Hacer uso de Spring REST PUT es muy habitual cuando queremos actualizar datos vía REST en nuestras aplicaciones. Vamos a ver un ejemplo elemental de como realizar esta operación de forma rápida. Lo primero que vamos a utilizar es un proyecto de Spring Boot que nos permita configurar un servicio REST con la anotación de @RestController.

```
<dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

En este caso simplemente necesitamos añadir a las dependencias el starter de Web . Una vez tenemos esto construido el siguiente paso es crearnos una clase Java de Persona con las propiedades básicas.

```
package com.arquitectura.restput;

public class Persona {

  private String nombre;
  private String apellidos;
  private int edad;

  public Persona() {
    super();
```

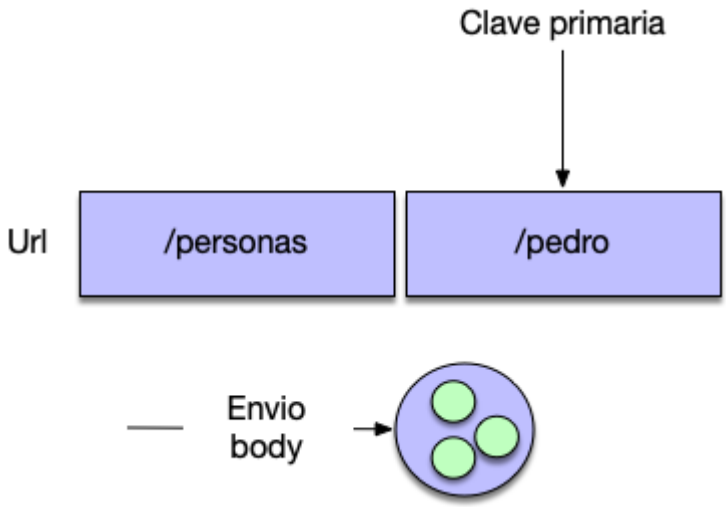
```
}

public Persona(String nombre, String apellidos, int edad) {
  super();
  this.nombre = nombre;
  this.apellidos = apellidos;
  this.edad = 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;
```

```
  }

  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((nombre == null) ? 0 :
nombre.hashCode());
    return result;
  }

  @Override
  public boolean equals(Object obj) {
    if (this == obj)
      return true;
    if (obj == null)
      return false;
    if (getClass() != obj.getClass())
      return false;
    Persona other = (Persona) obj;
    if (nombre == null) {
      if (other.nombre != null)
        return false;
    } else if (!nombre.equals(other.nombre))
      return false;
    return true;
  }

  public Persona(String nombre) {
    super();
    this.nombre = nombre;
```

```
  }

}
```

Spring REST PUT

Es momento de construir el servicio que sea capaz de devolvernos una lista de personas así como de actualizarla. Una petición PUT debe adjuntar a la url la clave primaría del recurso que deseamos actualizar. En este caso vamos a suponer que es el nombre . Aparte de esto debe enviar en formato JSON o XML el nuevo recurso que vamos a almacenar. De tal forma que a través de la clave primaria podamos elegir el recurso antiguo y actualizarlo por el nuevo.



Vamos a verlo en código con un @RestController.

```
package com.arquitectura.restput;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
```

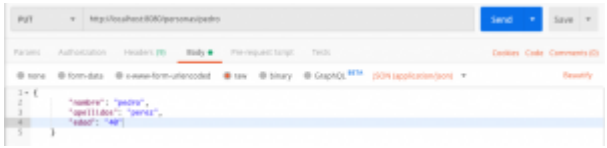
```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController()
@RequestMapping("/personas")
public class PersonaREST {

  static List<Persona> listaPersonas = new ArrayList<Persona>();
  static {

    listaPersonas.add(new Persona("pedro", "perez", 20));
    listaPersonas.add(new Persona("ana", "gomez", 30));
  }

  @GetMapping
  public List<Persona> lista() {

    return listaPersonas;
  }

  @PutMapping("/{nombre}")
  public void actualizar(@PathVariable("nombre") String nombre,
@RequestBody Persona persona) {
    listaPersonas.remove(new Persona(nombre));
    listaPersonas.add(persona);
  }
```

```
}
```

Si usamos Postman y realizamos una petición GET obtendremos la lista inicial de objetos que tenemos en memoria.

```
[
    {
        "nombre": "pedro",
        "apellidos": "perez",
        "edad": 20
    },
    {
        "nombre": "ana",
        "apellidos": "gomez",
        "edad": 30
    }
]
```

Si usamos Postman y construimos una petición Put enviando un objeto en formato JSON al servidor y pasando la clave primaria del recurso.

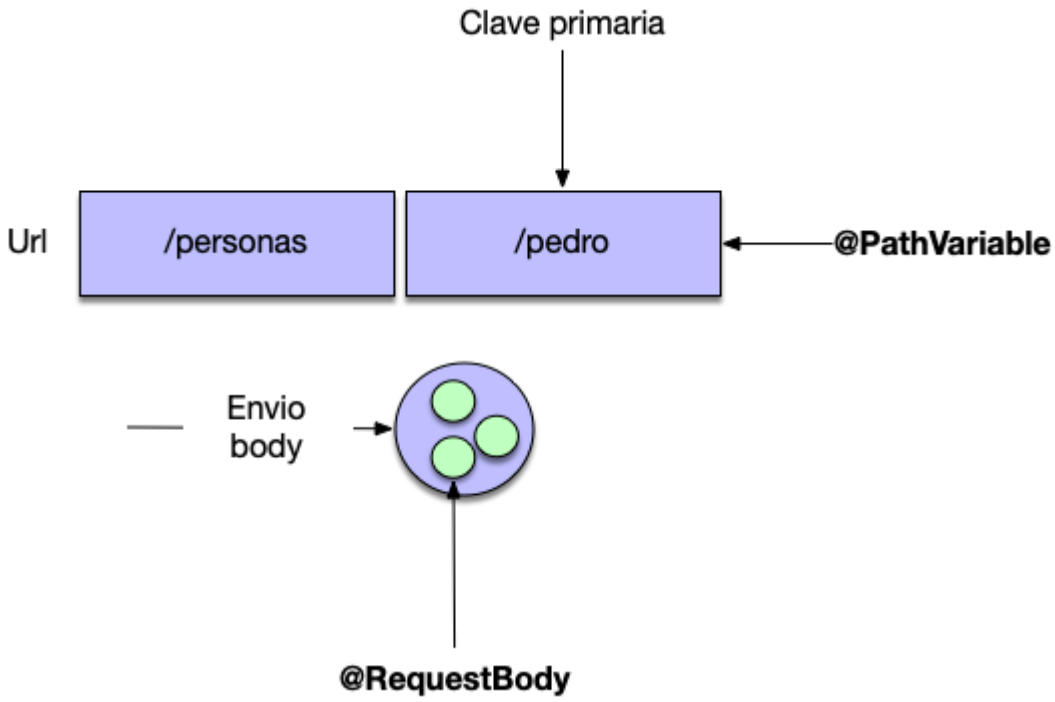


El objeto se actualizará y la nueva lista nos mostrará el recurso de pedro actualizado.

```
[
    {
        "nombre": "ana",
        "apellidos": "gomez",
        "edad": 30
```

```
    },
    {
        "nombre": "pedro",
        "apellidos": "perez",
        "edad": 40
    }
]
```

Para que esta funcionalidad se ejecutara correctamente hemos tenido que utilizar las anotaciones de @PathVariable que permite adquirir una variable a partir de un valor de la url y @RequestBody que permite generar un objeto a partir de los datos que nos llegan vía JSON.



Acabamos de realizar una petición con Spring REST PUT.

Otros artículos relacionados

1. Spring REST CORS y su configuración
2. Spring REST Test utilizando Rest Assured
3. Spring REST Service con @RestController
4. PUT vs POST