

Pack de Cursos Gratis para Juniors



Accede ahora

arquitecturajava

Pack de Cursos Gratis para Seniors



Accede ahora

arquitecturajava

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.

```

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

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

&lt;/dependencies&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;

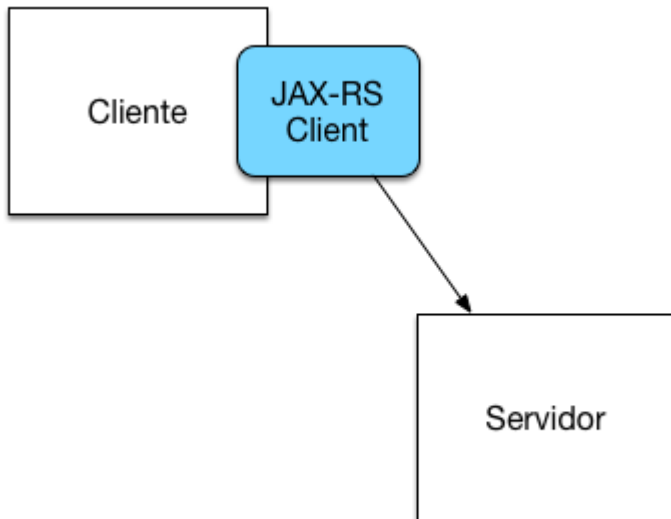
```

```
public class Persona implements Serializable {
private static final long serialVersionUID = 6285845014865471015L;
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:

```
<terminated> ClienteRest [Java Application] /Library/J
pepe
alvarez
20
```

Solicitar que nos devuelva 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;

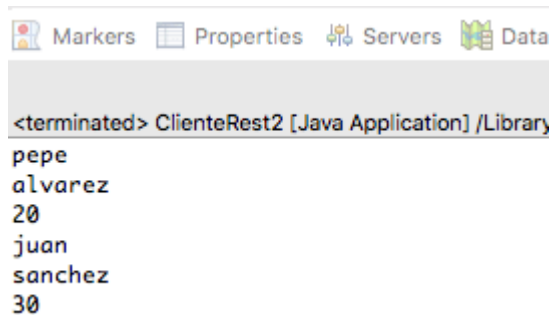
public class ClienteRest2 {

    public static void main(String[] args) {

        Client client = ClientBuilder.newClient().register(new
        JacksonFeature());
        List<Persona> personas = client
        .target("http://localhost:3000/persona")
        .request(MediaType.APPLICATION_JSON).get(new
        GenericType<List<Persona>>());
    }
};
```

```
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.



```
<terminated> ClienteRest2 [Java Application] /Library  
pepe  
alvarez  
20  
juan  
sanchez  
30
```

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](#)