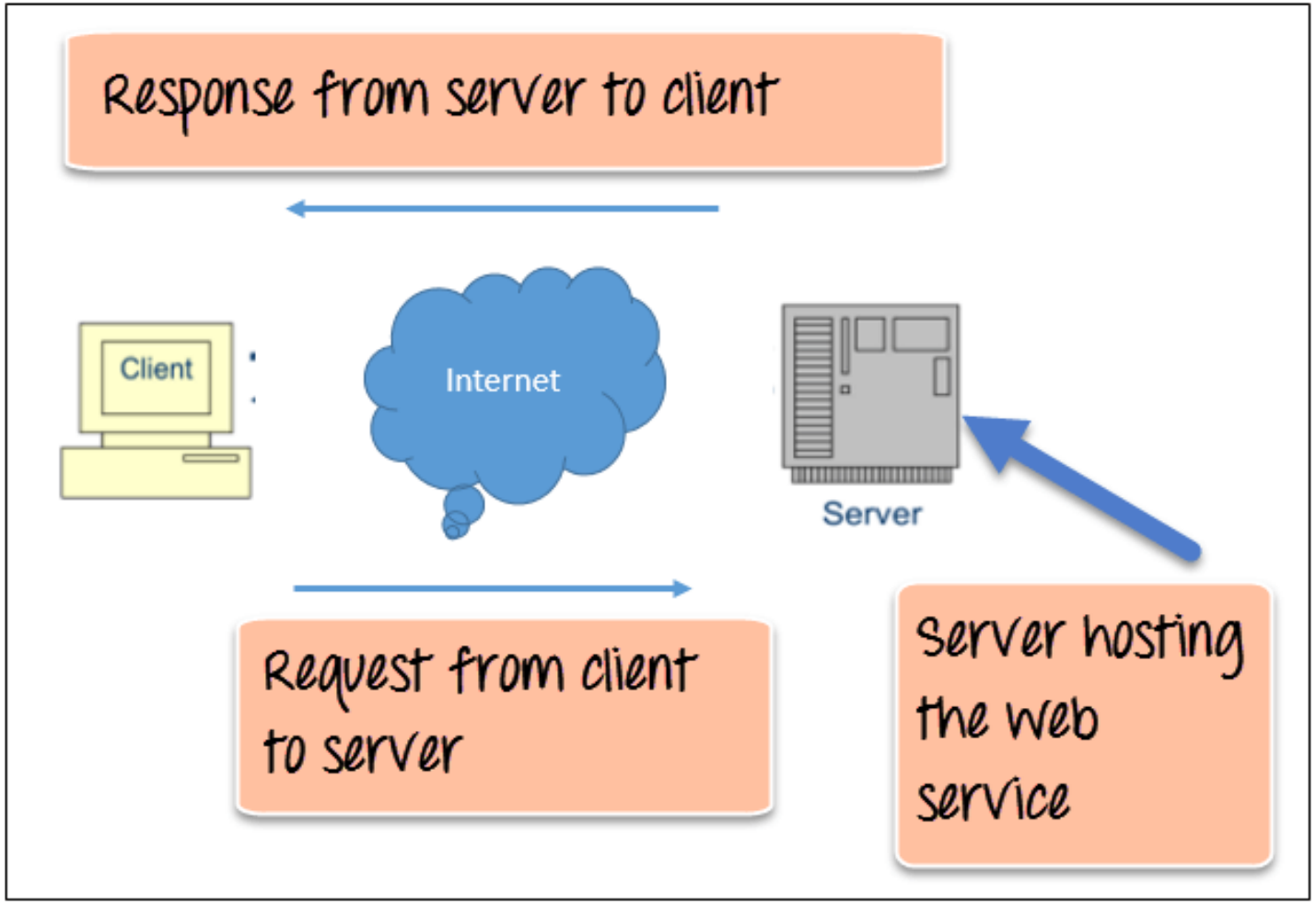
# Web Service

- Pengembangan Berbasis Platform -

# What is a Web Service?

- Web service is a standardized medium to propagate communication between the client and server applications on the WWW (World Wide Web).
  - Web server → as services provider
  - Client (ex: web browser) → as services requester
  - The client would invoke a series of web service calls via requests to a server which would host the actual web service.

# How WebServices Work?



# Why do we need a Web Service?

- Modern day business applications use variety of programming platforms to develop web-based applications.
  - Some applications may be developed in Java, others in .Net, while some other in Angular JS, Node.js, etc.
- Web services provide a common platform that allows multiple applications built on various programming languages to have the ability to communicate with each other.

# Web Services Advantages

- Exposing Business Functionality on the network
- Interoperability amongst applications
- A Standardized Protocol which everybody understands
- Reduction in cost of communication

# Type of Web Services

- SOAP (Simple Object Access Protocol) Web Services
- RESTful Web Services

# SOAP Web Services

- SOAP is known as a transport-independent messaging protocol.
- SOAP is based on transferring XML data as SOAP Messages.
- Each message is an XML document.
  - Only the structure of the XML document follows a specific pattern, but not the content.
- The best part of Web services and SOAP is that its all sent via HTTP, which is the standard web protocol.

# SOAP Building Block

- An Envelope element
  - It identifies the XML document as a SOAP message.
  - It is the root element of a SOAP message.
- A Header element
  - It contains header information or application-specific information (like authentication) about the SOAP message
- A Body element
  - It contains the actual SOAP message (request or response) intended for the ultimate endpoint of the message.
- A Fault element
  - This is the optional element that holds errors and status information for a SOAP message.

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

# Example of SOAP Request

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>

</soap:Envelope>
```

# Example of SOAP Response

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

# WSDL (Web Service Description Language)

- The client invoking the web service should know where the web service actually resides.
- The client application also needs to know what the web service actually does, so that it can invoke the right web service.
- This is done with the help of the **WSDL**.

# WSDL (Web Service Description Language)

- WSDL is used to describe the interface to available services at a particular website.
- WSDL files are XML documents that provide metadata for a SOAP service.
- They contain information about the functions or methods the application makes available and what arguments to use.
- By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

# Element of WSDL Document

| Element | Description |
| --- | --- |
| <types> | Defines the (XML Schema) data types used by the web service |
| <message> | Defines the data elements for each operation |
| <portType> | Describes the operations that can be performed and the messages involved. |
| <binding> | Defines the protocol and data format for each port type |

```
<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>

</definitions>
```

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

# RESTful Web Service

- REST stands for REpresentational State Transfer.
- The underlying protocol for REST is HTTP.
- REST is a way to access resources which lie in a particular environment.
- If a client, say a web browser needs any of resources in the server, it has to send a request to the server to access these resources.
- REST services defines a way on how these resources can be accessed.

# Key Elements of RESTful

- **Resources**
  - Let assume that a web application on a server has records of several employees.
  - Let's assume the URL of the web application is **http://xyz.com**.
  - Now in order to access an employee record resource via REST services, one can issue the command **http://xyz.com/employee/1.**
- **Request Verbs**
  - These describe what you want to do with the resource.
  - There are many other verbs available including things like POST, PUT, and DELETE.
  - Example: **http://xyz.com/employee/1** , the web browser is actually issuing a GET Verb because it wants to get the details of the employee record.
- **Request Headers**
  - These might define the type of response required or the authorization details.

# Key Elements of RESTful (cont.)

- **Request Body**
  - Data is normally sent in the request when a POST request is made to the REST web services.
  - In a POST call, the client actually tells the REST web services that it wants to add a resource to the server.
  - Hence, the request body would have the details of the resource which is required to be added to the server.
- **Response Body**
  - This is the main body of the response.
  - Example request **http://xyz.com/employee/1** , the web server might return an XML document with all the details of the employee in the Response Body.
- **Response Status codes**
  - These codes are the general codes which are returned along with the response from the web server.
  - An example is the code 200 which is normally returned if there is no error when returning a response to the client.

# RESTful Methods

- RESTful basically works on the HTTP web layer and uses the below key verbs to work with resources on the server
  - POST – To create a resource on the server
  - GET – To retrieve a resource from the server
  - PUT – To change the state of a resource or to update it
  - DELETE – To remove or delete a resource from the server

# RESTful Architecture

- State and functionality are divided into distributed resources
  - This means that every resource should be accessible via the normal HTTP commands of GET, POST, PUT, or DELETE.
- The architecture is client/server, stateless, layered, and supports caching
  - Client-server is the typical architecture where the server can be the web server hosting the application, and the client can be as simple as the web browser.
  - Stateless means that the state of the application is not maintained in REST.
    - For example, if you delete a resource from a server using the DELETE command, you cannot expect that delete information to be passed to the next request.

# Referensi


- https://www.guru99.com/web-service-architecture.html
- https://www.guru99.com/restful-web-services.html
- https://www.w3schools.com/xml/xml_wsdl.asp
- https://www.w3schools.com/xml/xml_soap.asp