PostMan App con Spring REST

El uso de PostMan App es muy habitual en el trabajo con Servicios REST. La mayor parte de nuestras aplicaciones web publican información en forma de recurso para que otras aplicaciones puedan acceder a ella.

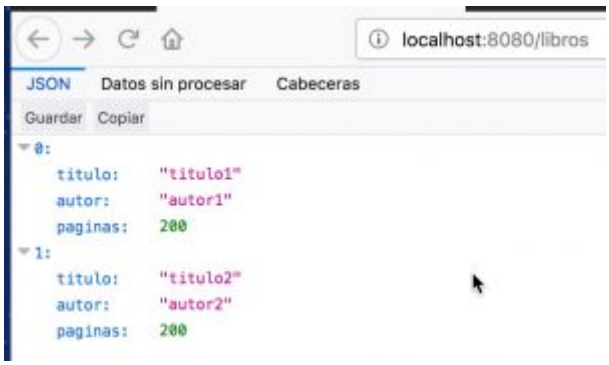
Eso nos obliga a estar continuamente probando el funcionamiento de estos servicios y gestionar las llamadas GET, POST, PUT, DELETE etc.

Postman App es una aplicación multiplataforma que nos permite probar nuestros servicios REST de una forma sencilla. Vamos a crear un proyecto con Spring Boot y Spring MVC para ver las ventajas que tiene. El primer paso es definir un @RestController que nos devuelva una colección de Libros.

```
package com.arquitecturajava.web.springweb;
import java.util.ArrayList;
import java.util.List;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class ControladorLibro {
	static List<Libro> lista=new ArrayList<Libro>();
	static {
		lista.add(new Libro("titulo1","autor1",200));
		lista.add(new Libro("titulo2","autor2",200));
	}
	@GetMapping("/libros")
	  public List<Libro> getLibros() {
		return lista;
	}
	@PostMapping(path="/libros",consumes="application/json")
	public ResponseEntity<String>  insertarLibro( @RequestBody Libro l) {
		lista.add(l);
		return new ResponseEntity(HttpStatus.OK);
	}
}
```

```
package com.arquitecturajava.web.springweb;
public class Libro {
	private String titulo;
	private String autor;
	private int paginas;
	public Libro(String titulo, String autor, int paginas) {
		super();
		this.titulo = titulo;
		this.autor = autor;
		this.paginas = paginas;
	}
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public String getAutor() {
		return autor;
	}
	public void setAutor(String autor) {
		this.autor = autor;
	}
	public int getPaginas() {
		return paginas;
	}
	public void setPaginas(int paginas) {
		this.paginas = paginas;
	}
	public Libro() {
super();
	}
}
```

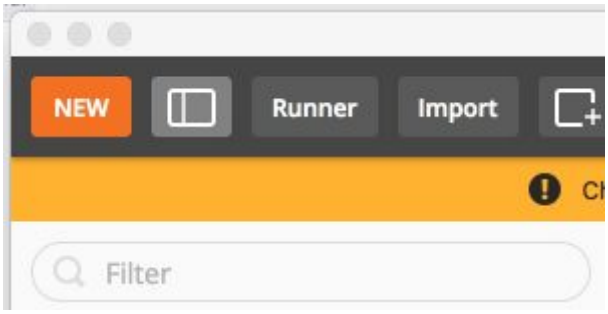
Si ejecutamos la aplicación podemos invocar el Servicio REST y acceder a la lista en formato JSON.



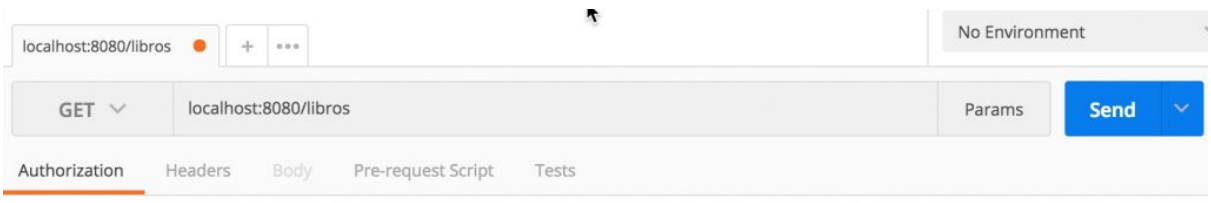
Utilizando PostMan App

En este caso ha sido muy sencillo ejecutar el proyecto y acceder a la lista de Libros. Lamentablemente existen muchas situaciones en las que las operaciones pueden ser más complejas. Vamos a ver como podemos realizar la misma operación utilizando **PostMan**. Para ello el primer paso es descargar la aplicación instalarla y registrarnos

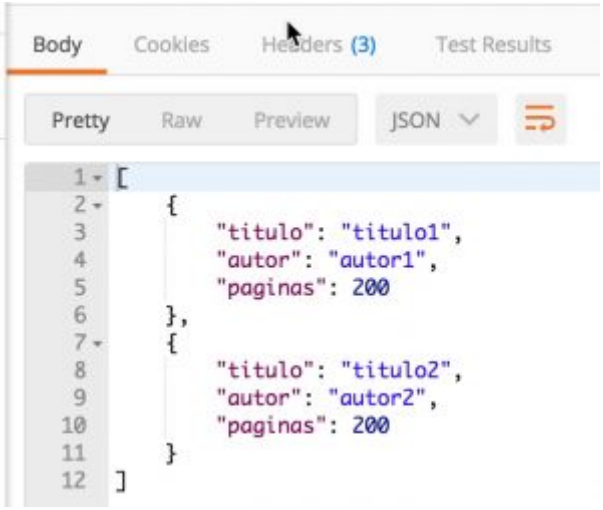
Una vez registrada la aplicación podremos acceder de forma sencilla a sus capacidades a la hora de realizar peticiones HTTP. Para ello pulsamos en el botón de new y creamos una nueva petición.



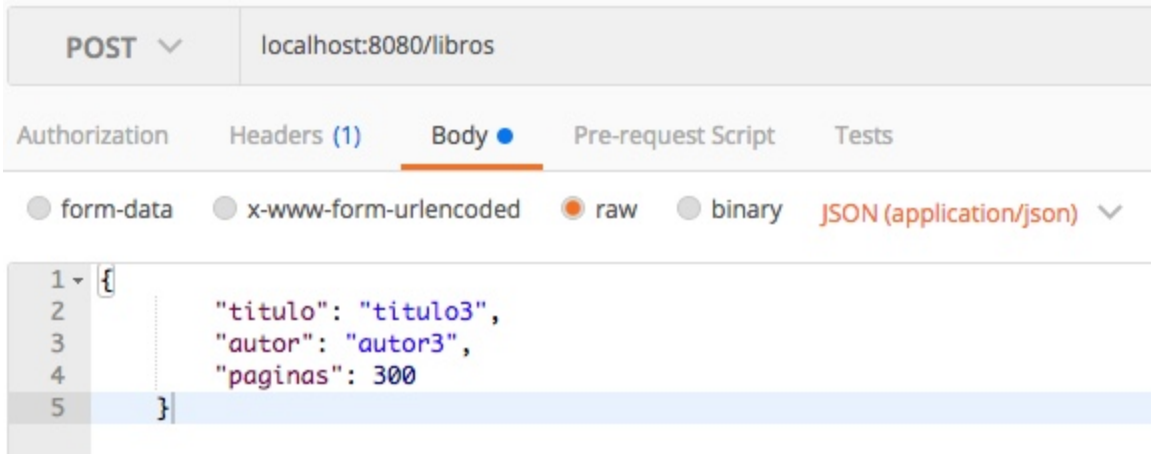
Nada más pulsar nos aparece una pestaña en donde podemos rellenar la información de la petición y ejecutarla a través del método send.



Obtendremos el resultado de una forma muy clara:

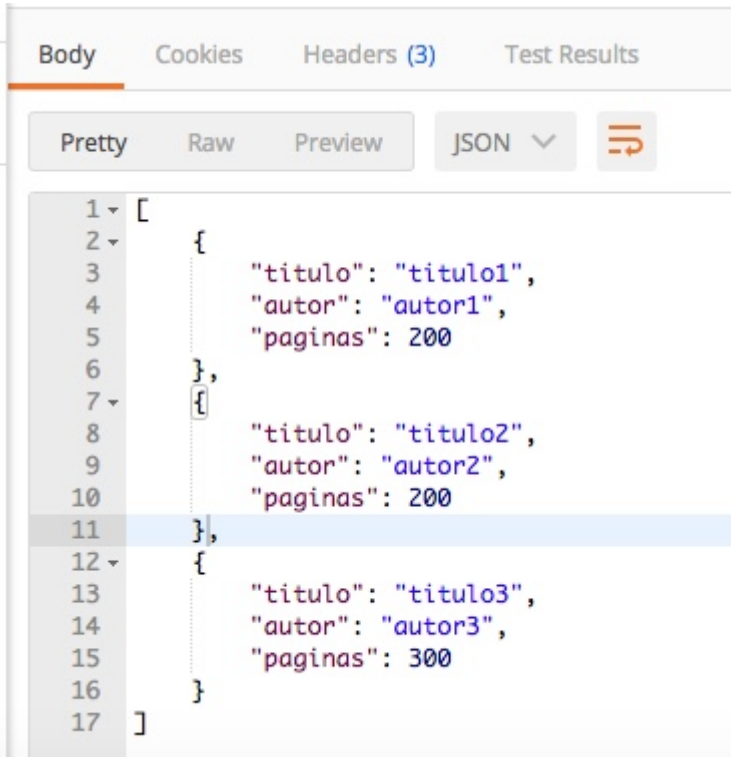


Esta primera operación no aporta mucho respecto a una petición clásica. Sin embargo, si realizamos un POST las ventajas son claras. Vamos a crear una nueva pestaña y configurar la petición en donde adjuntamos los datos en JSON de forma muy sencilla.



Realizamos una petición POST que recibirá nuestra aplicación de Spring MVC con muy poco trabajo. Acto seguido volvemos a realizar la primera petición GET que nos devolvería un libro adicional.

PostMan App con Spring REST



Acabamos de utilizar Postman para gestionar nuestras peticiones HTTP. Esta herramienta nos permite gestionar todo tipo de peticiones y formatos desde headers, tokens, peticiones json, peticiones xml etc.