Desarrollo de Servicios Web 1

Tema Nº13:SPRING REST

Indicador de logro Nº13:Aplica Spring Rest a través de un proyecto Spring implementando las principales anotaciones @GET, @POST, @DELETE, @PUT, @RestController, @RequestMapping y @ResponseBody.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº13:**

SPRING REST

**Subtema 13.1:**

Servicios REST con Spring Framework

**MARCO TEÓRICO**

Esto es un producto de la familia de Spring, que hace simple las invocaciones hacia servicios web complejos y simples. En general esto se enfoca en la creación de document-drive Web Services. El objetivo es facilitar el contrato del primer desarrollo de servicios SOAP, permitiendo la creación flexible de servicios web usando una o muchas formas de manipular los XML payloads.

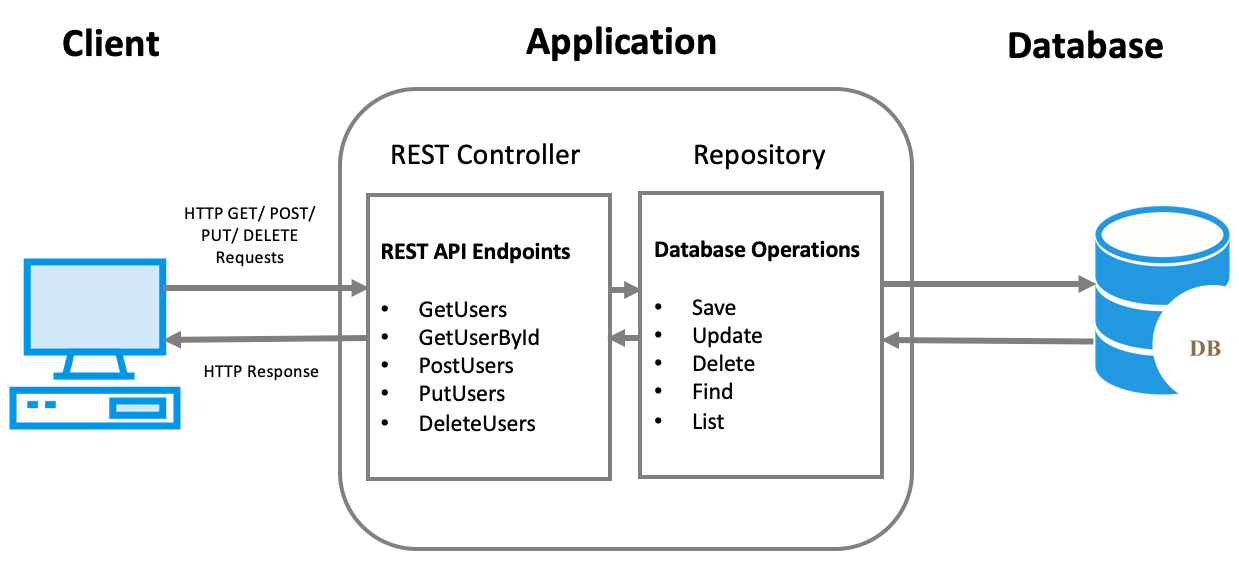
**REST**

Define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST emergió en los últimos años como el modelo predominante para el diseño de servicios.

De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.

Cuando se habla de REST y RESTful no se habla de cosas diferentes, solo de algunas consideraciones:

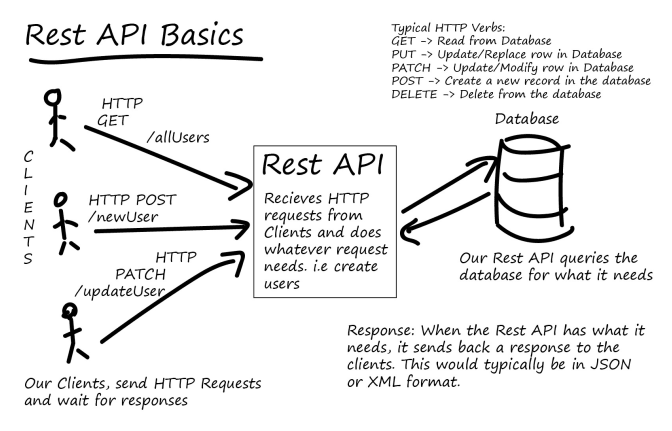
* REST: Son los principios de arquitectura de software.
* RESTful: Son los servicios web que siguen esos principios que implementa la arquitectura REST.



**Principios de la Arquitectura REST**

* Recurso: Representados por un formato (jpeg, mpeg, html, xml, etc.) Estos recursos se pueden ver en internet.
* Identificador Único: Cada recurso debe ser identificable por un identificador único (URI única).
* Métodos HTTP estándar: Acciones o verbos como GET, PUT, POST, DELETE, HEAD, OPTIONS, TRACE, CONNECT.
* GET: Para obtener un recurso.
* PUT: Para crear un recurso en el servidor.
* POST: Para cambiar un recurso o actualizarlo.
* DELETE: Para eliminar un recurso.
* HEAD: Para saber si existe un recurso.
* OPTIONS: Para retornar los métodos soportados por el servidor.
* TRACE: Para solicitar retorno del mensaje enviado.
* CONNECT: Convierte el request a un túnel TCP/IP.

Representación de Rest API Basics:



**Lista - Rest Framework**



* Apache CXF
* Jersey
* Rest Easy
* Restlet
* Spring MVC

**Subtema 13.2:**

REST – Spring MVC

**MARCO TEÓRICO**

[Spring Framework](https://projects.spring.io/spring-framework/) es un [completo marco](https://docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html) que permite crear aplicaciones empresariales Java. Las capacidades REST son proporcionadas por el módulo [Spring MVC](https://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html) (el mismo módulo que proporciona modelo-vista-controlador). No es una implementación de JAX-RS y puede verse como una alternativa de Spring al estándar de JAX-RS.

El ecosistema de Spring también proporciona un [amplio rango de proyectos](https://spring.io/projects) para crear aplicaciones empresariales, que cubren la persistencia, la seguridad, la integración con redes sociales, el procesamiento por lotes, etc.

API JAX-RS

@Path("/greetings")

public class JaxRsController {

@GET

@Path("/{name}")

@Produces(MediaType.TEXT\_PLAIN)

public Response greeting(@PathParam("name") String name) {

String greeting="Hello "+name;

return Response.ok(greeting).build();

}

}

API Spring MVC

@RestController

@RequestMapping("/greetings")

public class SpringRestController {

@RequestMapping(method=RequestMethod.GET, value="/{name}",

produces=MediaType.TEXT\_PLAIN\_VALUE)

public ResponseEntity<?> greeting(@PathVariable String name) {

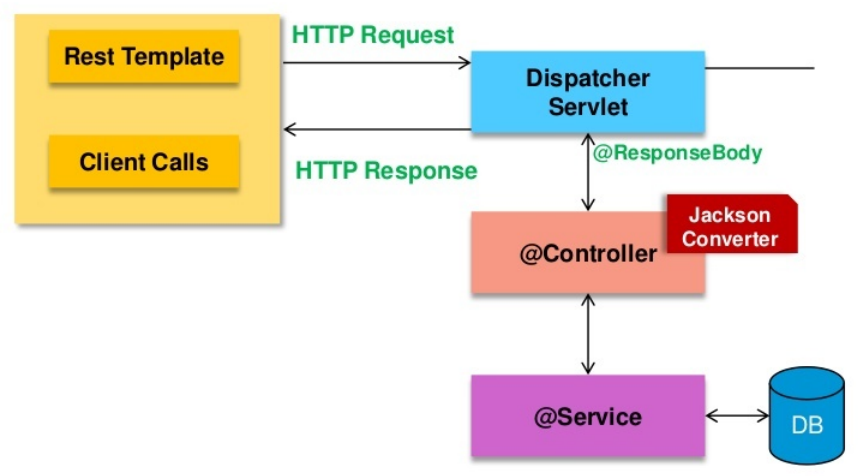
String greeting="Hello "+name;

return new ResponseEntity<>(greeting,HttpStatus.OK);

}

}

Rest calls in Spring



**1. MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

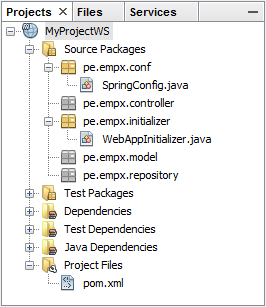
1. EQUIPO

Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS

* NetBeans y/o STS IDE
* Java Developer Kit (JDK) version 8.x
* Apache Maven
* spring-webmvc
* jackson-core
* jackson-databind
* slf4j-api
* jetty-maven-plugin

**2. PROCEDIMIENTO**

Caso aplicativo

Dependencias:

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

<version>2.4.1</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.4.1.1</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.12</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.7.12</version>

</dependency>

<!-- plugin para: jetty embebido en el proyecto maven -->

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<configuration>

<scanIntervalSeconds>10</scanIntervalSeconds>

<webApp><contextPath>/api</contextPath></webApp>

</configuration>

</plugin>

Clase VO



package pe.empx.model;

public class Producto {

private Long id\_producto;

private String descripcion;

private String presentacion;

private double precio;

private int cantidad;

public Producto() {}

public Producto(Long id\_producto, String descripcion, String presentacion,

double precio, int cantidad) {

this.id\_producto = id\_producto;

this.descripcion = descripcion;

this.presentacion = presentacion;

this.precio = precio;

this.cantidad = cantidad;

}

//Getters and Setters

}

Clase Repository



package pe.empx.repository;

import java.util.\*;

import org.springframework.stereotype.\*;

import pe.empx.model.\*;

@Repository

public class ProductoRepository {

private static Map<Long,Producto> *DBProductos*=new HashMap<>();

static {

Producto producto1=new Producto(1L,"Caramelos","Bolsa",5.00,100);

Producto producto2=new Producto(2L,"Chocolates","Caja",8.00,50);

Producto producto3=new Producto(3L,"Galletas","Bolsa",10.00,70);

*DBProductos*.put(producto1.getId\_producto(),producto1);

*DBProductos*.put(producto2.getId\_producto(),producto2);

*DBProductos*.put(producto3.getId\_producto(),producto3);

}

public Producto insert(Producto producto) {

producto.setId\_producto(new Long(*DBProductos*.size()+1));

*DBProductos*.put(producto.getId\_producto(),producto);

return producto;

}

public Producto update(Producto producto) {

*DBProductos*.put(producto.getId\_producto(),producto);

return producto;

}

public void delete(Long id\_producto) {

*DBProductos*.remove(id\_producto);

}

public Producto findById(Long id\_producto) {

return *DBProductos*.get(id\_producto);

}

public Collection<Producto> findAll() {

return new ArrayList<>(*DBProductos*.values());

}

public Collection<Producto> findAll(String presentacion) {

List<Producto> list=new ArrayList<>();

for(Producto card:new ArrayList<>(*DBProductos*.values())) {

if(card.getPresentacion().equals(presentacion))

list.add(card);

}

return list;

}

}

Clase Controller



package pe.empx.controller;

import java.util.\*;

import org.springframework.beans.factory.annotation.\*;

import org.springframework.http.\*;

import org.springframework.web.bind.annotation.\*;

import pe.empx.model.\*;

import pe.empx.repository.\*;

@RestController //indica que es un Servicio REST

@RequestMapping(value="/v1")

public class ProductoController

{

@Autowired //inyección de dependencia

private ProductoRepository repository;

public ProductoController() {}

@RequestMapping(value="/producto",method=RequestMethod.PUT,

headers="Accept=application/json")

public @ResponseBody ResponseEntity<?> insert(@RequestBody Producto producto) {

return new ResponseEntity<>(repository.insert(producto),HttpStatus.OK);

}

@RequestMapping(value="/producto",method=RequestMethod.POST,

headers="Accept=application/json")

public @ResponseBody ResponseEntity<?> update(@RequestBody Producto producto)

{

if(repository.isExists(producto.getId\_producto())) {

repository.update(producto);

return new ResponseEntity<>("Producto de ID "+producto.getId\_producto()+

" actualizado con éxito.",HttpStatus.OK);

}

//mensaje de error

return new ResponseEntity<>("Error... Producto no existe.",HttpStatus.BAD\_REQUEST);

}

@RequestMapping(value="/producto/{id\_producto}",method=RequestMethod.DELETE,

headers="Accept=application/json")

public @ResponseBody ResponseEntity<?> delete(@PathVariable("id\_producto")

Long id\_producto)

{

//si id\_producto existe

if(repository.isExists(id\_producto)) {

repository.delete(id\_producto);

return new ResponseEntity<>("Producto de ID "+id\_producto+

" eliminado con éxito.",HttpStatus.OK);

}

//mensaje de error

return new ResponseEntity<>("Error... Producto no existe.",HttpStatus.BAD\_REQUEST);

}

@RequestMapping(value="/producto/{id\_producto}",method=RequestMethod.GET,

headers="Accept=application/json")

public @ResponseBody ResponseEntity<?> getById(@PathVariable("id\_producto")

Long id\_producto) {

//si id\_producto existe

if(repository.isExists(id\_producto)) {

return new ResponseEntity<>(repository.findById(id\_producto),HttpStatus.OK);

}

//mensaje de error

return new ResponseEntity<>("Error... Producto no existe.",HttpStatus.BAD\_REQUEST);

}

@RequestMapping(value="/productos",method=RequestMethod.GET,

headers="Accept=application/json")

public @ResponseBody Collection<Producto> findAll() {

return repository.findAll();

}

@RequestMapping(value="/productos",method=RequestMethod.GET,

headers="Accept=application/json",params={"presentacion"})

public @ResponseBody Collection<Producto>

findAll(@RequestParam(value="presentacion",required=true)

String presentacion) {

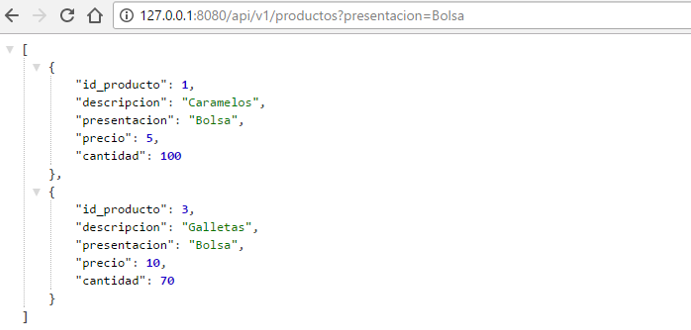
return repository.findAll(presentacion);

}

}

Pruebas unitarias







**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué diferencia existen entre JAX-RS y Spring Rest?
* ¿La anotación @QueryParam cuál es su equivalente en Spring MVC?
* ¿La anotación @RestController en qué se diferencia del @Controller?
* ¿Spring Boot es mejor que Spring Framework para los servicios Rest, dar una breve explicación?
* Mediante Netbeans y/o Eclipse IDE, Base de datos, Apache Mavent y Spring Rest crear web service para las operaciones a la BD. Realizar pruebas unitarias.

1. **ENLACES DE REFERENCIA**

* https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful
* https://www.baeldung.com/spring-mvc
* https://www.baeldung.com/building-a-restful-web-service-with-spring-and-java-based-configuration

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_