SPRING BOOT + ANGULAR +STRIPE

# PAGOS CON STRIPE EN SPRING BOOT Y ANGULAR 8. CAPÍTULO 3: CRUD

Los que seguís mis artículos habéis podido comprobar que en todos los tutoriales que llevo sobre Angular y Spring Boot hay que implementar las operaciones CRUD sobre una o más clases del back. Por ello voy a ser lo más conciso posible dado que es algo que ya hemos visto hasta la saciedad. Si estás comenzando con Spring te recomiendo que revises los artículos anteriores que tengo sobre CRUD.

Lo primero que vamos a implementar es la clase Articulo, en el paquete model,  que representa la entidad:

|  |  |
| --- | --- |
|  | package com.cavanosa.TutorialStripe.model; |
|  |  |
|  | import javax.persistence.\*; |
|  | import javax.validation.constraints.NotBlank; |
|  | import javax.validation.constraints.NotNull; |
|  |  |
|  | @Entity |
|  | public class Articulo { |
|  | @Id |
|  | @GeneratedValue(strategy = GenerationType.IDENTITY) |
|  | private long id; |
|  | @NotBlank |
|  | @Column(unique = true) |
|  | private String nombre; |
|  | private String descripcion; |
|  | @NotNull |
|  | private int precio; |
|  | private String imagenURL; |
|  |  |
|  | public long getId() { |
|  | return id; |
|  | } |
|  |  |
|  | public void setId(long id) { |
|  | this.id = id; |
|  | } |
|  |  |
|  | public String getNombre() { |
|  | return nombre; |
|  | } |
|  |  |
|  | public void setNombre(String nombre) { |
|  | this.nombre = nombre; |
|  | } |
|  |  |
|  | public String getDescripcion() { |
|  | return descripcion; |
|  | } |
|  |  |
|  | public void setDescripcion(String descripcion) { |
|  | this.descripcion = descripcion; |
|  | } |
|  |  |
|  | public int getPrecio() { |
|  | return precio; |
|  | } |
|  |  |
|  | public void setPrecio(int precio) { |
|  | this.precio = precio; |
|  | } |
|  |  |
|  | public String getImagenURL() { |
|  | return imagenURL; |
|  | } |
|  |  |
|  | public void setImagenURL(String imagenURL) { |
|  | this.imagenURL = imagenURL; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/96407d7c7cd51b3ef65d0c9270fd4224/raw/04536809a8cbee5a4c7fabb099419813d6b97cff/articulo.java)[**articulo.java**](https://gist.github.com/cavanosa/96407d7c7cd51b3ef65d0c9270fd4224#file-articulo-java) hosted with  by [**GitHub**](https://github.com/)

Luego, la interfaz ArticuloRepository:

|  |  |
| --- | --- |
|  | package com.cavanosa.TutorialStripe.repository; |
|  |  |
|  | import com.cavanosa.TutorialStripe.model.Articulo; |
|  | import org.springframework.data.jpa.repository.JpaRepository; |
|  | import org.springframework.stereotype.Repository; |
|  |  |
|  | import java.util.Optional; |
|  |  |
|  | @Repository |
|  | public interface ArticuloRepository extends JpaRepository<Articulo, Long> { |
|  | Optional<Articulo> findByNombre(String nombre); |
|  | boolean existsByNombre(String nombre); |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/77812c62d9f67af7c0c430c1c32f062f/raw/2cd5537f08ed6a1485b6d0bb502a82b4aa6388d0/articulorepository.java)[**articulorepository.java**](https://gist.github.com/cavanosa/77812c62d9f67af7c0c430c1c32f062f#file-articulorepository-java) hosted with  by [**GitHub**](https://github.com/)

El servicio:

|  |  |
| --- | --- |
|  | package com.cavanosa.TutorialStripe.service; |
|  |  |
|  | import com.cavanosa.TutorialStripe.model.Articulo; |
|  | import com.cavanosa.TutorialStripe.repository.ArticuloRepository; |
|  | import org.springframework.beans.factory.annotation.Autowired; |
|  | import org.springframework.stereotype.Service; |
|  |  |
|  | import java.util.List; |
|  | import java.util.Optional; |
|  |  |
|  | @Service |
|  | public class ArticuloService { |
|  |  |
|  | @Autowired |
|  | ArticuloRepository articuloRepository; |
|  |  |
|  | public List<Articulo> lista(){ |
|  | List<Articulo> lista = articuloRepository.findAll(); |
|  | return lista; |
|  | } |
|  |  |
|  | public Optional<Articulo> getById(long id){ |
|  | return articuloRepository.findById(id); |
|  | } |
|  |  |
|  | public Optional<Articulo> getByNombre(String nombre){ |
|  | return articuloRepository.findByNombre(nombre); |
|  | } |
|  |  |
|  | public void save(Articulo articulo){ |
|  | articuloRepository.save(articulo); |
|  | } |
|  |  |
|  | public void delete(long id){ |
|  | articuloRepository.deleteById(id); |
|  | } |
|  |  |
|  | public boolean existsId(long id){ |
|  | return articuloRepository.existsById(id); |
|  | } |
|  |  |
|  | public boolean existsNombre(String nombre){ |
|  | return articuloRepository.existsByNombre(nombre); |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/1009316feb229f18061fc087f7f900b6/raw/6c30de3f77c7b4263af8179641a96f955133f37d/articuloservice.java)[**articuloservice.java**](https://gist.github.com/cavanosa/1009316feb229f18061fc087f7f900b6#file-articuloservice-java) hosted with  by [**GitHub**](https://github.com/)

En el paquete http la clase mensaje:

|  |  |
| --- | --- |
|  | package com.cavanosa.TutorialStripe.http; |
|  |  |
|  | public class Mensaje { |
|  |  |
|  | private String mensaje; |
|  |  |
|  | public Mensaje(String mensaje) { |
|  | this.mensaje = mensaje; |
|  | } |
|  |  |
|  | public String getMensaje() { |
|  | return mensaje; |
|  | } |
|  |  |
|  | public void setMensaje(String mensaje) { |
|  | this.mensaje = mensaje; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/b21341c6dca3634a0aedd01f689ad53b/raw/92fb4cbf48f9808f4c48da774e1429afb8adbdf4/mensaje.java)[**mensaje.java**](https://gist.github.com/cavanosa/b21341c6dca3634a0aedd01f689ad53b#file-mensaje-java) hosted with  by [**GitHub**](https://github.com/)

Por último, para terminar con el CRUD de los artículos nos queda el controlador, ArticuloController (por brevedad hemos omitido las operaciones de actualizar y eliminar)

|  |  |
| --- | --- |
|  | package com.cavanosa.TutorialStripe.controller; |
|  |  |
|  | import com.cavanosa.TutorialStripe.http.Mensaje; |
|  | import com.cavanosa.TutorialStripe.model.Articulo; |
|  | import com.cavanosa.TutorialStripe.service.ArticuloService; |
|  | import org.apache.commons.lang3.StringUtils; |
|  | import org.springframework.beans.factory.annotation.Autowired; |
|  | import org.springframework.http.HttpStatus; |
|  | import org.springframework.http.ResponseEntity; |
|  | import org.springframework.web.bind.annotation.\*; |
|  |  |
|  | import java.util.List; |
|  |  |
|  | @RestController |
|  | @RequestMapping("/articulo") |
|  | @CrossOrigin(origins = "\*") |
|  | public class ArticuloController { |
|  |  |
|  | @Autowired |
|  | ArticuloService articuloService; |
|  |  |
|  | @GetMapping("/lista") |
|  | public ResponseEntity<List<Articulo>> lista(){ |
|  | List<Articulo> lista = articuloService.lista(); |
|  | return new ResponseEntity<List<Articulo>>(lista, HttpStatus.OK); |
|  | } |
|  |  |
|  | @GetMapping("/detalle/{id}") |
|  | public ResponseEntity<Articulo> detalle(@PathVariable("id") long id){ |
|  | if(!articuloService.existsId(id)) |
|  | return new ResponseEntity(new Mensaje("no existe"), HttpStatus.NOT\_FOUND); |
|  | Articulo articulo = articuloService.getById(id).get(); |
|  | return new ResponseEntity<Articulo>(articulo, HttpStatus.OK); |
|  | } |
|  |  |
|  | @PostMapping("/nuevo") |
|  | public ResponseEntity<?> crear(@RequestBody Articulo articulo){ |
|  | if(StringUtils.isBlank(articulo.getNombre())) |
|  | return new ResponseEntity(new Mensaje("nombre obligatorio"), HttpStatus.BAD\_REQUEST); |
|  | if((Integer)articulo.getPrecio() == null || articulo.getPrecio() < 1) |
|  | return new ResponseEntity(new Mensaje("precio obligatorio"), HttpStatus.BAD\_REQUEST); |
|  | if(articuloService.existsNombre(articulo.getNombre())) |
|  | return new ResponseEntity(new Mensaje("ese nombre ya existe"), HttpStatus.BAD\_REQUEST); |
|  | articuloService.save(articulo); |
|  | return new ResponseEntity(new Mensaje("artículo creado"), HttpStatus.CREATED); |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/1fc4c2f8bf5c528499edabbeeb6ee30c/raw/353ea01d5a977d7317d03d441d96f1a5dded9f80/articulocontroller.java)[**articulocontroller.java**](https://gist.github.com/cavanosa/1fc4c2f8bf5c528499edabbeeb6ee30c#file-articulocontroller-java) hosted with  by [**GitHub**](https://github.com/)

Ejecutamos el proyecto para que JPA cree la tabla Articulo en la base de datos.

Ya para finalizar creamos uno o varios productos. Como imagenURL utilicé imágenes de productos de Amazon. Para ello buscamos un artículo cualquiera en su web, lo seleccionamos y con el ratón sobre la imagen, botón derecho, abrir imagen en una pestaña nueva. La sentencia SQL:

|  |  |
| --- | --- |
|  | INSERT INTO `articulo` (`id`, `descripcion`, `imagenurl`, `nombre`, `precio`) VALUES ('1', 'A Practical Approach to RESTful Services using RabbitMQ, Eureka, Ribbon, Zuul and Cucumber', 'https://images-na.ssl-images-amazon.com/images/I/41aqtBw0JOL.jpg', 'Learn Microservices with Spring Boot', '2079'), ('2', 'The Definitive Guide', 'https://images-eu.ssl-images-amazon.com/images/I/51gNTyekd8L.jpg', 'JavaScript', '3678'); |

[**view raw**](https://gist.github.com/cavanosa/7f1b8cd0a9d1b785560a8ff5f204bb25/raw/47576ce93c120b87839234fb7c3e52233c454fc9/articulo.sql)[**articulo.sql**](https://gist.github.com/cavanosa/7f1b8cd0a9d1b785560a8ff5f204bb25#file-articulo-sql) hosted with  by [**GitHub**](https://github.com/)

En el siguiente post implementaremos el pago con Stripe.