International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 4 | ISSN : 2456-3307

RESTful -Webservices

Lalit Kumar[1], Dr. R. Chinnaiyan[2]
[1]Student of MCA, New Horizon College of Engineering, Bangalore, Karnataka, India
[2]Professor, Department Of Master of Computer Applications, New Horizon College of Engineering, Karnataka, India

ABSTRACT

The RESTful Web Service is a type of Web Service that help to share the resource among more than two software application. The Web Service help to data integration among two or more application of same or different technology. Web Service are mainly two type JAX-WS (SOAP) and JAX-RS (RESTful). The RESTful Web Service implement the JAX-RS API. Web Services are client or server-side application that share information over the HTTP (Hyper Text Transfer Protocol). As described by the W3C. We need to design a web resource and that resource identify the URI of that resource. The RESTful Webservice is also State less protocol like HTTP. It's support XML, JSON, TEXT etc. media type to represent the resource.

Keywords: RESTful, Jersey, XML, String, JSON, Java, Python, Angular- JS, Microsoft .NET, Resource, Service, Web Service. HTML Methods, GET, POST, PUT, DELETE, OPTION, MySQL, Client, Postman, Annotation, Maven.

I. INTRODUCTION

WEB SERVICE is Client or Server-side application that share the information among different kind of application it might be developed using different kind of technology platform or same technology. It will share information remotely using the URI of the Web Service. Web Service support the XML and JSON to send the information to other application. The Web Service mainly two type JAX-WS (SOAP or Big Web Service) and JAX-RS (RESTful) API.

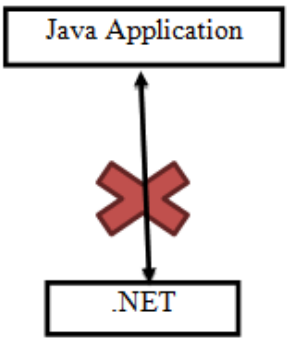

Figure 1. The Applicatios can not share data without Web Services.

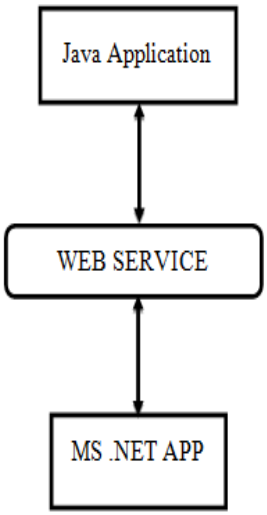

Figure 2. The Proccess of sharing data by using web services.

The RESTful Webservice stand for Representational State Transfer. JAX-RS: makes it easier to write web applications that apply some or all of the constraints of the REST style to induce desirable properties in the application, such as loose coupling (evolving the

server is easier without breaking existing clients), scalability (start small and grow), and architectural simplicity (use off-the-shelf components, such as proxies or HTTP routers). You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services

Whichever Information we want to share among other application then we need to define resource. The Resource will contain the information that need to transfer to other client application of the service. The RESTful support the repartition of the Web Service help of XML and JSON. The RESTful main plus point is that it supports JSON. The JSON is JavaScript Object Notation. It's easy to understand then XML.

The RESTful can be define any technology like JAVA, Python, JavaScript, Microsoft .NET, AngularJS, NodeJS, Android, Objective C, Swift etc.
These are the principle of RESTful Web Service.

1. Resource identification through URI
2. Uniform interface
3. Self-descriptive messages
4. Stateful interactions through hyperlinks

It's stateless Web Service like HTTP. It's support all HTPP method like POST, GET, PUT etc. The description are given below.

A. GET.

The Client can get information only in read only mode. The Client cannot create or update the resource data. It's like HTTP GET method.

B. POST.

The Client can edit delete or update the Resource information. It's similar to HTTP POST method.

C. PUT.

The Client can create new resource, but it cannot delete any existing resource data. It's like HTTP PUT method.

D. DELETE.

The client can delete the existing resource information. It's similar like HTTP Delete method.

E. HEAD.

The client can access the Header information like HTTP Head Method.

F. OPTION.

This is optional. Client can select any one above method in the OPTION method.

We need to use the JAX-RS library. We need to import he javax.ws.rs. *; package in java to implement and create resource. We can develop a RESTful Web Service Application by using jersey.

II. RESOURCE

Resource is an architecture of contains. The resource can be a text files, html pages, Images, Videos or any dynamic business data. REST Server simply provides access to resources and REST client accesses and modifies the resources. Here each resource is identified by URIs/ Global IDs. REST uses various representations to represent a resource where Text, JSON, XML. The most popular representations of resources are XML and JSON.

III. RESOURCE REPRESENTATION

Resource can be represent using either xml or JSON. A resource in REST is a similar Object in Object Oriented Programming or is like an Entity in a Database. Once a resource is identified then its representation is to be decided using a standard format so that the server can send the resource in the above said format and client can understand the same format.

Examples:

(XML)

```
<user>
        <id> 01 </id>
        <name> Lalit Kumar </name>
```

```
        <addr> Bangalore </addr>
</user>
(JSON)
{
        “id”:01,
        “name” : ”Lalit Kumar”
        “addr” : ”Bangalore”
}
```

IV. BASIC ANNOTATION

This are the basic annotation that help us to define a resource.

A. 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}.

```
  @Path(“/users”)
  public class UserResource{
        intuserId;
  }
```

B. 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.

```
  @Path(“/users”)
  public class UserResource{
        intuserId;
        @GET
        public String getId(){
                return this.userId;
        }
  }
```

C. 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 behaviour of a resource is determined by the HTTP method to which the resource is responding.

```
  @Path(“/users”)
  public class UserResource{
        intuserId;
        @POST
        public String getId(){
                return this.userId;
        }
}
```

D. 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 behaviour of a resource is determined by the HTTP method to which the resource is responding.

```
  @Path(“/users”)
  public class UserResource{
        intuserId;
        @PUT
        public String setId(){
                this.userId=1;
        }
}
```

E. 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.

F. 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 behaviour of a resource is determined by the HTTP method to which the resource is responding.

G. 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.

H. 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.

I. CONSUMES.

The @Consumes annotation is used to specify the MIME media types of representations a resource can consume that were sent by the client.

J. 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".

K. CONTEXT.

This is helping to access the context servlet object like response, request, cache, session etc.

V. RESTFUL IMPALEMENT

We can create a RESTful application in java in two ways. First, we need to create a Web Application and we need to include and configure the all Jersey jar library to Java Build Path and servlet mapping to RESTful service servlet in WEB.XML. Second, we can create a MAVEN application as maven-webapp and We need to add the RESTful dependency to POM.XML. Then we need to map the REST Servlet in WEB.XML.

A sample of a resource is following

```
package com.lalit.rest;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Path("/hello")
public class Hello {
@GET
@Path(value="/sayHi")
@Produces(MediaType.TEXT_HTML)
private String sayPlainTextHello() {
                    return "<html><body>Hello
jersey 1.0</body></html> ";
        }
}
```

We can test this resource by the REST Client like POSTMAN. The POSTMAN is freely available application that help to test a web service. The about resource URI might me like this

http://localhost:8080/myfirstrest/rest/hello/sayHi

It will produce the result based on GET method like this

Hello jersey 1.0

VI. CONCLUSION

RESTful web service help to create a web service. It supports the JSON to represent the Resource and transfer the information. It is easy to handle the web service. It's support by almost all platform. It helps to share the data among applications. The RESTful Web service is using widely in IT. The Web service are very helpful to share data among Application based on different technology and located at different locations.

VII. ACKNOWLEDGMENT

We would like to express our special thanks of gratitude to our teachers as well as our principal, HOD and guides who gave us the golden opportunity to do this wonderful project on the topic (RESTful Web Service), which also helped us in doing a lot of Research and we came to know about so many new things We are thankful to them.

VIII. REFERENCES

[1]. https://javapapers.com/java/java-restful-web-services-with-json-and-jersey/
[2]. http://stackoverflow.com
[3]. https://openliberty.io/guides/rest-client-java.html
[4]. RESTful Webservices, Tutorials point
[5]. RESTful Web Videos Tutorials, Skillsoft.