Copyright Notice

Service Oriented Architecture and Web services

Prakash Badhe

prakash.badhe@vishwasoft.in

About Me – Prakash Badhe

- ✓Founder, CEO - VishwaSoft Technologies
- ✓Enjoying the IT for 19+ years
- ✓Web Technologies consultant and mentor
- ✓Passion for technologies and frameworks
- ✓Worked on medium and large sized dynamic web projects.
- ✓Proficient with Internet standards
- ✓Skilled in java , JavaEE technologies and frameworks
- ✓ Working with Apache CXF, Oracle ESB, Spring Integration, Apache Camel etc.

About you..

- **Prerequisite** : Awareness about internet application development and java programming skills along with knowledge about internet application standards.
- Job Role and skill-Sets
- Training Objectives
- Experience/knowledge in Java Web application development.
- Awareness about SOA and web services development.

Agenda

- ❖SOA Introduction
- ❖Web Service introduction
- ❖Web service technology components and frameworks
- ❖SOAP, WSDL review
- ❖Apache CXF Web services framework
- ❖REST Web Services
- ❖REST with CRUD API
- ❖Micro Services
- ❖Spring and Hibernate Framework
- ❖Spring-Boot for Micro-services
- ❖Agile Development Practices TDD and BDD

Setup

- Windows 7/8/10 64 bit
- Adobe PDF Reader
- JDK1.8 64 bit
- Eclipse-Jee-Photon
- MySql/Oracle RDBMS with client
- Apache Tomcat 8.0 Web Server
- Google Chrome web browser
- Apache CXF and TcpMon
- RestEasy Framework.
- Spring and Hibernate
- Spring Boot and Apache Maven
- Live Internet connection

SOA : Service Oriented Architecture

Why SOA..?

- Distributed Processing..
- Applications need data sharing and communication in the network across different machines.
- Heterogeneous applications with different programming and different Operating systems.
- Lots of data compatibility issues
- Standard protocol and data format needed.

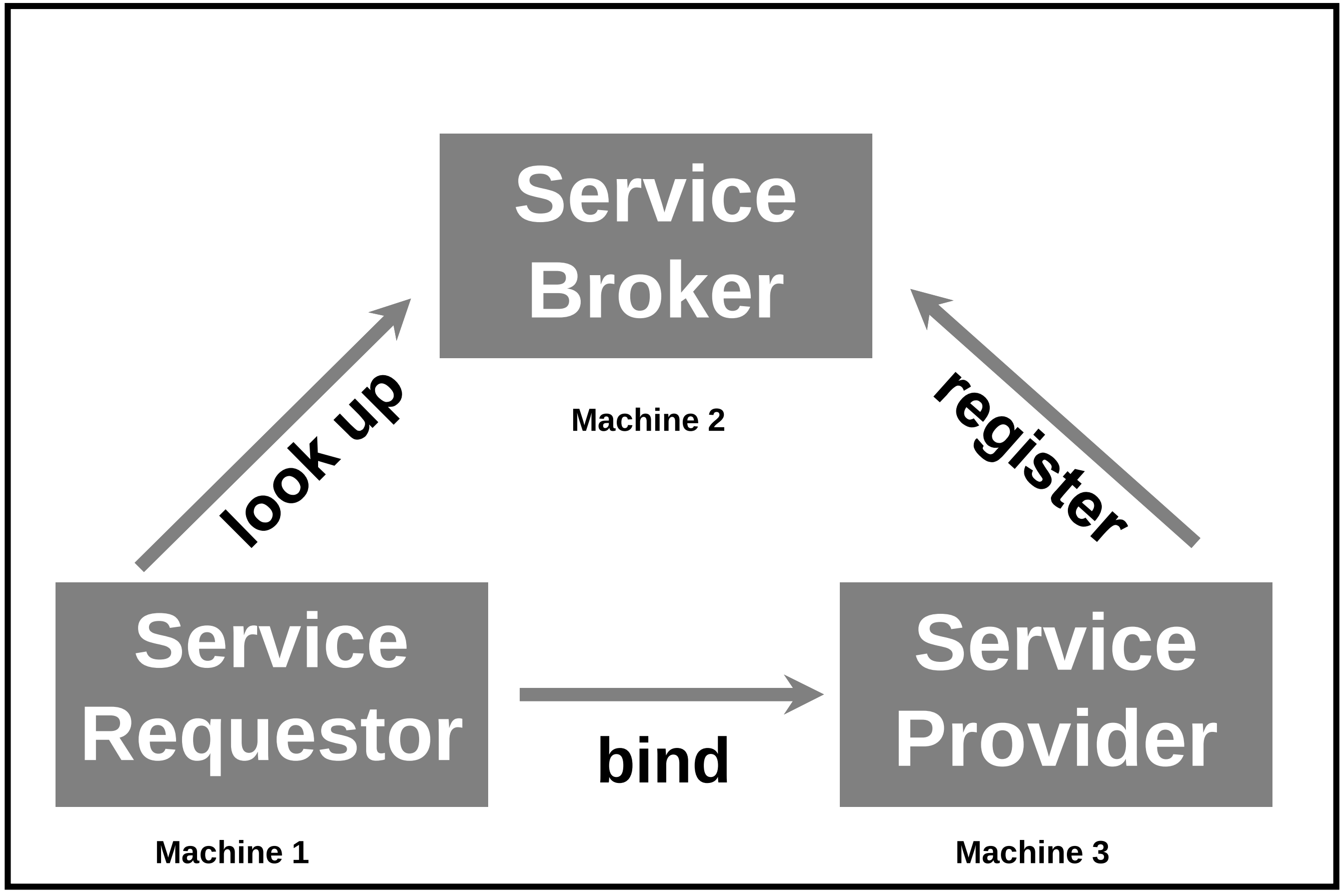
Service Oriented Architecture

- In a service-oriented architecture, applications are made up from loosely coupled software services, which interact to provide all the functionality needed by the application.
- The applications exchange information across the platforms.
- Each service is generally designed to be very self-contained and stateless to simplify the communication that takes place between them.

Roles Involved In A SOA

- There are three main roles involved in a service-oriented architecture:
  - **Service provider**
  - **Service broker**
  - **Service requestor**

SOA Communication Model



Service Oriented Architecture

Service Provider

- The *service provider* creates a service and may publish its interface and access information to a *service broker*.
- A service provider must decide which services to expose and how to expose them.
- There is often a trade-off between security and interoperability; the service provider must make technology decisions based on this trade-off.
- If the service provider is using a service broker, decisions must be made on how to categorize the service, and the service must be registered with the service broker using agreed-upon protocols.

Service Broker

- The *service broker,* also known as the *service registry*, is responsible for making the service interface and implementation access information available to any potential service requester.
- The service broker will provide mechanisms for registering and finding services.
- A particular broker might be public (for example, available on the Internet) or private—only available to a limited audience (for example, on an intranet).
- The type and format of the information stored by a broker and the access mechanisms used will be implementation-dependent.

Service Requestor

- The *service requestor* locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its Web services.
- One important issue for users of services is the degree to which services are statically chosen by designers compared to those dynamically chosen at runtime. Even if most initial usage is largely static, any dynamic choice opens up the issues of how to choose the best service provider and how to assess quality of service.
- Another issue is how the user of services can assess the risk of exposure to failures of service suppliers.

SOA Characteristics - I

- The service-oriented architecture employs a loose coupling between the participants. Such a loose coupling provides greater flexibility:
- In this architecture, a client is not coupled to a server, but to a service. Thus, the integration of the server to use takes place outside of the scope of the client application programs.
- Old and new functional blocks are encapsulated into components that work as services.

SOA Characteristics - II

- Functional components and their interfaces are separated. Therefore, new interfaces can be plugged in more easily.
- Within complex applications, the control of business processes can be isolated. A business rule engine can be incorporated to control the workflow of a defined business process. Depending on the state of the workflow, the engine calls the respective services.
- Services can be incorporated dynamically during runtime.
- Bindings are specified using configuration files and can thus easily be adapted to new needs.

SOA Requirements -I

For an efficient use of a service-oriented architecture, a number of requirements have to be fulfilled:

Interoperability between different systems and programming languages: The most important basis for a simple integration between applications on different platforms is a communication protocol that is available for most systems and programming languages.

SOA Requirements - II

- **Clear and unambiguous description language**
  - To use a service offered by a provider, it is not only necessary to be able to access the provider system, but also the syntax of the service interface must be clearly defined in a platform-independent fashion.

SOA Requirements-III

- **Retrieval of the service**
  - To allow a convenient integration at design time or even runtime of the system, we require a mechanism that provides search facilities to retrieve suitable available services.
  - Such services should be classified into computer-accessible, hierarchical categories, or taxonomies, based on what the services in each category do and how they can be invoked.

SOA Requirements-IV

- **Security**
  - Protection of the services, including the information passed to and received from the service against unauthorized and malicious access, must be supported by the platform to win the confidence of the requestor (chain)—at the end the business customers.

continued….

SOA Requirements-V

- The type and extent of security depends on the type and placement of the participants—service requestors and service providers—and the services themselves. Service usage monitoring and security incident action plans have to be in place to detect unauthorized access (attempts) and trigger counter measures. Security is required to empower and retain authenticated and authorized requestors/customers while fencing off everything and everyone.

Web services as SOA

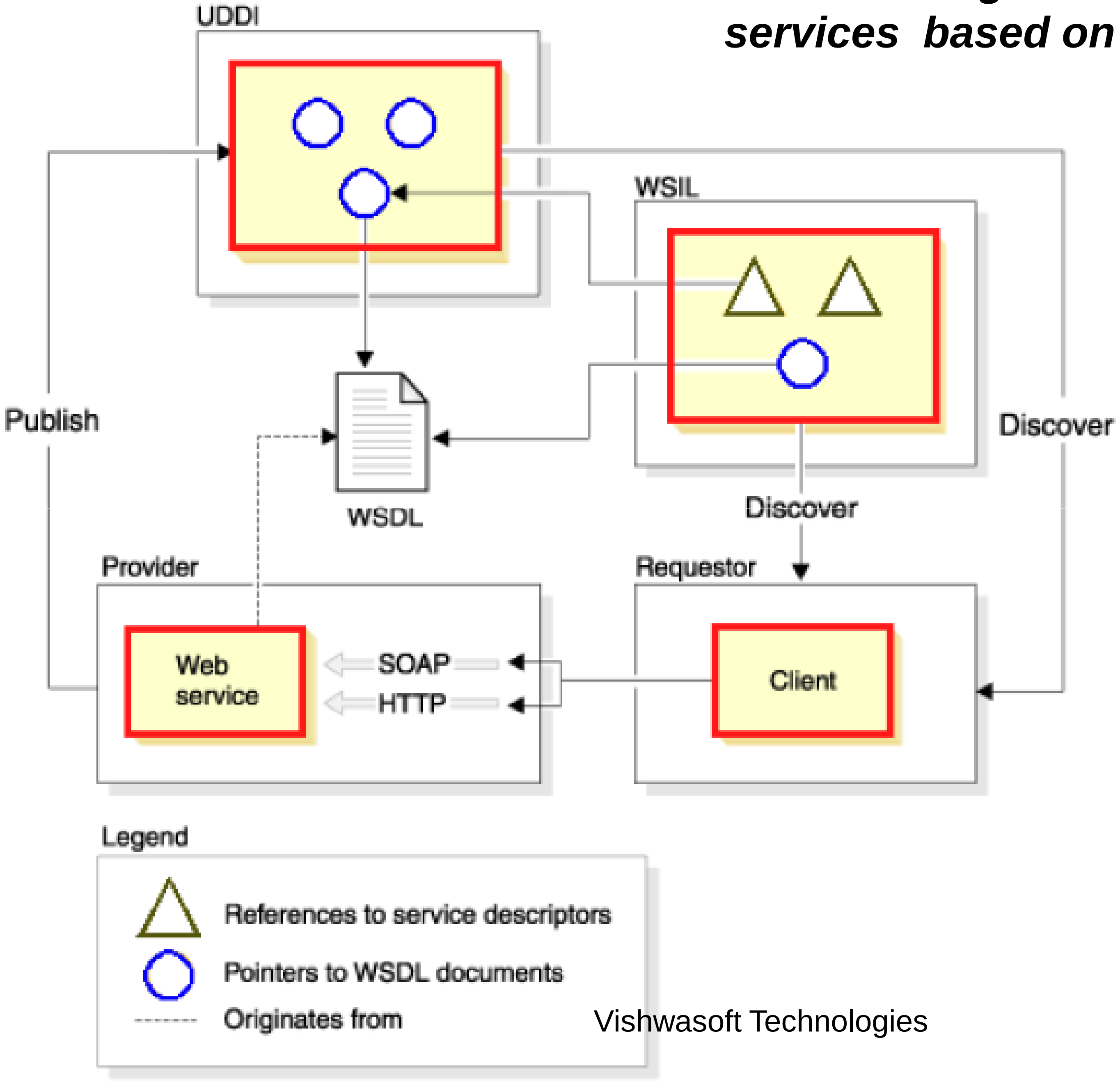
- Web services are a implements a service-oriented architecture.
- During the development of this standards, a major focus was put on making functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages.
- If we had to describe Web services using just one sentence, we would use the following:

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network.

Web services

- Web services perform encapsulated business functions, ranging from simple request-reply to full business process interactions.
- These services can be new applications or just wrapped around existing legacy systems to make them network-enabled.
- Services can rely on other services to achieve their goals.

Main building blocks in Web services based on SOA Architecture



Web services Features

Web services are self-contained.

On the client side, no additional software is required. A programming language with XML and HTTP client support is enough to get you started. On the server side, merely an HTTP server and a SOAP server are required.

Web services Features

- **Web services are self-describing**.

  The definition of the message format travels with the message; no external metadata repositories or code generation tools are required.

- **Web services can be published, located, and invoked across the Web**.

  This technology uses established lightweight Internet standards such as HTTP. It leverages the existing infrastructure.

  Some additional standards that are required to do so include SOAP, WSDL, and UDDI.

Web services Features

– **Web services are modular**.

Simple Web services can be aggregated to more complex ones, either using workflow techniques or by calling lower-layer Web services from a Web service implementation.

Web services can be chained together to perform higher-level business functions. This shortens development time and enables best-of-breed implementations.

Web services Features

- **Web services are language-independent and interoperable**.

  The client and server can be implemented in different environments. Existing code does not have to be changed in order to be Web service enabled. Basically, any language can be used to implement Web service clients and servers

- **Web services are inherently open and standard-based**.

  XML and HTTP are the major technical foundation for Web services. A large part of the Web service technology has been built using open-source projects. Therefore, vendor independence and interoperability are realistic goals.

Web services Features

– **Web services are loosely coupled**.

Traditionally, application design has depended on tight interconnections at both ends. Web services require a simpler level of coordination that allows a more flexible reconfiguration for an integration of the services.

– **Web services are dynamic**.

Dynamic e-business can become reality using Web services, because with UDDI and WSDL, the Web service description and discovery can be automated. In addition, Web services can be implemented and deployed without disturbing clients that use them.

Web services Features

- **Web services provide programmatic access**.

  The approach provides no graphical user interface; it operates at the code level. Service consumers have to know the interfaces to Web services but do not have to know the implementation details of services.

- **Web services provide the ability to wrap existing applications**.

  Already existing stand-alone applications can easily be integrated into the service-oriented architecture by implementing a Web service as an interface.

Web services Features

- **Web services build on proven, mature technology**.

  There are a lot of commonalities, as well as a few fundamental differences, with other distributed computing frameworks. For example, the transport protocol is text based and not binary.

Web services Technologies

- The following are the core technologies used for Web services:
  - **XML** (Extensible Markup Language) is the markup language that underlies most of the specifications used for Web services. XML is a generic language that can be used to describe any kind of content in a structured way, separated from its presentation to a specific device.
  - **SOAP** (Simple Object Access Protocol) is a network, transport, and programming language and platform-neutral protocol that allows a client to call a remote service. The message format is XML.

Web services Technologies-II

- **WSDL** (Web Services Description Language) is an XML-based interface and implementation description language. The service provider uses a WSDL document in order to specify the operations a Web service provides and the parameters and data types of these operations. A WSDL document also contains the service access information.

- **WSIL** (Web Services Inspection Language) is an XML-based specification about how to locate Web services without the necessity of using UDDI. However, WSIL can be also used together with UDDI, that is, it is orthogonal to UDDI and does not replace it.

Web services Technologies -III

- **UDDI** (Universal Description, Discovery, and Integration) is both a client-side API and a SOAP-based server implementation that can be used to store and retrieve information on service providers and Web services.

Summary

- Here, we introduced service-oriented architecture and Web services
- Web services are self-contained, modular applications that can be described, published, located, and invoked over a network.
- We described the basic standards that are used in building service-oriented solutions and their relation to service-oriented architectures.