El API de JAX-RS Client va a terminar por ser una de las más utilizadas , la llegada de las nuevas arquitecturas de Microservicios supondrá una explosión a la hora de crear servicios REST. Vamos a introducirla brevemente y ver como funciona. En este caso usaremos una aplicación de consola que invoque un servicio REST. Para ello construiremos un proyecto de Maven con varias dependencias.

```
&amp;amp;lt;dependencies&amp;amp;gt;
 &amp;amp;lt;dependency&amp;amp;gt;
&amp;amp;lt;groupId&amp;amp;gt;org.glassfish.jersey.media&amp;amp;lt;/
groupId&amp;amp;gt;
 &amp;amp;lt;artifactId&amp;amp;gt;jersey-media-json-
jackson&amp;amp;lt;/artifactId&amp;amp;gt;
 &amp;amp;lt;version&amp;amp;gt;2.23.1&amp;amp;lt;/version&amp;amp;gt;
 &amp;amp;lt;/dependency&amp;amp;gt;

 &amp;amp;lt;!--
https://mvnrepository.com/artifact/org.glassfish.jersey.core/jersey-cl
ient --&amp;amp;gt;
 &amp;amp;lt;dependency&amp;amp;gt;
&amp;amp;lt;groupId&amp;amp;gt;org.glassfish.jersey.core&amp;amp;lt;/g
roupId&amp;amp;gt;
 &amp;amp;lt;artifactId&amp;amp;gt;jersey-
client&amp;amp;lt;/artifactId&amp;amp;gt;
 &amp;amp;lt;version&amp;amp;gt;2.23.1&amp;amp;lt;/version&amp;amp;gt;
 &amp;amp;lt;/dependency&amp;amp;gt;

 &amp;amp;lt;/dependencies&amp;amp;gt;
```

Una vez instaladas las dependencias el uso del API es muy sencillo , en nuestro caso vamos

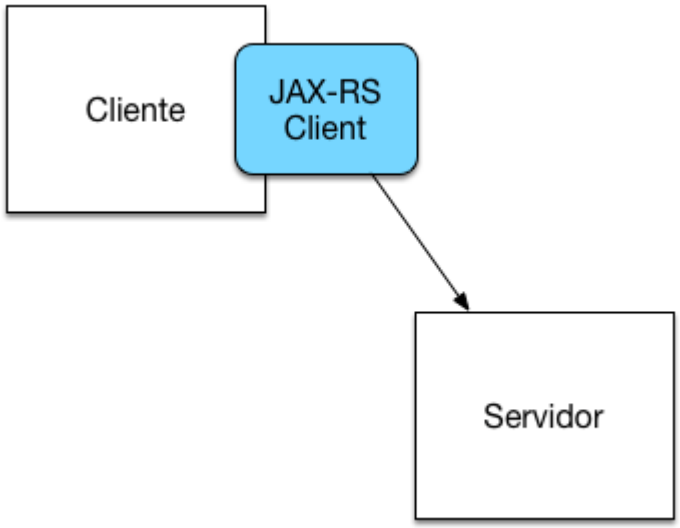
a invocar un servicio REST que nos devuelve una lista de Personas. El primer paso es crear la clase Persona.

```
package com.arquitecturajava;

import java.io.Serializable;

public class Persona implements Serializable {
private static final long serialVersionUID = 628584501486547101 5L;
private String nombre;
private String apellidos;
private int edad;

public String getNombre() {
return nombre;
}
public void setNombre(String nombre) {
this.nombre = nombre;
}
public String getApellidos() {
return apellidos;
}
public void setApellidos(String apellidos) {
this.apellidos = apellidos;
}
public int getEdad() {
return edad;
}
public void setEdad(int edad) {
```

```
this.edad = edad;
}
}
```

JAX-RS Client API

Una vez disponemos de nuestro objeto de negocio usamos JAX-RS Client API e invocamos al servicio desde una aplicación de consola.

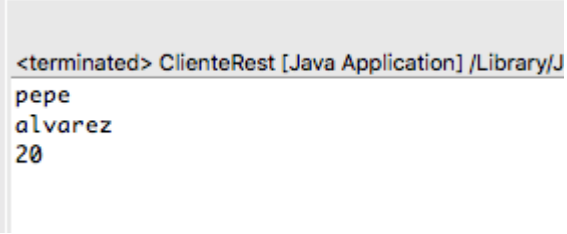


Vamos a ver el código:

```
Client client = ClientBuilder.newClient().register(new
JacksonFeature());
Persona persona = client
.target("http://localhost:3000/persona/pepe")
.request(MediaType.APPLICATION_JSON).get(Persona.class);
```

```
System.out.println(persona.getNombre());
System.out.println(persona.getApellidos());
System.out.println(persona.getEdad());
```

La aplicación imprimirá los datos por pantalla:

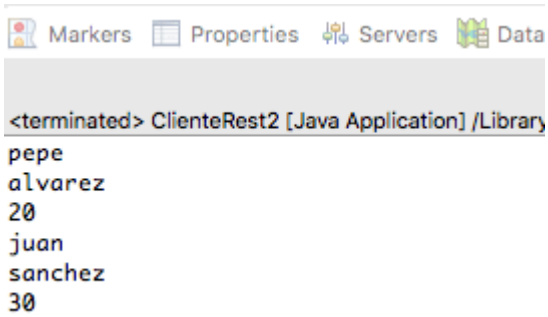


Solicitar que nos devuelta la lista de personas es similar pero hace falta apoyarse en genéricos.

```
package com.arquitecturajava;
import java.util.List;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.core.GenericType;
import javax.ws.rs.core.MediaType;

import org.glassfish.jersey.jackson.JacksonFeature;

public class ClienteRest2 {

public static void main(String[] args) {

Client client = ClientBuilder.newClient().register(new
```

```
JacksonFeature());
List&amp;amp;lt;Persona&amp;amp;gt; personas = client
.target("http://localhost:3000/persona")
.request(MediaType.APPLICATION_JSON).get(new
GenericType&amp;amp;lt;List&amp;amp;lt;Persona&amp;amp;gt;&amp;amp;gt;
() {
});

for(Persona p: personas) {

System.out.println(p.getNombre());
System.out.println(p.getApellidos());
System.out.println(p.getEdad());

}

}

}
```

El resultado nos mostrará una lista de elementos.



Hemos realizado dos peticiones a un servicio REST que devuelve los datos en formato JSON usando JAX-RS Client API

CURSO Java Herencia

GRATIS

APUNTATE!!

Otros artículos relacionados :

Spring Rest Controllers

Servicios REST