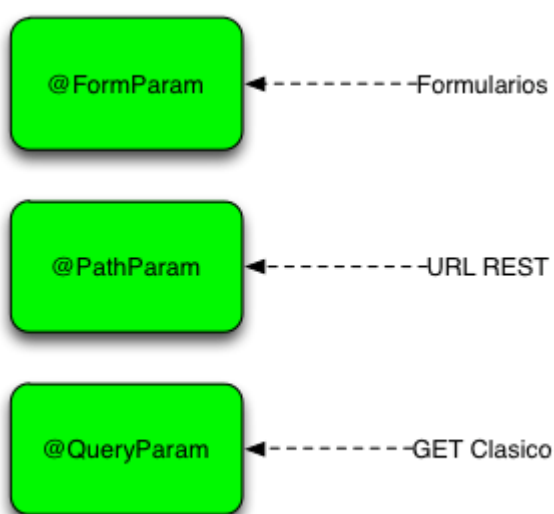


Acabamos de ver como usar Java equals y hashCode

En Java EE el conjunto de anotaciones para Servicios REST es amplio .Voy a comentar a continuación varias de las mas utilizadas cuando creamos servicios de este tipo dentro del mundo de los standares de Java.



`@FormParam` :Esta anotación es una anotación a nivel de parámetro y sirve para ligar parámetros de un formulario HTML con variables del servicio REST .

`@PathParam`: Esta anotación es una anotación a nivel de parámetro y sirve para ligar parámetros de una petición clasica REST con las variables del servicio como por ejemplo `/persona/nombre/pedro`. En donde la variable nombre tiene asignado el valor de pedro.

`@QueryParam`: Esta anotación es una anotación útil ya que nos permite leer parámetros que vengan a traves de una petición clásica GET como por ejemplo `http://url?nombre=pedro`

A continuación podemos ver un servicio REST similar a los anteriores que se apoya en este tipo de anotaciones para clarificar el código .En el se han usado anotaciones `@POST` para substituir `@PUT` y `@DELETE` y se han añadido las anotaciones comentadas antes.

```
Path("/servicioPersonasAnotaciones/")
@Produces("application/json")
public class ServicioPersonasAnotaciones {

    private static
    List<&&&&&&&&<Persona&&&&&&&&>
    listaPersonas = new
    ArrayList<&&&&&&&&<Persona&&&&&&&&>
    t();

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

    @GET
    @Path("/personas")
    public
    List<&&&&&&&&<Persona&&&&&&&&>
    getPersonas() {

        return listaPersonas;
    }

    @POST
    @Path("/persona/add")
    public void addPersona(@FormParam("nombre") String nombre,
        @FormParam("apellidos") String apellidos) {
```

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

```
@GET
```

```
@Path("/persona/nombre/{nombre}")
```

```
public Persona buscarPersona(@PathParam("nombre") String nombre) {
```

```
Persona persona = null;
```

```
for (Persona p : listaPersonas)
```

```
if (nombre.equals(p.getNombre())) {
```

```
persona = p;
```

```
}
```

```
return persona;
```

```
}
```

```
@GET
```

```
@Path("/persona")
```

```
public Persona buscarPersonaClasico(@QueryParam("nombre")String  
nombre) {
```

```
Persona persona = null;
```

```
for (Persona p : listaPersonas)
```

```
if (nombre.equals(p.getNombre())) {
```

```
persona = p;
```

```
}
```

```
return persona;  
}
```

```
@POST  
@Path("/persona/delete")  
public void borrarPersona(@FormParam("nombre") String nombre) {
```

```
Persona p = new Persona();  
p.setNombre(nombre);  
listaPersonas.remove(p);  
  
}
```

```
@POST  
@Path("/persona/replace/")  
public void reemplazar(@FormParam("nombre") String nombre,  
@FormParam("apellidos") String apellidos) {
```

```
Persona persona = new Persona(nombre, apellidos);  
listaPersonas.remove(persona);  
listaPersonas.add(persona);  
  
}
```

Para clarificar un poco el contenido del código vamos a revisar el método buscarPersona .El cual será ejecutado cuando la petición sea similar a la siguiente.

/persona/nombre/pedro

El código de nuestro método se muestra a continuación:

```
@GET
@Path("/persona/nombre/{nombre}")
public Persona buscarPersona(@PathParam("nombre") String nombre) {

    Persona persona = null;
    for (Persona p : listaPersonas)

        if (nombre.equals(p.getNombre())) {

            persona = p;
        }

    return persona;
}
```

La anotación `@PathParam` que precede al parámetro “nombre” del método será la encargada de mapear el valor “pedro” a la variable nombre

```
@PathParam("nombre") String nombre
```

.De esta forma todo será mucho mas sencillo . El resto de anotaciones realiza asignaciones similares pero apoyandose en los parámetros enviados por POST o por QueryString.