Acabamos de ver como usar Java equals y hashcode

Spring @PathVariable es la anotación que nos sirve dentro de Spring framework para configurar variables dentro de los propios segmentos de la URL algo que a nivel de Arquitecturas REST es imprescindible. Vamos a ver un par de ejemplos de como construir unas urls REST que se apoyen en este tipo de anotación. Para ello vamos a construir la clase Facturas y la clase LineaFactura que tienen una relación de 1 a n y nos ayude a entender mejor este tipo de variables.

```
package com.arquitecturajjava.web;

import java.util.ArrayList;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

public class Factura {

  private int numero;
  private String concepto;
  @JsonIgnore
  private List<LineaFactura> lineas= new ArrayList<LineaFactura>();
  public Factura(int numero, String concepto) {
    super();
    this.numero = numero;
    this.concepto = concepto;

  }
  public int getNumero() {
    return numero;
  }
```

```
  public void setNumero(int numero) {
    this.numero = numero;
  }
  public String getConcepto() {
    return concepto;
  }
  public void setConcepto(String concepto) {
    this.concepto = concepto;
  }
  public List<LineaFactura> getLineas() {
    return lineas;
  }
  public void setLineas(List<LineaFactura> linea) {
    this.lineas = linea;
  }
  public void addLinea(LineaFactura linea) {
    this.lineas.add(linea);
  }
}
```

```
package com.arquitecturajjava.web;

public class LineaFactura {

  private int numero;
  private String concepto;
  private double importe;
  public LineaFactura(int numero, String concepto, double importe) {
    super();
    this.numero = numero;
    this.concepto = concepto;
    this.importe = importe;
```

```
  }
  public int getNumero() {
    return numero;
  }
  public void setNumero(int numero) {
    this.numero = numero;
  }
  public String getConcepto() {
    return concepto;
  }
  public void setConcepto(String concepto) {
    this.concepto = concepto;
  }
  public double getImporte() {
    return importe;
  }
  public void setImporte(double importe) {
    this.importe = importe;
  }
}
```

Tenemos dos clases relacionadas . Podemos usar un clásico despliegue de Spring Boot para una aplicación web y generar un Servicio REST que incluya el acceso a una lista de Facturas y Lineas como variables estáticas:

```
package com.arquitecturajjava.web;

import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

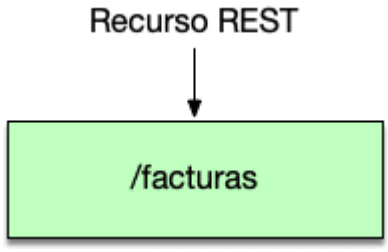
```
import java.util.Optional;

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

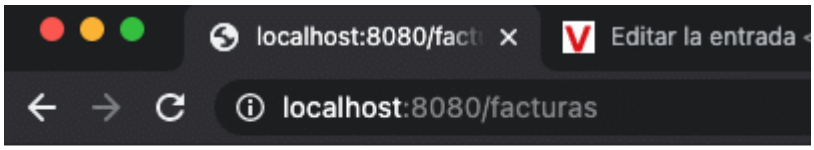
@RestController
@RequestMapping("/facturas")
public class FacturaREST {

  static List<Factura>  lista= new ArrayList<Factura>();
  static {
    Factura f= new Factura(1,"informatica");
    f.addLinea(new LineaFactura(1,"auricular",200));
    f.addLinea(new LineaFactura(2,"telefono",300));
    Factura f2= new Factura(2,"alimentacion");
    f2.addLinea(new LineaFactura(1,"galletas",2));
    f2.addLinea(new LineaFactura(2,"leche",1));
    Factura f3= new Factura(1,"limpieza");
    f3.addLinea(new LineaFactura(1,"gel",2));
    f3.addLinea(new LineaFactura(2,"jabon",4));
    lista.add(f);
    lista.add(f2);
    lista.add(f3);
  }
  @RequestMapping
  public List<Factura> buscarTodas() {
    return lista;
  }
```

En un primer momento solo tenemos la url de Facturas.



Con la cual accedemos al listado por completo :

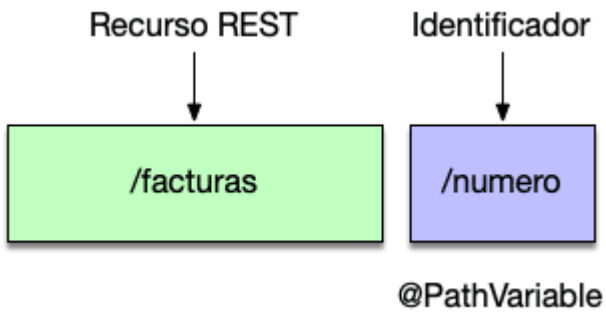



```
[{"numero":1,"concepto":"informatica"},
{"numero":2,"concepto":"alimentacion"},
{"numero":1,"concepto":"limpieza"}]
```

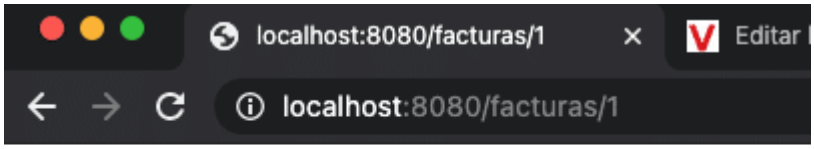

Es momento de añadir una url que nos devuelve una única Factura y para ello usaremos @PathVariable que nos permite añadir una variable en nuestros métodos para filtrar la lista y quedarnos con un único elemento:

```
@RequestMapping("{numero}")
  public Optional<Factura> buscarUna( @PathVariable int numero) {
    return
lista.stream().filter((f)->f.getNumero()==numero).findFirst();
  }
```

La estructura es :



En este caso la url de filtrado será /facturas/1



{"numero":1,"concepto":"informatica"}

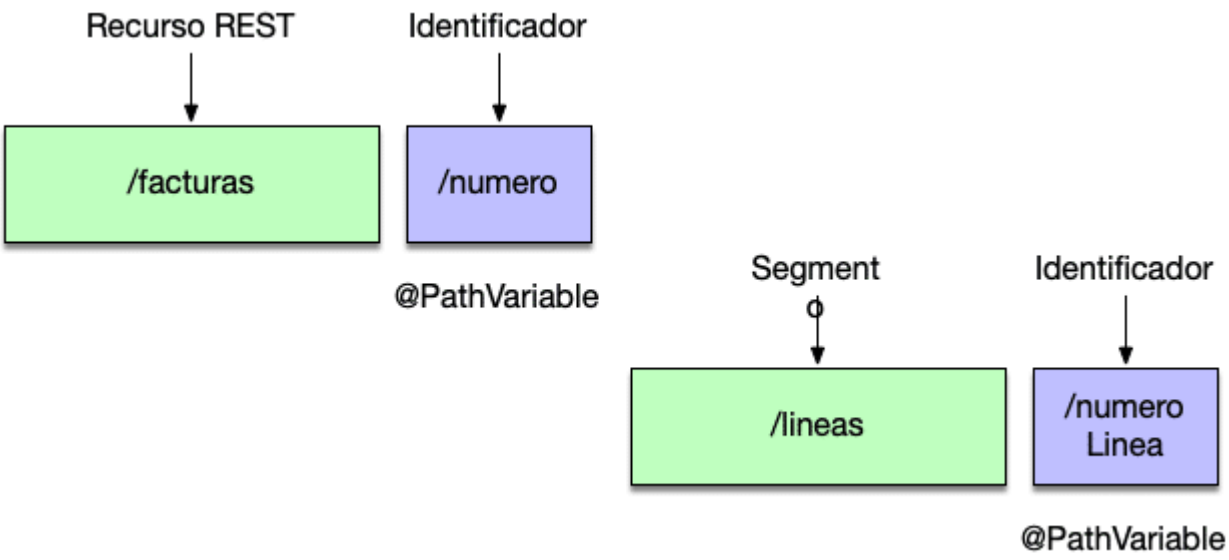
Hemos añadido una variable a la url de acceso :

Spring @PathVariable

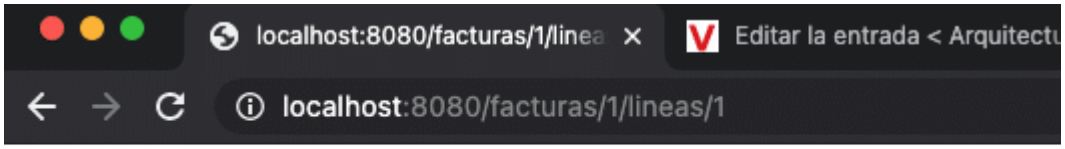
El uso de @PathVariable es muy común ya que en muchas casuísticas tenemos la estructura en forma de agregado , es decir. ¿Cómo podemos acceder a la linea de una factura concreta? . Es relativamente sencilla basta con añadir un @PathVariable adicional y generar la estructura deseada.

```
@RequestMapping("{numero}/lineas/{numerolinea}")
  public Optional<LineaFactura> buscarUna( @PathVariable int
numero,@PathVariable int numerolinea) {
    Optional<Factura>
factura=lista.stream().filter((f)->f.getNumero()==numero).findFirst();
    if (factura.isPresent()) {
      return factura.get()
.getLineas().stream().filter((f)->f.getNumero()==numerolinea).findFirs
t();
    }
    return Optional.empty();
  }
```

Aquí podemos ver como cada segmento incluye {} para identificar a ambas variables con diferente nombre {numero} y {numerolinea} .



Podemos verlo en el explorador :




```
{"numero":1,"concepto":"auricular","importe":200.0}
```


Otros artículos relacionados

1. REST API y su inmutabilidad
2. REST Resources Nombres vs Verbos
3. Spring WebFlux Router y REST
4. Spring MVC