WEB SERVICES

Outline

Why are we here?

- "Web Services" contain the word "Web" and Web technologies matter.

  More and more people get connected using computers, televisions, mobiles, ...

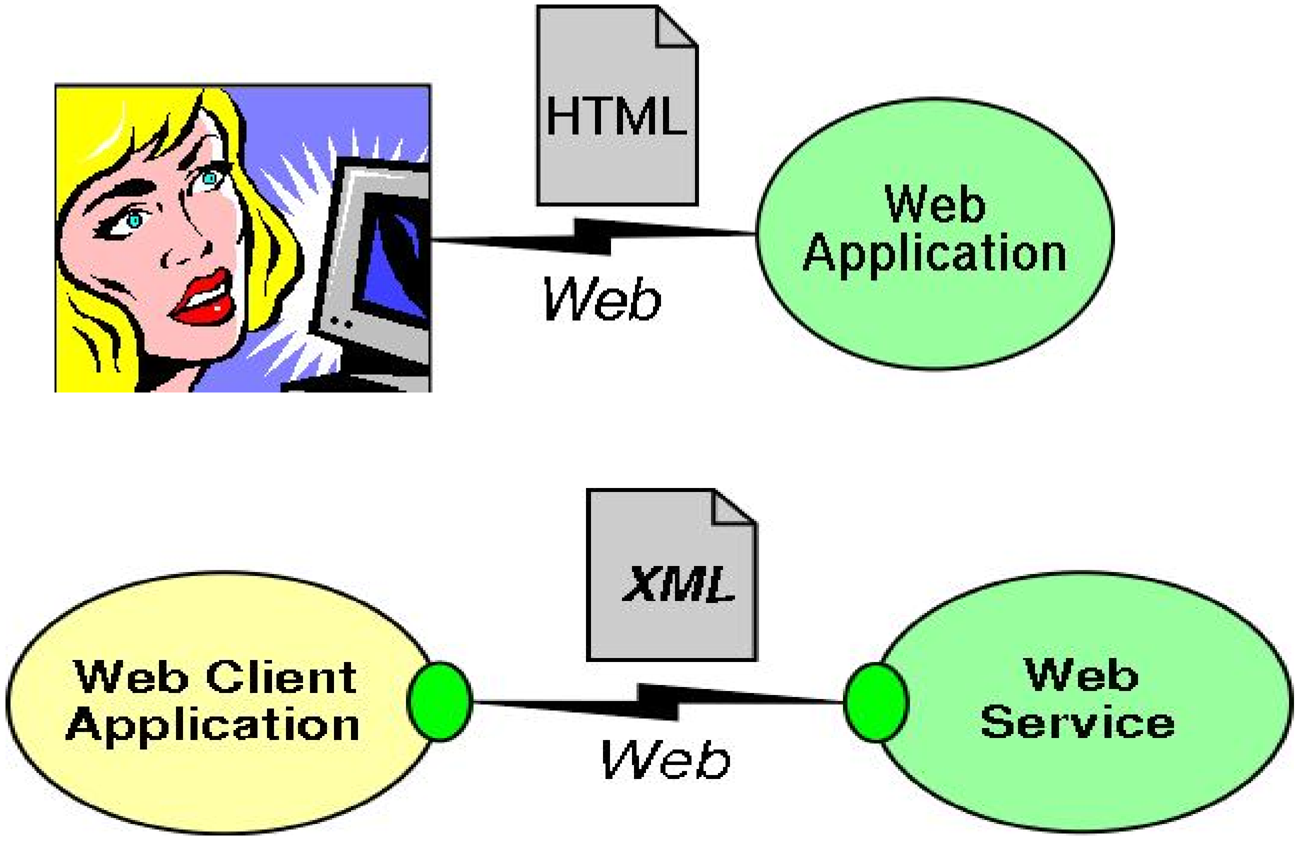
- "Web Services" contain the word "services" and we wonder what kind of services they will fulfill for us.

  We already have services on the Web, so what's missing?

The Problem (e.g. how to buy tomatoes)

- **Find the tomato sellers;**
  Yellow Pages: contain companies that are selling tomatoes, their location, and contact information.
- **Find the service offered according to my needs;**
  Where, when and how can I buy tomatoes?
- **Buy the tomatoes;**
  do the transaction

Traditional web application / Web service



Web Service Requirement

- **Applications need to communicate:**

  -Unless tomato buyers can communicate to tomato sellers, there will be no business.

- **The system must scale:**

  -As many tomato buyers and sellers as possible.

  -Being able to use any future new kind of tomatoes or any kind of products (potatoes, video games, ...)

  -Add new layers: security!

- **The solution must be portable across environments:**

  -Not all tomato buyers and sellers will use the same system or the same language.

  Used by anyone, anywhere, and with any kind of devices.

Existing solutions

How can two (or more) applications communicate today?

- **Ad Hoc solutions:**
  Unless you want to buy your own tomatoes, it won't be useful.
- **Language-oriented solutions (Java RMI, ...)**
  Works well if the tomato seller uses your language
- **Service-oriented solutions (CORBA, DCE, DCOM)**
  And about the Web?

Web Service: definition

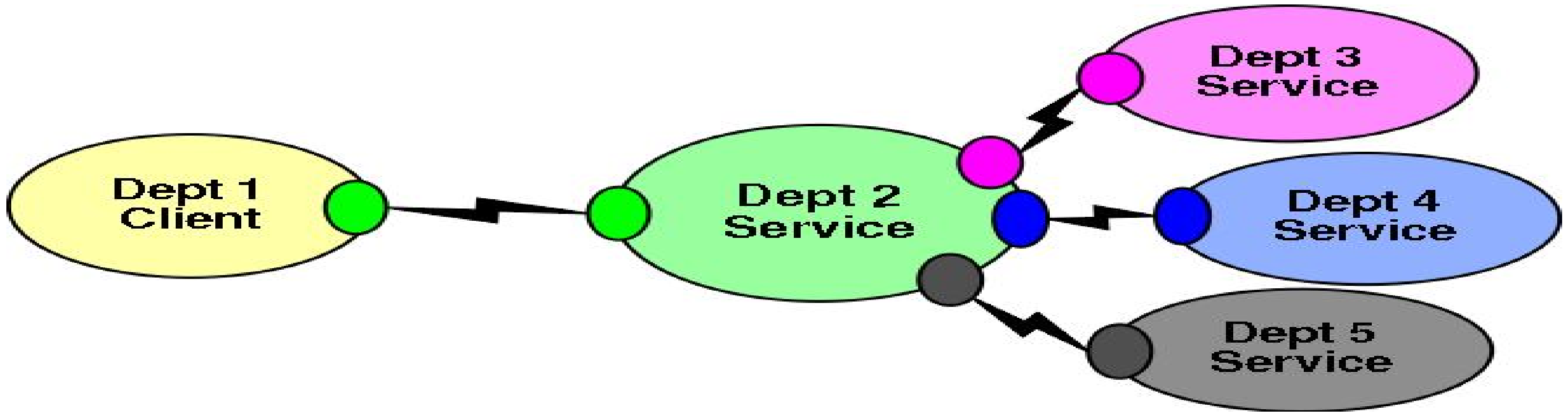
- An application component accessible via standard web protocols
- Client and server applications that communicate over the World Wide Web's (WWW) HyperText Transfer Protocol (HTTP)
- Loosely coupled, reusable software components that semantically encapsulate discrete functionality and are distributed and programmatically accessible over standard Internet protocols.
- The W3C defines a Web service as:

  "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards"

Benefits of Web service

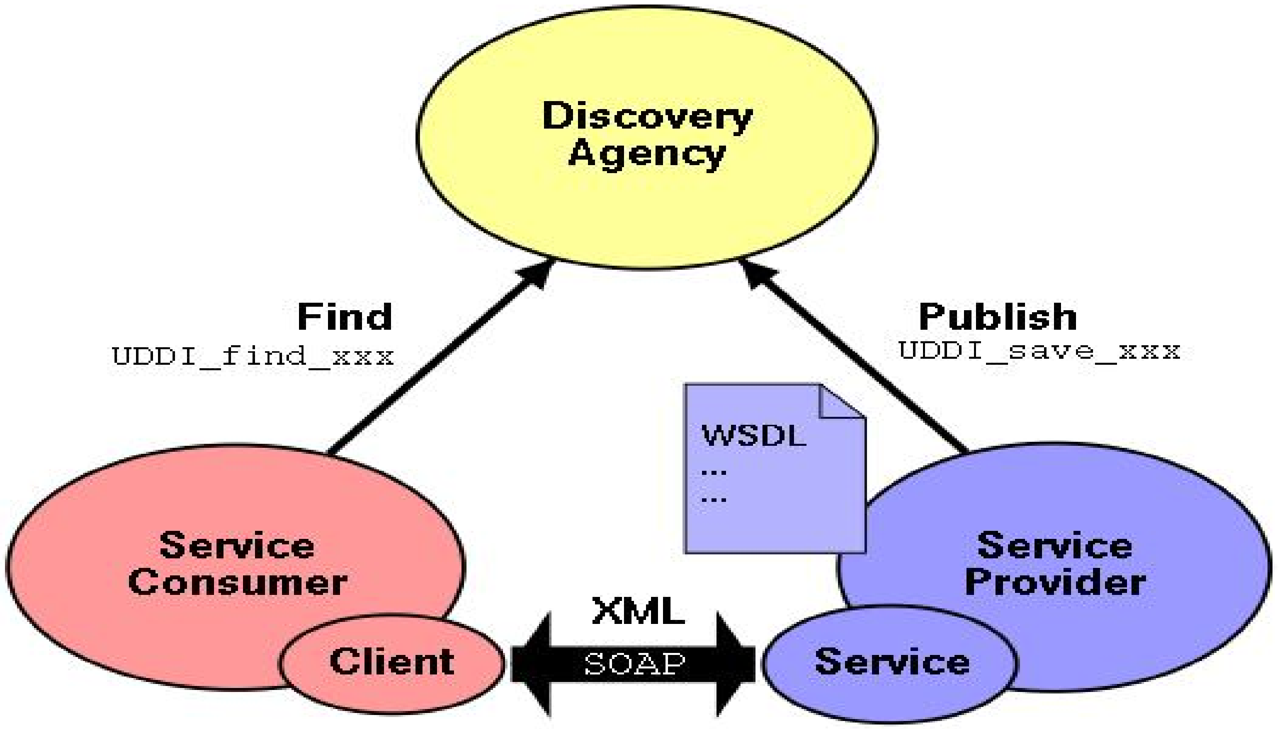
- Interoperability
- Usability
- Reusability
- Deployability
- Convergence of SOA (Service-Oriented Architecture) and Web.

Web services application



Can use Web Services to integrate across departments, agencies, to companies, etc.

Web service Architecture



An architecture view based on SOAP, WSDL, and UDDI.

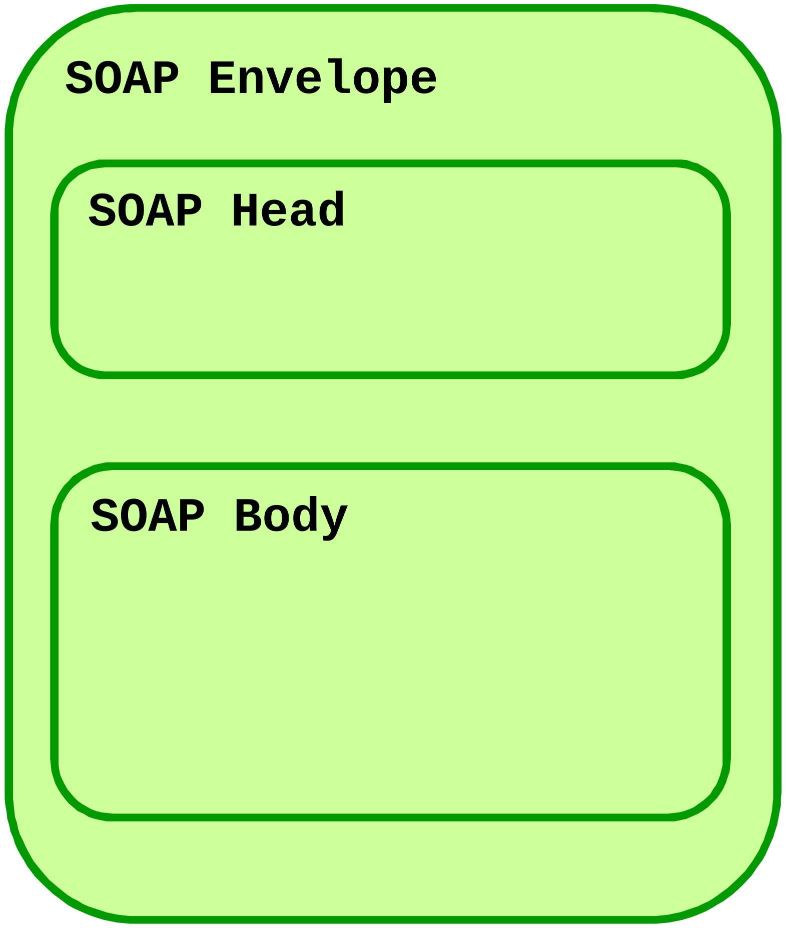
WS built on existing standards

- Extensible Markup Language (**XML**)
- The **HTTP** (Hypertext Transfer Protocol) standard is allowing more systems to communicate with one another.
- **SOAP** (Simple Object Access Protocol) (built on XML) standardizes the messaging capability on different systems.
- **WSDL** (Web Services Description Language) standardizes the description of Web services so providers and requesters are speaking the same language.
- **UDDI** (Universal Description, Discovery, and Integration) standardizes the publishing and finding of Web services.

SOAP (Simple Object Access Protocol)

- SOAP is an XML-based protocol from the W3C for exchanging data over HTTP
- Provides a simple, standards-based method for sending XML messages
- Elements
    - **Envelope** – specifies that the XML document is a SOAP message; encloses the message itself.
    - **Header** (optional) – contains information relevant to the message, e.g., the date the message was sent, authentication data, etc.
    - **Body** – includes the message payload.
    - **Fault** (optional) – carries information about a client or server error within a SOAP message.

SOAP-Packaging



WSDL (Web Services Description Language)

- WSDL is an XML-based format for describing
- Provide details about where to access web service, what operations can be performed, communication protocols and message format
- Elements
    - **Port type** – groups and describes the operations performed by the service through the defined interface.
    - **Port** – specifies an address for a binding, i.e., defines a communication port.
    - **Message** – describes the names and format of the messages supported by the service.
    - **Types** – defines the data types (as defined in an XML Schema) used by the service for sending messages between the client and server.
    - **Binding** – defines the communication protocols supported by the operations provided by the service.
    - **Service** – specifies the address (URL) for accessing the service.

WSDL (Web Services Description Language)

```
<definitions>

    <types>
      data type definitions........
    </types>

    <message>
      definition of the data being communicated....
    </message>

    <portType>
      set of operations......
    </portType>

    <binding>
      protocol and data format specification....
    </binding>

    <service>
      service definition with address (url) to access the service...
    </service>

</definitions>
```

UDDI (Universal Description, Discovery and Integration)

- UDDI is a standard sponsored by OASIS (Organization for the Advancement of Structured Information Standards).
- Specification for creating an XML-based registry that lists information about businesses and the Web services they offer
- Provides uniform way to list and discover the services
- Registering is optional step
- Registry can be public or private
- To search for a Web service, a developer can query a UDDI registry to obtain the WSDL for the service he/she wishes to utilize

JAX-WS

JAX-WS (Java API for XML based Web service)

- **Features and Developer Benefits** :

  - Portable and interoperable web services
  - Ease of development of web services endpoints & clients
  - Increased developer productivity
  - Support for open standards: XML, SOAP, WSDL
  - Standard API developed under Java Community Process
  - Support for tools
  - Support for SOAP message processing model & extensions
  - Secure web services
  - Extensible type mapping

JAX-WS vs JAX RPC

- JAX-WS 2.0 is successor of JAX-RPC

- JAX-WS support asynchronous & message oriented web service
but JAX-RPC does not

- JAX-WS binding data (data binding define in JAX-B) & support all (data) type of XML schema
but JAX-RPC uses Java type binding, supports 90% XML data type

- JAX-WS uses annotation (feature of Java 5) and DD is optional
JAX-RPC DD is must

- JAX-RPC and JAX-WS support SOAP 1.1. JAX-WS also supports SOAP 1.2.

Web Services Frameworks

4 major Web services toolkits/frameworks being used widely

- **.NET Web services**: Developed by Microsoft and is an integral part of the complete .NET framework. Integrated and easy to use with Visual Studio .NET. services are hosted on IIS web servers.
- **Spring Web services:** Developed by spring community and focusing on developing document driven webservices
- **Apache Axis(2):** Initially developed by IBM and donated to the Apache group. One of the earliest and stable Web service implementation. Runs on Apache Web servers.
- **Apache CXF:** Open source. Celtix (IONA Technologies) + XFire (Codehaus) = CXF.

REST (Representational State Transfer)

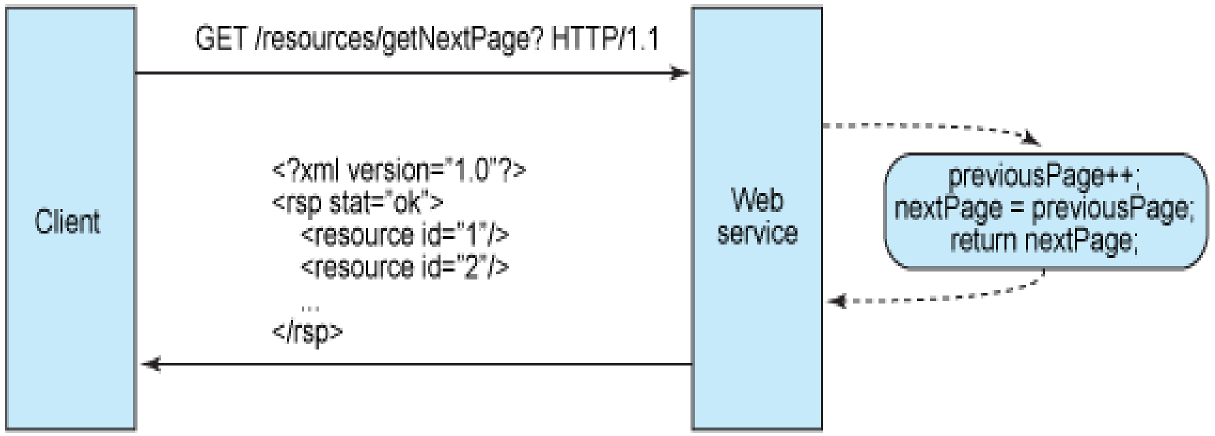
REST defines a set of architectural principles by which you can design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages

Web service follows four basic design principles:

- Use HTTP methods explicitly.
- Be stateless.
- Expose directory structure-like URIs.
- Transfer XML, JavaScript Object Notation (JSON), or both.

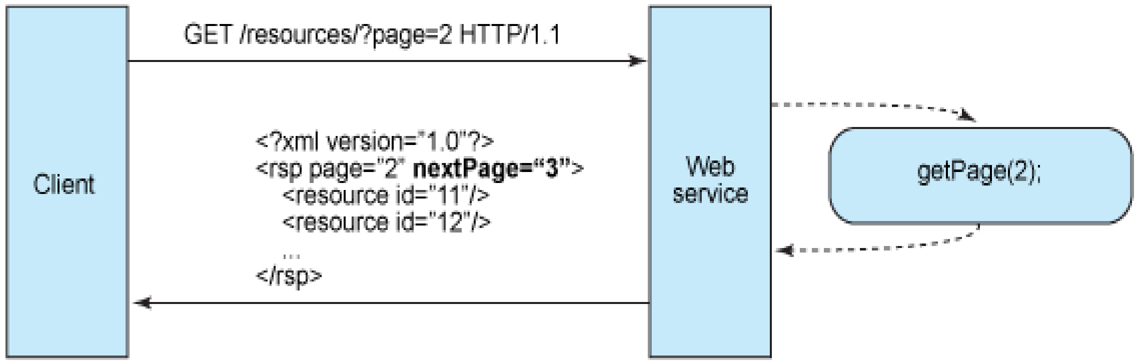
Stateful and Stateless Design

Figure 1. Stateful design



```
<?xml version="1.0"?>
<rsp stat="ok">
   <resource id="1"/>
   <resource id="2"/>
   ...
</rsp>
```

Figure 2. Stateless design



```
<?xml version="1.0"?>
<rsp page="2" nextPage="3">
   <resource id="11"/>
   <resource id="12"/>
   ...
</rsp>
```

RESTful Web services example

```
@Path("/hello")
public class Hello {
        // This method is called if TEXT_PLAIN is request
        @GET
        @Produces(MediaType.TEXT_PLAIN)
        public String sayPlainTextHello() {
                return "Hello Jersey";
        }
        @GET
        @Produces(MediaType.TEXT_XML)
        public String sayXMLHello() {// This method is called if XML
is request
                return "<?xml version=\"1.0\"?>" + "<hello> Hello
Jersey" + "</hello>";}}
```

Web services: challenges

- A whole suite of technologies to design:
  communication protocol
  description of services
  enable security and privacy
  describe complex interactions
  ...
- Must design to be interoperable and Web-friendly

Security

- WS do not define how to do security, they rely on other mechanisms layered on top.
- Very common to use SSL
  - Good for simple cases
    - Weak when multi-tier
    - Forces encryption of all data – sometimes not needed

References

- Java web services specification from Oracle https://www.oracle.com/technetwork/java/javaee/tech/webservices-139501.html
- Spring Web Services - https://spring.io/projects/spring-ws
- Java Web services In a Nutshell O’Reilly – Kim Topley
- XML In a Nutshell O'Reilly
- Java Web Services for Experienced Programmers Deitel developer series
- Java Web Services David Chappell O’Reilly