Web Services

What is Web Service

Web Service- W3C Definition

Web Service is "a software system designed to support interoperable machine-to-machine interaction over a network."

- ***What is Web:*** An enormous interconnected network of resources, where a *resource* is a Uniform Resource Identifier (URI)-named data source such as a PDF-based document, a video stream, a Web page, or even an application

  - These resources can be accessed by using standard Internet protocols such as Hyper Text Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP).

What is Web Service

- ***Service:*** A server-based application or software component that exposes a resource to clients via an exchange of messages

- **Interoperability** means for example, a Java based application on Windows can communicate with a .NET based application on Linux.

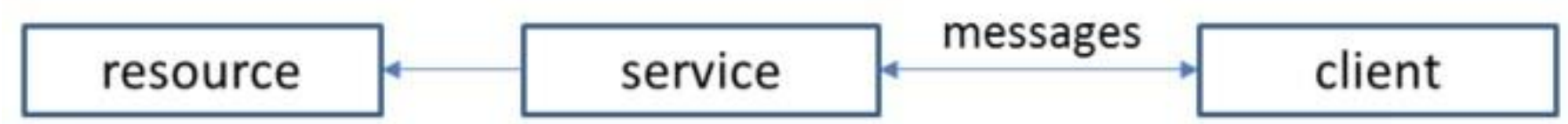
    - The communication can be done through a set of messages in XML or JSON format over HTTP protocol.

What is Web Service

3 Key points

- **Designed for machine-to-machine(or application-to-application) interaction**
- **Should be interoperable- Not platform dependent**
- **Should allow communication over a network**

A client accesses a resource by exchanging messages with a Web service

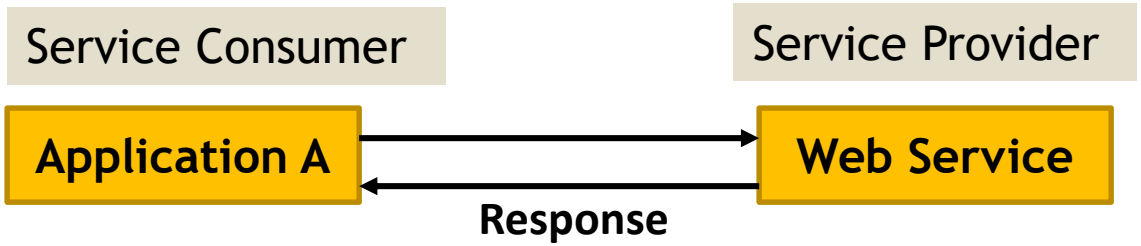


Key Terminologies

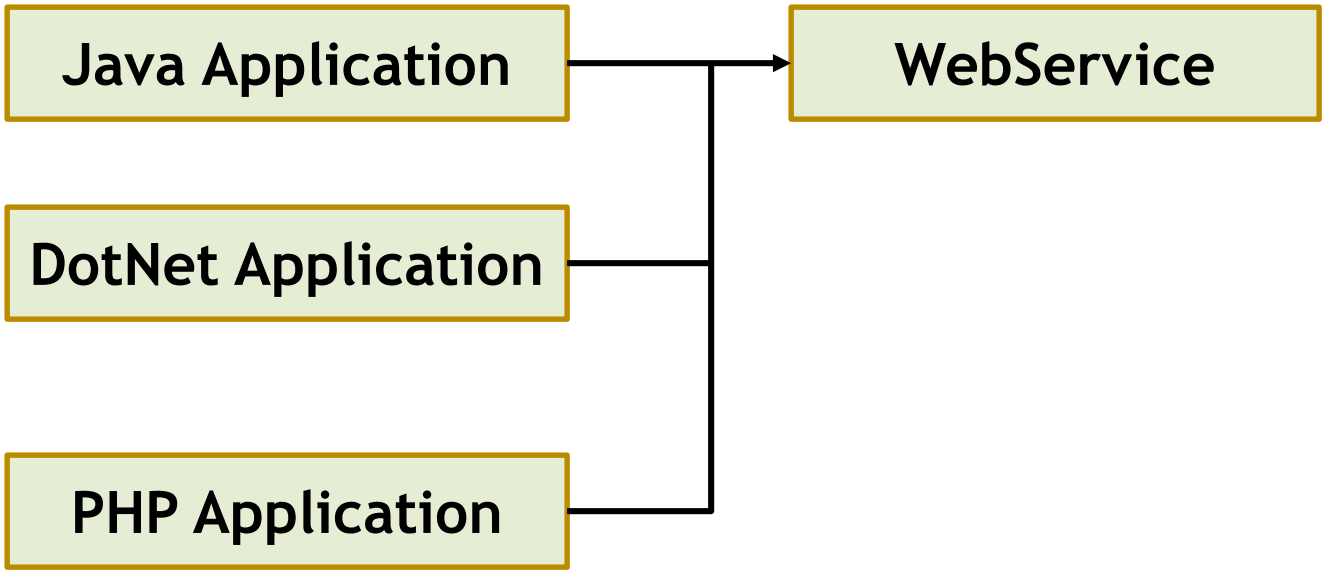
- Request and Response
- Message Exchange Format
  - XML and JSON
- Service Provider or Server
- Service Consumer or Client
- Service Definition
- Transport
  - HTTP and MQ

Key Terminologies

- Request and Response
  - Request is an input to the web service and Response is the output from a web service.
- Message Exchange Format
  - XML and JSON
    - Message exchange format signifies the format of the Request and Response , like is it XML or JSON.
- Service Provider and Service Consumer



Service Provider and Service Consumer



Java Application, DotNet Application and PHP Application are Service consumers while WebService is the Service Provider

RESTful Web Service

RESTful web services

A *RESTful Web Service* is a widely used Web service category that's based on *Representational State Transfer (REST)*, a software architecture style for distributed *hypermedia systems(www)* (systems in which images, text, and other resources are located around networks and are accessible via hyperlinks).

RESTful web services

The central part of REST is the URI-identifiable resource. REST identifies resources by their Multipurpose Internet Mail Extensions (MIME) types (such as text/xml).

When a client requests a resource from a RESTful Web service, the service sends a MIME-typed representation of the resource to the client.

Also, resources have states that are captured by their representations.

Clients use HTTP's POST, GET, PUT, and DELETE verbs to retrieve resource representations and to manipulate resources.

RESTful web services

REST maps these verbs onto the database Create, Read, Update, and Delete (CRUD) operations, as follows:

- POST: Create new resource based on request data.
- GET: Read existing resource without producing side effects (don't modify the resource).
- PUT: Update existing resource with request data.
- DELETE: Delete existing resource.

Each verb is followed by a URI that identifies the resource.

The URI might refer to a collection, such as

http://javajeff.ca/library,

Or to an element of the collection, such as

http://javajeff.ca/library/9781484219157

Building RESTful Web Services

The following principles encourage RESTful applications to be simple, lightweight, and fast:

- **Resource identification through URI**: A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients.

Resources are identified by URIs, which provide a global addressing space for resource and service discovery.

- **Uniform interface**: Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.

Building RESTful Web Services

- **Self-descriptive messages**: Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others.

- **Stateful interactions through hyperlinks**: Every interaction with a resource is stateless; that is, request messages are self-contained.

Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields.

State can be embedded in response messages to point to valid future states of the interaction.

Swagger

Swagger is an open source software framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful Web services.

Swagger is an Interface Description Language for describing RESTful APIs expressed using JSON.

Swagger is used together with a set of open-source software tools to design, build, document, and use RESTful web services.

Swagger includes automated documentation,

Building RESTful Web Services with JAX-RS

JAX-RS is a Java programming language API designed to make it easy to develop applications that use the REST architecture.

The JAX-RS API uses Java programming language annotations to simplify the development of RESTful web services.

Jersey, the reference implementation of JAX-RS, implements support for the annotations defined in JSR 311, making it easy for developers to build RESTful web services by using the Java programming language.

JAX-RS Annotations

Annotation	Description
@Path	The @Path annotation's value is a relative URI path indicating where the Java class will be hosted: for example, /helloworld. You can also embed variables in the URIs to make a URI path template. For example, you could ask for the name of a user and pass it to the application as a variable in the URI: /helloworld/{username}.
@GET	The @GET annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP GET requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.
@POST	The @POST annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP POST requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.
@PUT	The @PUT annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP PUT requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.

JAX-RS Annotations

Annotation	Description
@DELETE	The @DELETE annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP DELETE requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.
@HEAD	The @HEAD annotation is a request method designator and corresponds to the similarly named HTTP method. The Java method annotated with this request method designator will process HTTP HEAD requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.
@PathParam	The @PathParam annotation is a type of parameter that you can extract for use in your resource class. URI path parameters are extracted from the request URI, and the parameter names correspond to the URI path template variable names specified in the @Path class-level annotation.
@QueryParam	The @QueryParam annotation is a type of parameter that you can extract for use in your resource class. Query parameters are extracted from the request URI query parameters.
@Consumes	The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client.

JAX-RS Annotations

@Produces	The @Produces annotation is used to specify the MIME media types of representations a resource can produce and send back to the client: for example, "text/plain".
@Provider	The @Provider annotation is used for anything that is of interest to the JAX-RS runtime, such as **MessageBodyReader** and **MessageBodyWriter** **Providers are a simply a way of extending and customizing the JAX-RS runtime.** **JAX-RS** runtime comes with a number of predefined providers that will be responsible for implementing a base level of functionality (e.g for mapping to and from XML, translating the most common exceptions etc etc). You can also create your own providers as needed.

Overview of a JAX-RS Application

```
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;

// The Java class will be hosted at the URI path "/helloworld"

@Path("/helloworld")
public class HelloWorldResource {
// The Java method will process HTTP GET requests
@GET
// The Java method will produce content identified by the MIME Media  type "text/plain"
@Produces("text/plain")
public String getClichedMessage() {
   // Return some textual content
   return "Hello World";
}
}
```

Thank You!