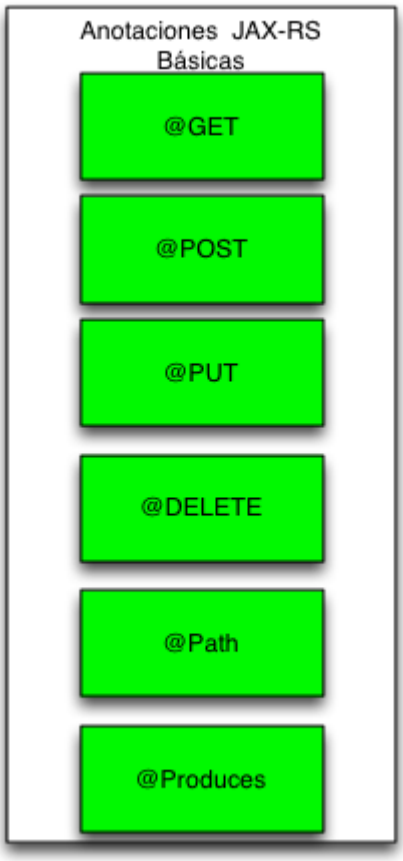
Spring WebFlux Router es uno de los enfoques innovadores que tiene Spring Framework a la hora de construir arquitecturas REST para programación Reactiva . Normalmente a partir de Spring 4 cuando nosotros necesitamos construir un servicio REST lo hacemos con la anotación @RestController que de una forma sencilla gestiona el servicio REST y lo hace muy sencillo de construir. Vamos a ver un ejemplo sencilo apoyándonos en Spring Boot. Lo primero que vamos a hacer es nos vamos a instalar los Starters necesarios de Boot , para ello mostramos el contenido del pom.xml.

Vamos a construir un servicio REST utilizando los estandares de JAX-RS . Para ello lo primero que tenemos que hacer es comenzar a conocer las distintas anotaciones que el estandar define.



@GET :Esta anotación marca un método y define una operación GET .Es similar a cuando realizamos una petición HTTP GET y solo debe usarse en el caso que de queramos leer

información .Nunca a la hora de escribir o modificar el estado del recurso al que estemos accediendo.

@POST :Esta anotación marca un método y define una operación POST .Es similar a cuando realizamos una petición HTTP POST y se usa para añadir un recurso o modificar un recurso existente.

@DELETE :Esta anotación marca un método y define una operación DELETE .Como su nombre indica se trata de eliminar un recurso del servidor . No siempre se usa ya que redirecciona a traves de POST cuando trabajamos con HTML plano.

@PUT : Esta anotación se encargar de reemplazar un recurso del servidor y como en el caso anterior suele redireccionarse a traves de POST.

@Path :Esta anotación es distinta a las anteriores y se encarga de definir un punto de entrada al servicio. Puede usarse tanto a nivel de clase como a nivel de método.

@Produces :Esta anotación se encarga de que el contenido del servicio REST sea generado con distintos formatos.En nuestro caso usaremos JSON

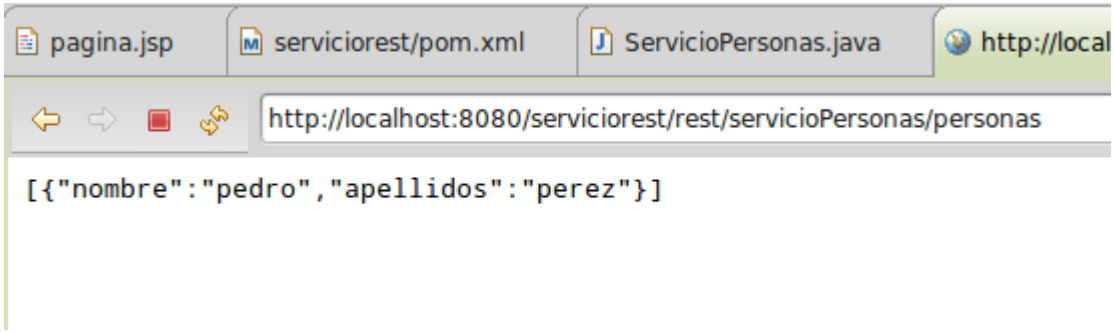
Vamos a ver un ejemplo de código sencillo.

```
package com.arquitecturajava.serviciosexternos;

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

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
```

```
import javax.ws.rs.core.MultivaluedMap;

@Path("/servicioPersonas/")
@Produces("application/json")
public class ServicioPersonas {

private static List&amp;lt;Persona&amp;gt; listaPersonas = new
ArrayList&amp;lt;Persona&amp;gt;();

public ServicioPersonas() {
 super();
 Persona yo = new Persona("pedro", "perez");
 listaPersonas.add(yo);
 }

@GET
 @Path("/personas")
 public List&amp;lt;Persona&amp;gt; getPersonas() {

return listaPersonas;
 }

@POST
 @Path("/personas")
 public void addPersona(MultivaluedMap&amp;lt;String, String&amp;gt;
parametros) {

Persona p = new Persona(parametros.getFirst("nombre"),
 parametros.getFirst("apellidos"));
 listaPersonas.add(p);
 }
```

```
}
```

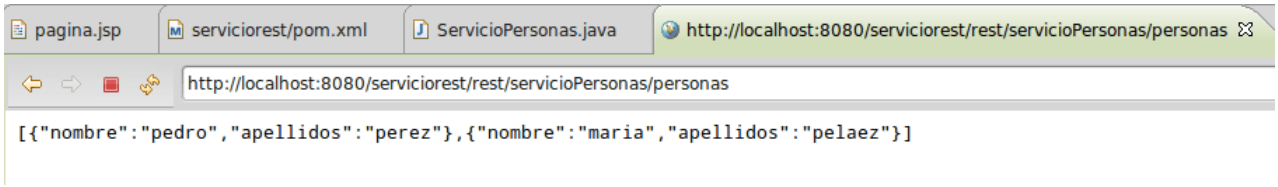
En este caso hemos utilizado un servicio REST que se encarga de mostrar las personas que tenemos almacenas en una variable “listaPersonas” que al ser estatica se mantiene en memoria. Por lo tanto definir un método GET es suficiente para acceder a la información. Este método se apoya en la anotación @Produces para generar información en formato JSON.



Una vez hemos invocado la primera vez al servicio via URL podemos usar un formulario HTML para invocar al metodo /personaa pasando los parametros por POST y añadiendo nuevas personas a la lista

```
&amp;lt;html&amp;gt;
&amp;lt;body&amp;gt;
&amp;lt;form method="POST"
action="rest/servicioPersonas/personas"&amp;gt;
&amp;lt;input type="text" name="nombre"/&amp;gt;
&amp;lt;input type="text" name="apellidos"/&amp;gt;
&amp;lt;input type="submit"/&amp;gt;
&amp;lt;/form&amp;gt;
&amp;lt;/body&amp;gt;
&amp;lt;/html&amp;gt;
```

Realizada esta operación la lista de personas queda modificada.



Spring WebFlux Router es uno de los enfoques innovadores que tiene Spring Framework a la hora de construir arquitecturas REST para programación Reactiva . Normalmente a partir de Spring 4 cuando nosotros necesitamos construir un servicio REST lo hacemos con la anotación @RestController que de una forma sencilla gestiona el servicio REST y lo hace muy sencillo de construir. Vamos a ver un ejemplo sencilo apoyándonos en Spring Boot. Lo primero que vamos a hacer es nos vamos a instalar los Starters necesarios de Boot , para ello mostramos el contenido del pom.xml.

Aunque el ejemplo es muy sencillo muchas veces es mas complicado configurar algún framework para que pueda construir estos servicios .En el siguiente POST hablaremos de como configurar Apache CXF para que este servicio REST funcione sin problemas.