***Service-level data management***
   - Usage of XML Schema for data validation, data transformation, mapping data between different message structures including data filtering or data aggregation
6. ***Service-level communication***
   - Support for multiple interaction patterns and communication styles using SOAP.
7. ***Multiple protocol and transport support***
   - Messaging infrastructure should support multiple transports/protocols to support the wide range of clients, servers, and platforms.
8. ***Service-level qualities of service***
   - Support for message-ordering, guaranteed delivery or at-most-once delivery
   - Transaction management capabilities for defining and supporting transaction execution and control including two-phase commit
   - High-availability capabilities include clustering, failover, automatic-restart, load balancing, and hot-deployment of services.

Elements of a WS Platform (4/4)

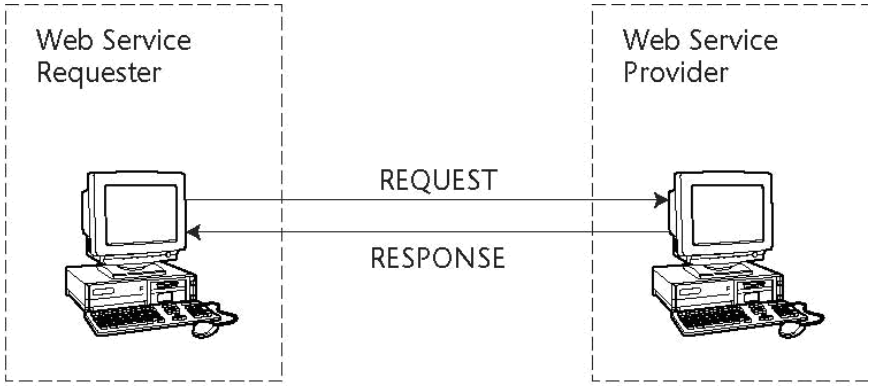
9. ***Service-level management***
   - Support for deploying, starting, stopping, and monitoring services.
   - Support for versioning services.
   - Support for auditing service usage.
   - Support for metering and billing for service usage.
   - Service monitoring, service status, service responsiveness, and compliance or deviations from service-level agreements.
10. ***Support for multiple programming languages***
   - Support for generating service proxies and service skeletons for all supported programming languages.
11. ***Service programming interfaces***
   - Provide service programming interfaces so that developers can access the facilities of the WSs platform from their favorite programming language(s)

Types of WS Architectures (WSAs)

- Differences in how they do their jobs
- Most common WSA:
  1. Remote Procedure Call (RPC)
     - XML-RPC provides a basic set of tools for creating cross-platform RPC calls, using HTTP as a foundation
     - WSs encapsulate RPC with XML as the data packaging.
  2. Service-Oriented Architecture (SOA)
     - Combining SOA techs with WSs basically gives:
       - the WSs protocol stack,
       - a collection of network protocols that are used to define and implement how WSs interact with one another.
  3. Representational State Transfer (REST)

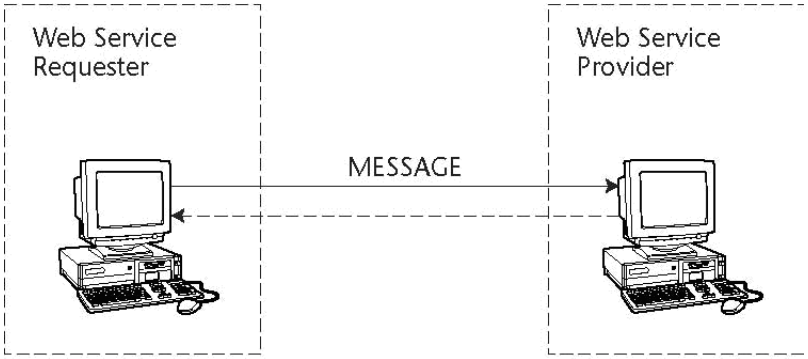
WS Communication Models: 1- RPC model

- defines a request/response-based synchronous communication.
- RPC-based Web services are tightly coupled and are implemented with remote objects to the client appl.
- Both the service provider and requestor can register and discover services



WS Communic. models – 2 Messasing-based

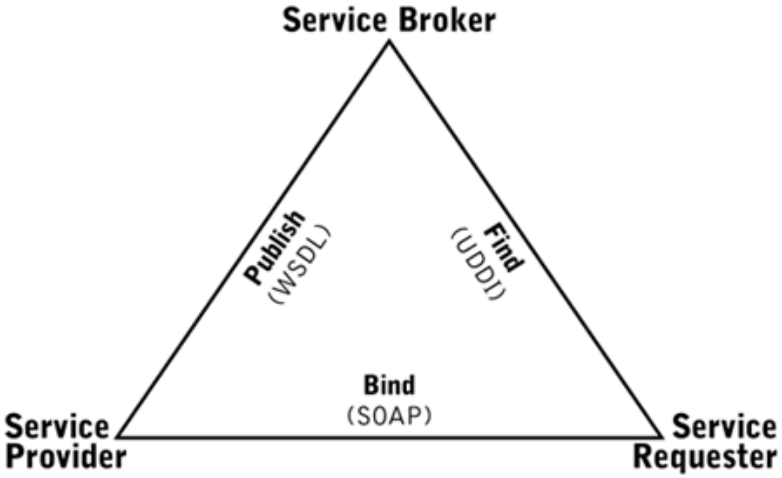
- defines a loosely coupled and document-driven communication
- the service requestor does not wait for a response.
- the client sends an entire document rather than sending a set of parameters.
- the service provider may or may not return a message.

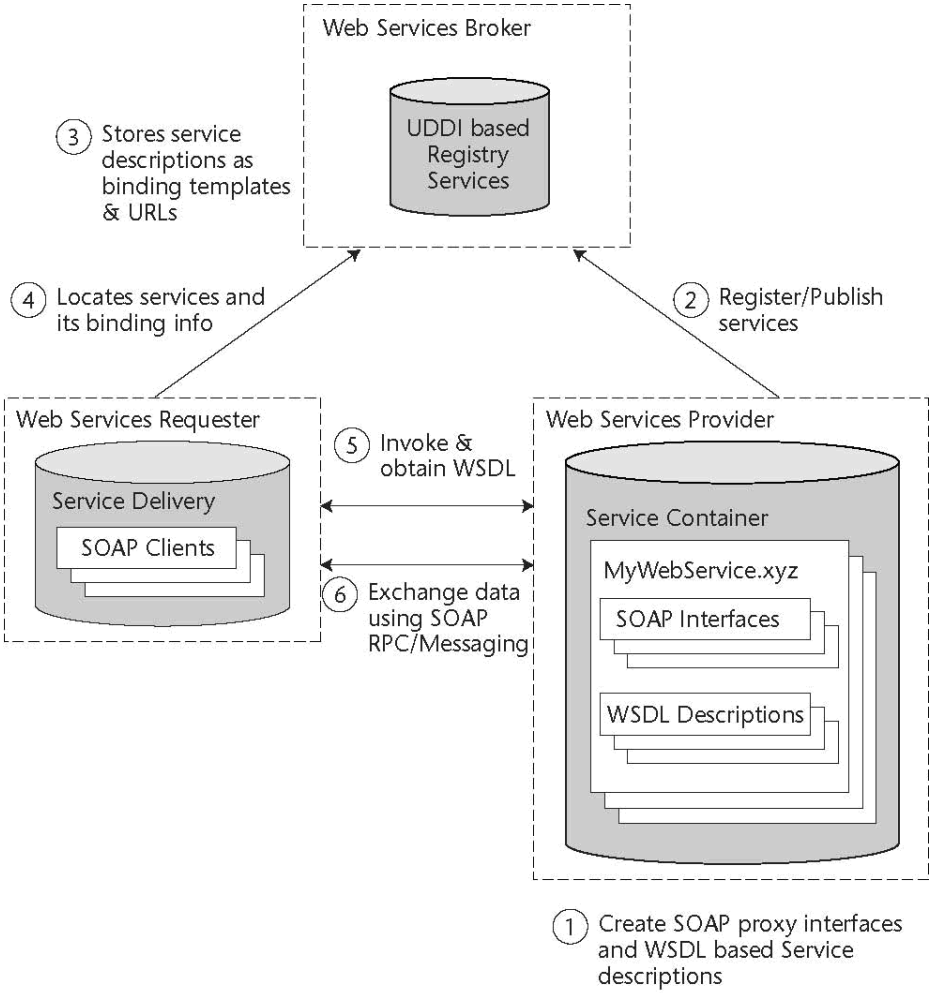


- Adopting a communication model also depends upon the WS provider infrastructure and its compliant protocol for RPC or Messaging.
- The current version of SOAP and ebXML Messaging support these communication models;

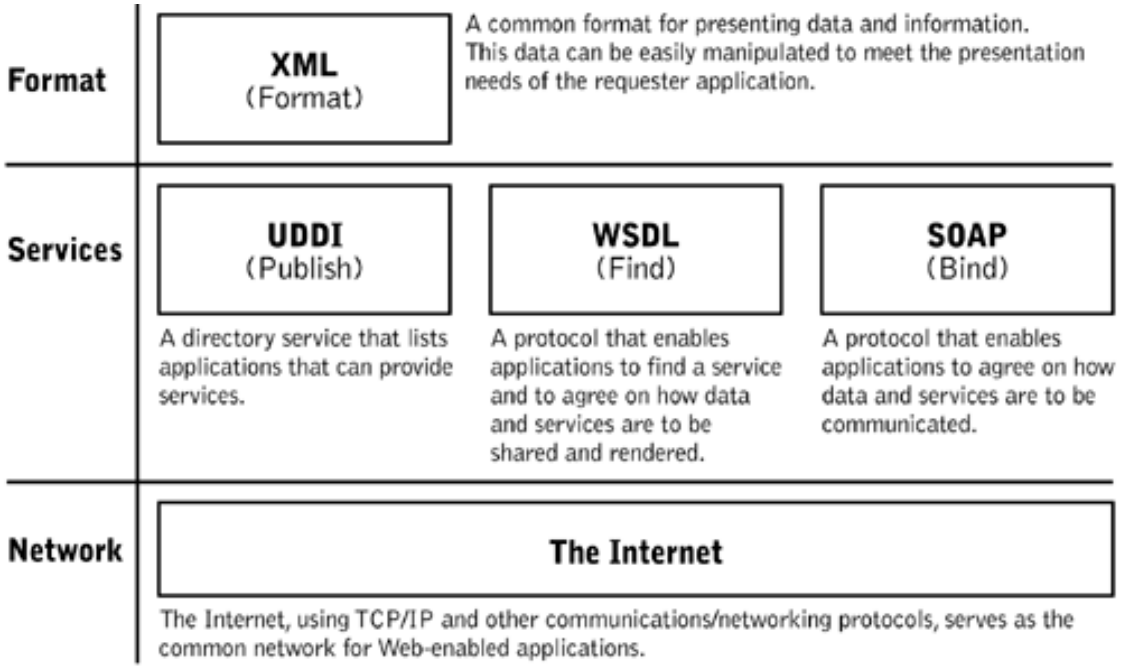
How Web services work

- Publish, find, and bind.





WS standards (see next lecture!)



Needs:

- Common markup language for communication
- Common message format for exchanging information
- Common service specification formats
- Common means for service lookup

Web Services Protocol Stack- 4 basic levels

- Service Transport:
  - HTTP or HTTPS, SMTP, and FTP.
- Service Messaging
  - XML-RPC and SOAP.
- Service Description
  - WSDL format is usually used.
- Service Discovery
  - the UDDI protocol is used for this purpose.

Service binding

- Is different for an SOA based on WS compared to an SOA based on J2EE or CORBA:
  - J2EE, CORBA: binding via reference pointers or names,
  - WSs bind using discovery of services, which may be dynamic.
- If the service requester can understand
  - the WSDL and
  - associated policy files supplied by the provider,

  SOAP messages can be generated dynamically to execute the provider's service.

WS Implementations

- 2 major technological camps in the WSs industry.
  1. Microsoft.
     - got a head start because its people developed the *SOAP* standard and then gave it to the open-source community.
     - before other developers knew about the *SOAP* standard, Microsoft had already begun developing programming languages such as C# and *Visual Basic.NET* to create a proprietary implementation.
  2. Revolves around Java,
     - E.g. Sun Microsystems, creator of Java, deploying technologies.
     - Other vendors include BEA, Cape Clear Software, IBM, etc
     - Several WS libraries that are free and easily downloaded e.g. from Sun and the Apache group.
- Third-party products available that allow WSs to integrate with CORBA, COBOL, C++, and other legacy systems.

Microsoft .NET implementation

- Provides a large number of tools to make the creation and use of WSs very easy.
- includes
  - the automatic generation of WSDL,
  - discovery tools such as disco (which searches servers that have *.NET* Web services),
  - browser-based testing and discovery of methods, and
  - easy creation of WSs within Microsoft's proprietary languages.
- But all of the Microsoft WS technologies rely on their Web server *Internet Information Server* (IIS).
  - Has security problems.

Java implementation

- Sun and IBM deliver WS products.
- Apache group provides a great free *SOAP* library to access WSs.
- Advantages to using WSs with Java.
  - several different vendors implement WSs with Java
  - Java WSs work on top of
    - both Java Server Pages (JSP) and servlets,
    - *Tomcat* which is a free Java server that integrates easily with *Apache*, is free from the *Apache* group.
    - IBM's *Websphere*, BEA's *WebLogic*, and Sun's *iPlanet* server are all commercially available WS that allow a developer more options when deploying WSs.

Different implementation in Java

- SUN SDK: starts from an interface

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface AccountIF
    extends Remote {
   public void deposit (int amount)
    throws RemoteException;
   public void withdraw (int
    amount) throws
    RemoteException;
   public int balance () throws
    RemoteException;
}
```

- BEA's WebLogic Server product: starts from a Java class

```
public class Account implements
    com.bea.jws.WebService
 {
   static final long
   serialVersionUID = 1L;
 /**
 * @common:operation
 */
 public void deposit (int amount);
   {
  ...
```