



Web Services

REST

Se expone como un RECURSO de una aplicación Web

Trabaja directamente sobre el protocolo HTTP

Para consumir un REST, se requiere el URI y la acción HTTP(GET, POST o PUT)

Para que dentro de la aplicación web se de un tratamiento especial al REST, se configura un servlet en el archivo web.xml

Ejemplo de configuración en una aplicación a desplegarse en JBOSS:

```
<servlet-mapping>  
  <servlet-name>javax.ws.rs.core.Application</servlet-name>  
  <url-pattern>/rest/*</url-pattern>  
</servlet-mapping>
```

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

Ruta para acceder a la clase.

```
@Path(value = "/bienvenida")  
public class Bienvenida {
```

Acción HTTP a la que responde el método

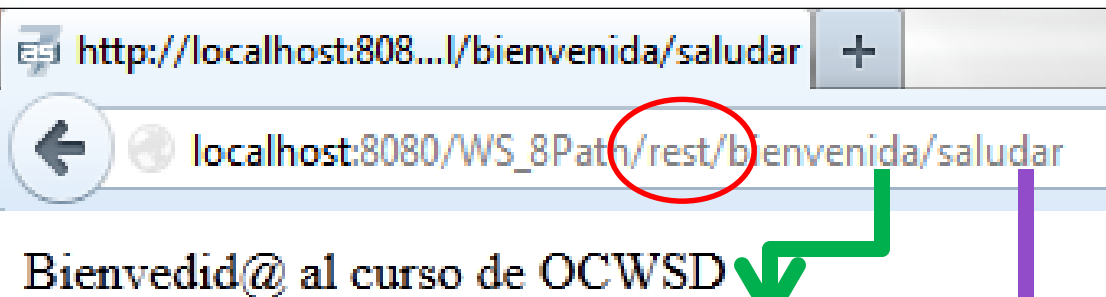
```
@GET
```

Ruta para acceder al método

```
@Path(value = "/saludar")  
public String saludar() {  
    return "Bienvedid@ al curso de OCWSD";  
}  
}
```

Web.xml

```
<servlet-mapping>
  <servlet-name>javax.ws.rs.core.Application</servlet-name>
  <url-pattern>/rest/*</url-pattern>
</servlet-mapping>
```




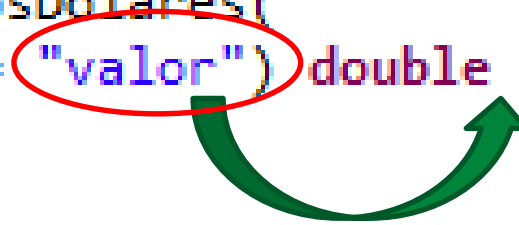
Cuando se envía un Request desde un navegador, se dispara un request con una acción GET.

```
@Path(value = "/bienvenida")
public class Bienvenida {

    @GET
    @Path(value = "/saludar")
    public String saludar() {
        return "Bienvedid@ al curso de OCWSD";
    }
}
```

Bienvenida.java


```
@Path(value = "/convertidor")
public class Convertidor {
    @GET
    @Path(value = "/convertirEaD")
    public double convertirEurosDolares(
        @QueryParam(value = "valor") double euros) {
        return euros * 1.3378;
    }
}
```

 HttpReqtester

Request

URL:

GET

Submit

GET

POST

PUT

New request

Paste Request

Content to Send

Headers

Parameters

El método **recibe** una Persona en trama **json**

```
@POST
@Path("/insertar")
@Consumes("application/json")
public void insertar(Persona persona){
```

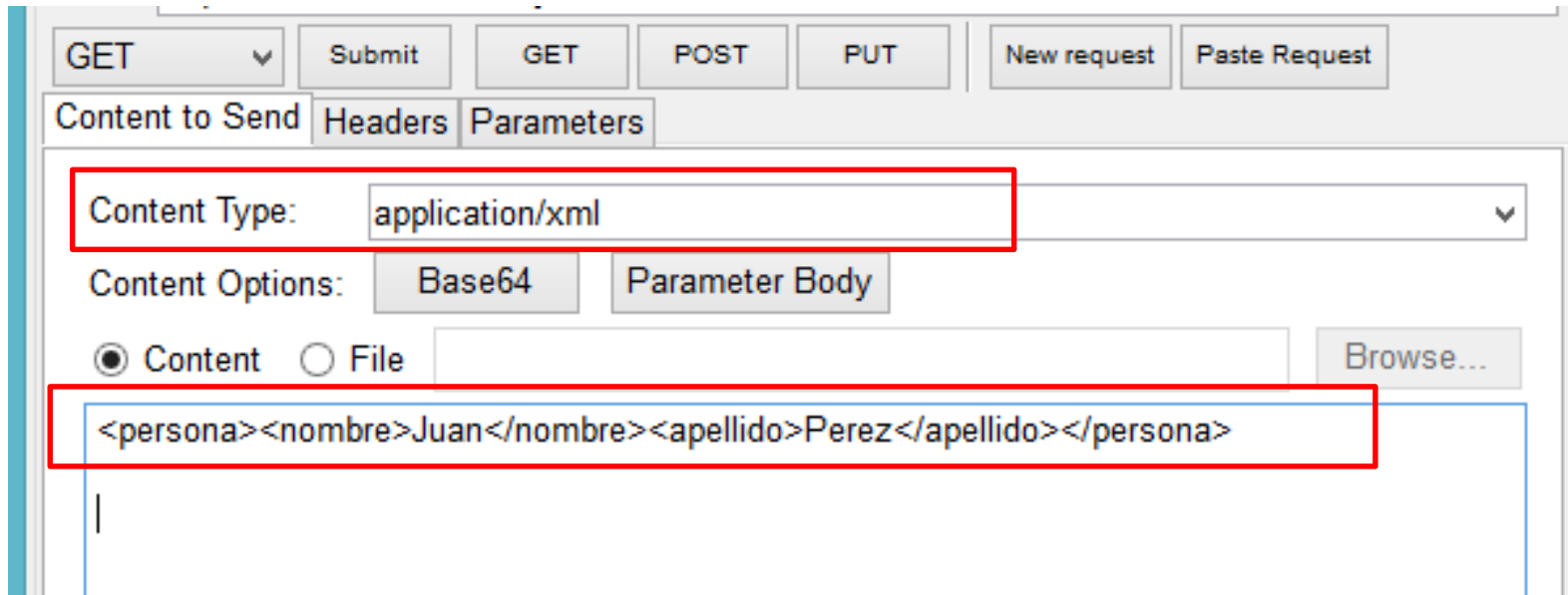
El método **retorna** una Persona en trama **json**

```
@PUT  
@Path("/buscar")  
@Produces("application/xml")  
public Persona buscar(){
```


El método **recibe** una Persona en trama **xml**
retorna una Persona en trama **json**

```
@POST  
@Path("/modificar")  
@Produces("application/json")  
@Consumes("application/xml")  
public Persona modificar(Persona persona){
```

```
@POST
@Path("/modificar")
@Produces("application/json")
@Consumes("application/xml")
public Persona modificar(Persona persona){
```



The screenshot shows the HTTP Requester application interface. At the top, there are buttons for GET, POST, PUT, and others. Below these, there are tabs for Content to Send, Headers, and Parameters. The Content to Send tab is active. In this tab, the Content Type is set to application/xml. Below this, there are Content Options: Base64 and Parameter Body. The Content option is selected. At the bottom, there is a text area for the request body, which contains the XML string: <persona><nombre>Juan</nombre><apellido>Perez</apellido></persona>. The text area is highlighted with a red box.

GET Submit GET POST PUT New request Paste Request

Content to Send Headers Parameters

Content Type: application/xml

Content Options: Base64 Parameter Body

☒ Content ☐ File Browse...

<persona><nombre>Juan</nombre><apellido>Perez</apellido></persona>

```
@POST
@Path("/modificar")
@Produces("application/json")
@Consumes("application/xml")
public Persona modificar(Persona persona){
```

GET



Submit

GET

POST

PUT

New request

Paste Request

Content to Send

Headers

Parameters

Name: Accept



Value: application/json

Add/Change

Name

Value

Accept

application/json

CLIENTE

SERVIDOR

HttpHeader: Accept

Indica que tipo de contenido
ACEPTA. ←

HttpHeader: Content-Type

Indica que tipo de contenido
ENVIA → en el Request.

@Produces

Indica que tipo de contenido
← **ENVIA** en el response.

@Consumes

Indica que tipo de contenido
ACEPTA →

El tipo de contenido que **envía** debe ser
igual al tipo de contenido que **acepta**.

Enviar Datos

```
URL url = new URL(  
    "http://localhost:8080/WS_ProducesConsumes/ws/s3/m1");  
URLConnection connection = (URLConnection) url  
    .openConnection();
```

Indica la url donde se encuentra el método y genera la conexión.

```
connection.setDoOutput(true);  
connection.setRequestMethod("POST");
```

Indica que se va a enviar un contenido en el request

Indica la acción en la cual se va a invocar al método

```
connection.setRequestProperty("Content-Type", "application/xml");
```

Indica el **tipo de contenido** que se envía en el request.

Enviar Datos

```
OutputStreamWriter writer = new OutputStreamWriter(  
    connection.getOutputStream());
```

Se crea un OutputStreamWriter recuperando el OutputStream de la conexión

```
writer.write("<cuenta><codigo>10</codigo><saldo>250</saldo></cuenta>");  
writer.close();
```

El método write recibe como parámetro un String con el contenido que se va a enviar en el request.

```
System.out.println(connection.getResponseCode());  
System.out.println(connection.getResponseMessage());
```


Recibir Datos

```
URL url = new URL(  
    "http://localhost:8080/WS_ProducesConsumes/ws/s1/m4");
```

Se indica la url donde se encuentra el servicio.

```
URLConnection connection = (URLConnection) url  
    .openConnection();
```

```
connection.setDoInput(true);
```

Indica que va a recibir datos en el response.

```
connection.setRequestMethod("GET");
```

El parámetro indica la acción mediante la cual se ejecuta el método.

```
connection.setRequestProperty("Accept", "application/xml");
```

Indica el tipo de contenido que se acepta en el response.

Recibir Datos

```
InputStreamReader reader = new InputStreamReader(  
    connection.getInputStream());
```

Se crea un InputStreamReader recuperando el InputStream de la conexión.

```
BufferedReader br=new BufferedReader(reader);  
String tmp=null;  
String respuesta="";  
while((tmp=br.readLine())!=null){  
    respuesta += tmp;  
}  
  
System.out.println("Respuesta: "+respuesta);  
System.out.println(connection.getResponseCode());  
System.out.println(connection.getResponseMessage());
```