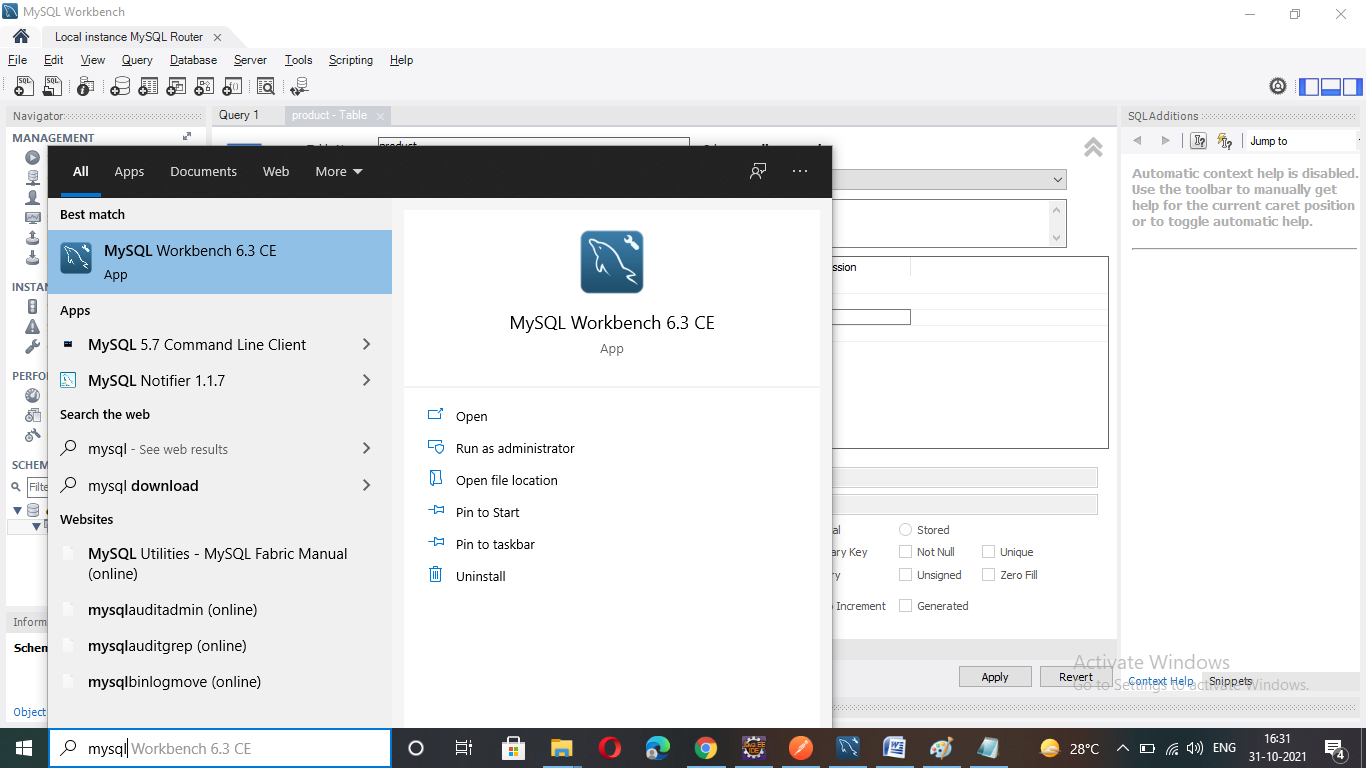
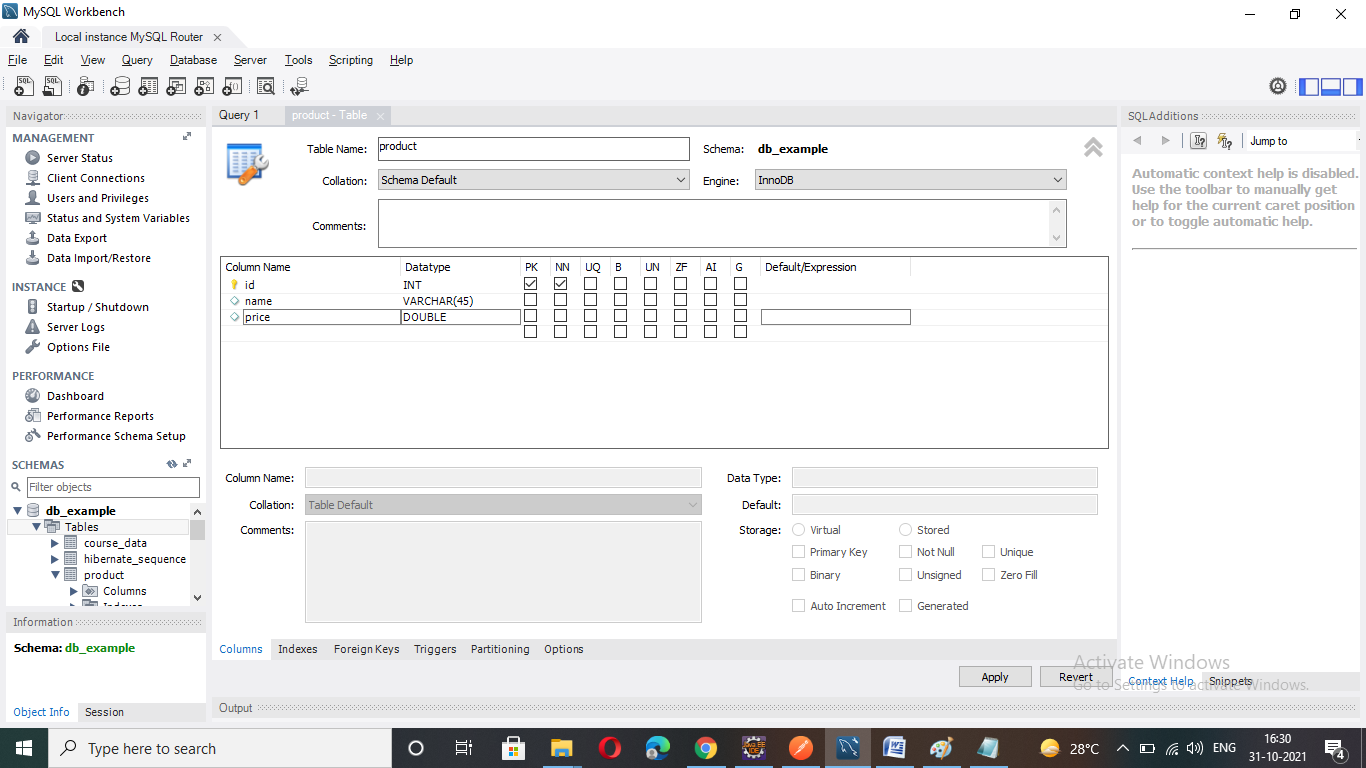
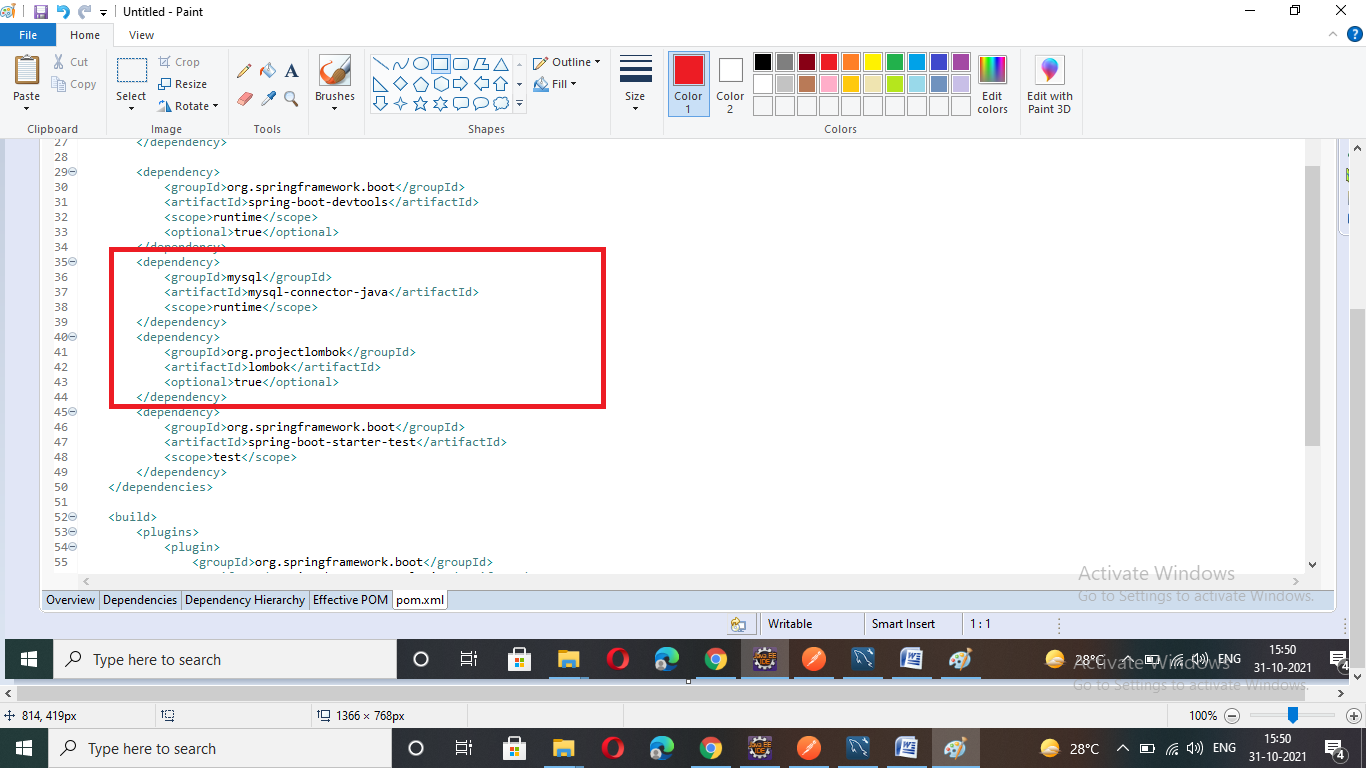
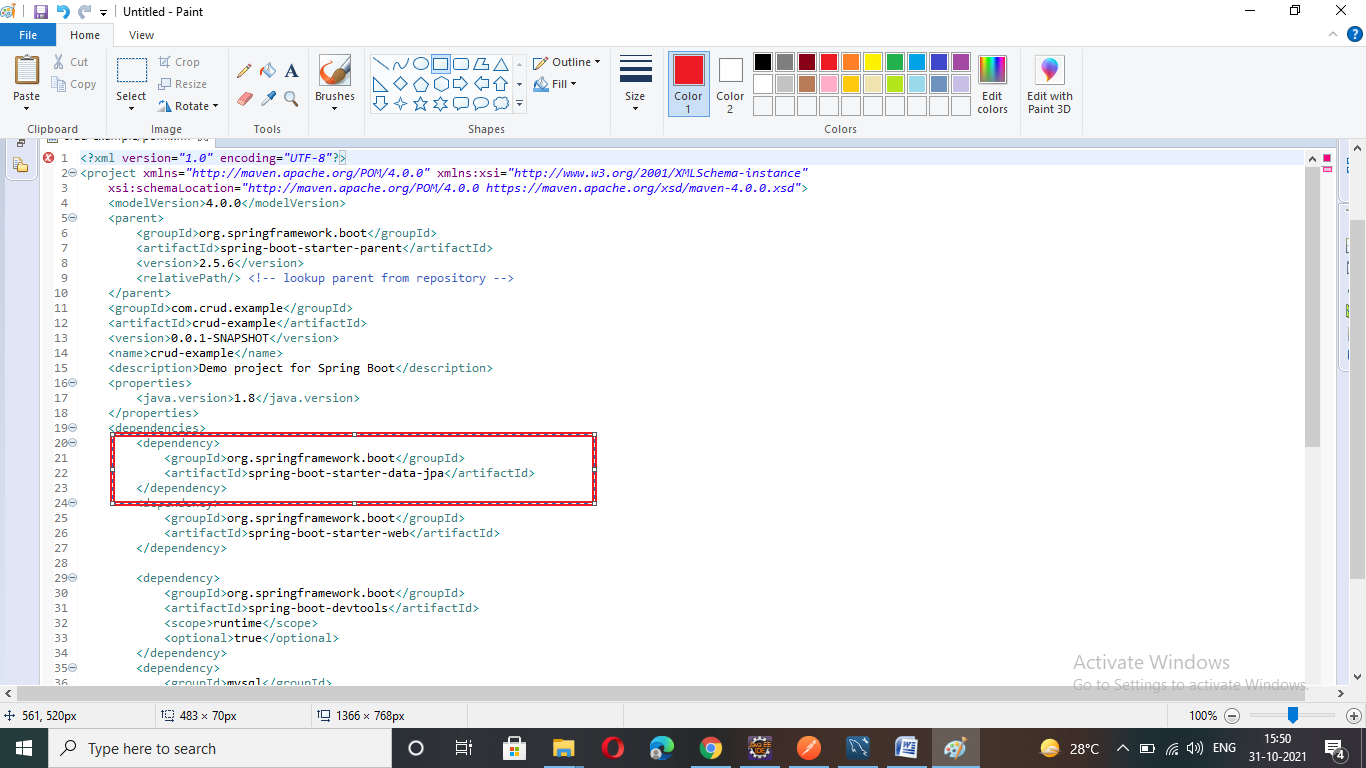
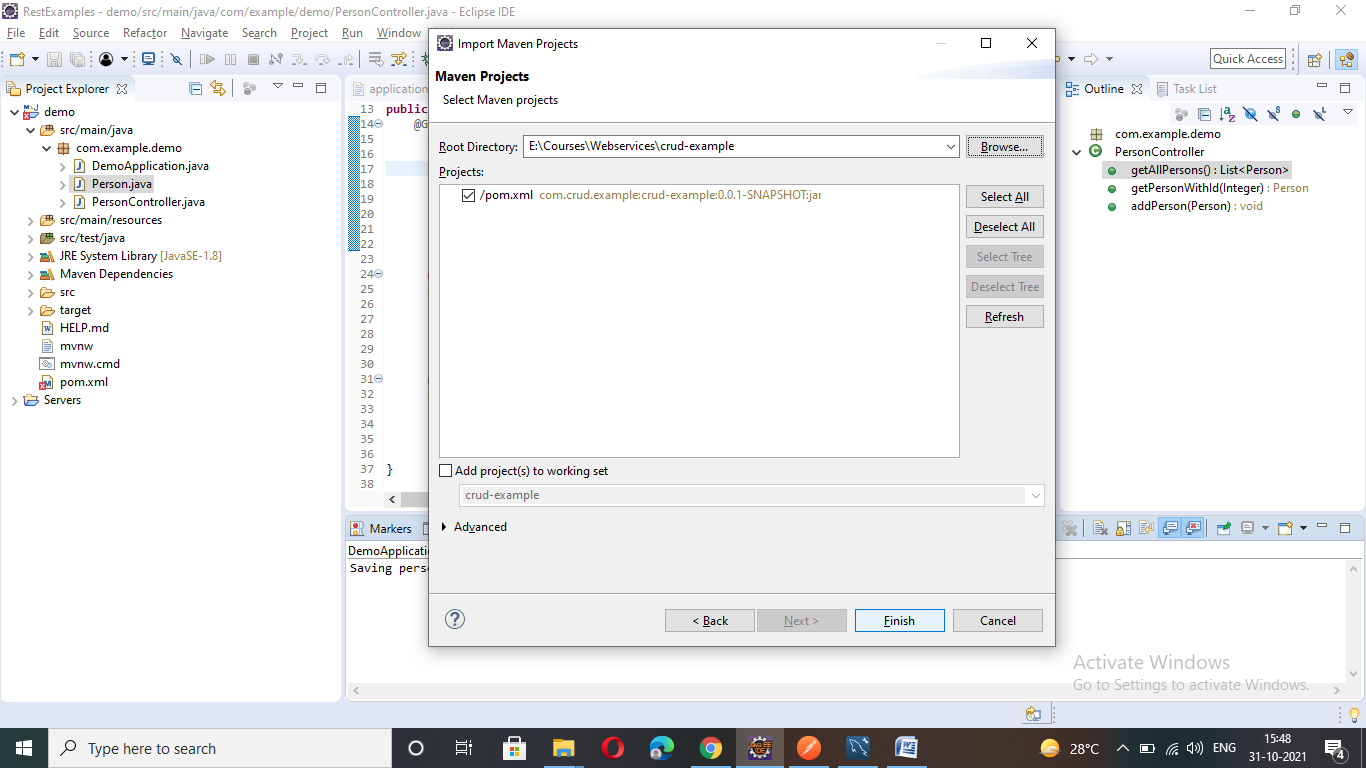
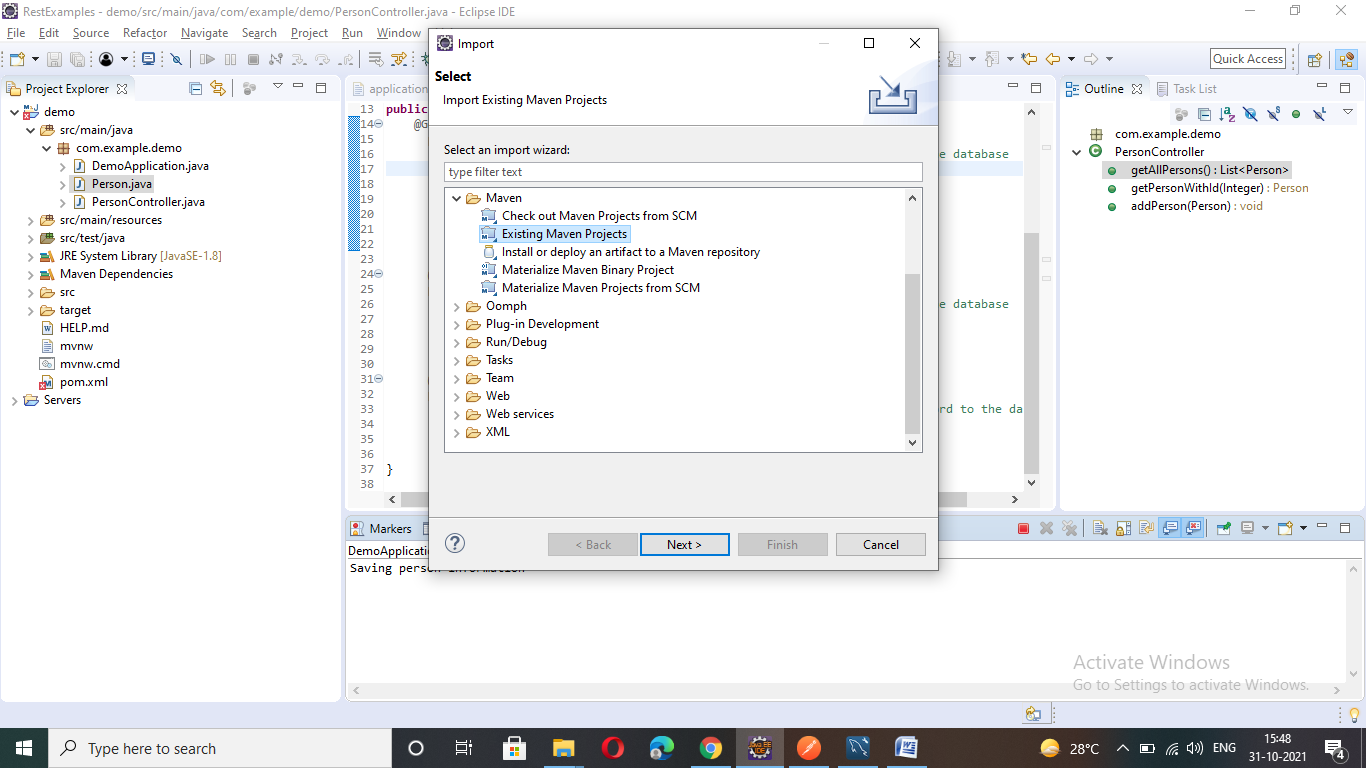
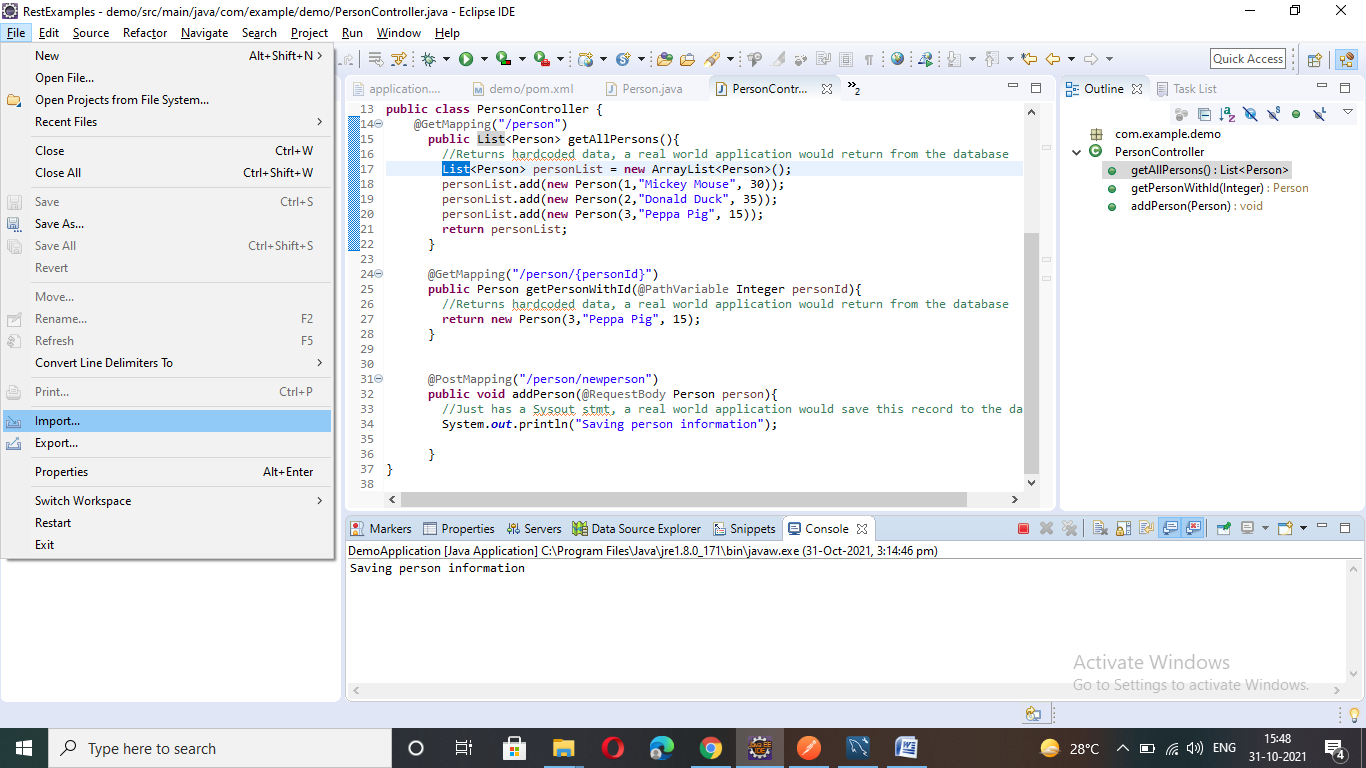


Crate a table in database







Create a POJO class

**package** com.crud.example.crud.example;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="product")

**public** **class** Product {

@Id

@GeneratedValue(strategy = GenerationType.***SEQUENCE***)

**private** Long id;

@Column(name="name")

**private** String name;

@Column(name="price")

**private** **float** price;

**protected** Product() {

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getPrice() {

**return** price;

}

**public** **void** setPrice(**float** price) {

**this**.price = price;

}

}

2 ) create a interface

**package** com.crud.example.crud.example;

**import** org.springframework.data.jpa.repository.JpaRepository;

**public** **interface** ProductRepository **extends** JpaRepository<Product, Long> {

}

3) create a service

**package** com.crud.example.crud.example;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

@Service

**public** **class** ProductService {

@Autowired

**private** ProductRepository repo;

**public** List<Product> listAll() {

**return** repo.findAll();

}

**public** **void** save(Product product) {

repo.save(product);

}

**public** Product get(**long** id) {

**return** repo.findById(id).get();

}

**public** **void** delete(**long** id) {

repo.deleteById(id);

}

}

3) controller to create a product

**package** com.crud.example.crud.example;

**import** java.util.List;

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

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

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

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

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

@RestController

**public** **class** AppController {

@Autowired

**private** ProductService service;

@PostMapping("/newproduct/save")

**public** String saveProduct(@RequestBody Product product) {

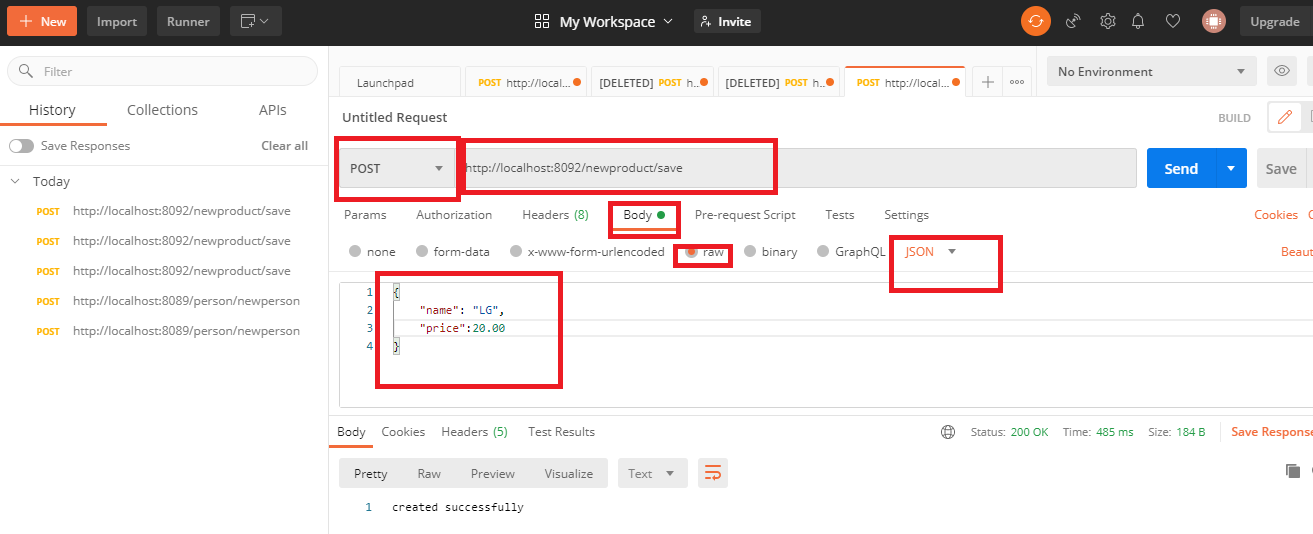
service.save(product);

**return** "created successfully";

}

}

4) open postman , trigger post reqesut



If we provide response status then it is easy to customer.

ResponseEntity Class is used to represent whole HTTP responses. It supports the  
the message body,  
headers,  
status, etc.  
Internally Response extends org.springframework.http.HttpEntity The advantage of using ResponseEntity is that it is easy to set headers status on this object directly which gets serialized as an HTTP response.

In controller classes you can add response status and newly created object details with id .SO change the logic licke below.

**package** com.crud.example.crud.example;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

@Service

**public** **class** ProductService {

@Autowired

**private** ProductRepository repo;

**public** List<Product> listAll() {

**return** repo.findAll();

}

**public** **Product** save(Product product) {

**return** repo.save(product);

}

**public** Product get(**long** id) {

**return** repo.findById(id).get();

}

**public** **void** delete(**long** id) {

repo.deleteById(id);

}

}

Controller:

**package** com.crud.example.crud.example;

**import** java.util.List;

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

**import** org.springframework.http.ResponseEntity;

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

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

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

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

@RestController

**public** **class** AppController {

@Autowired

**private** ProductService service;

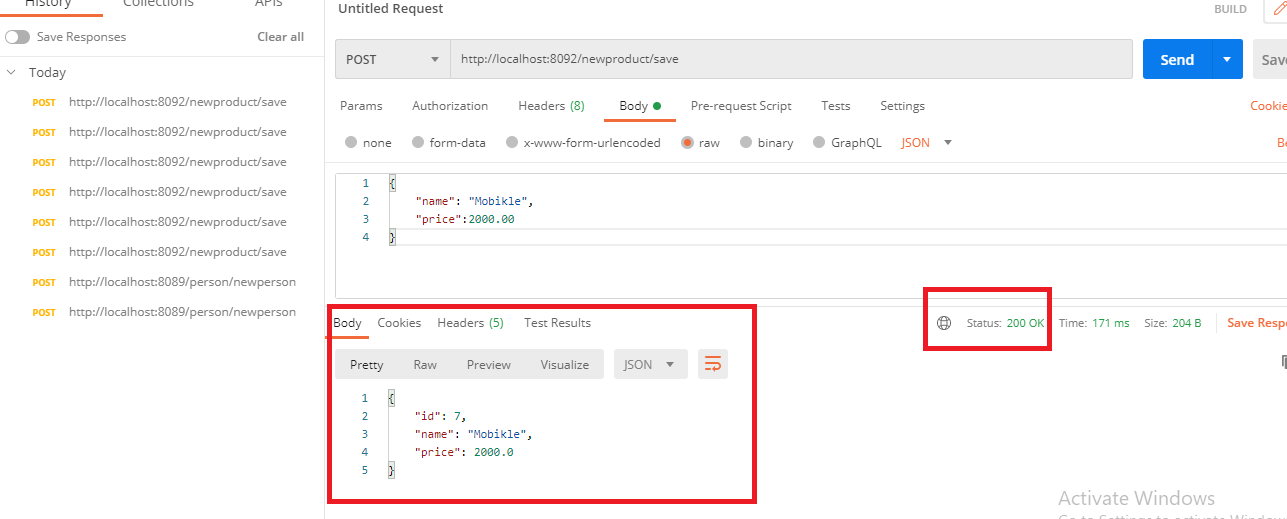
@PostMapping("/newproduct/save")

**public** ResponseEntity<Product> saveProduct(@RequestBody Product product) {

**return** ResponseEntity.*ok*().body(**this**.service.save(product));

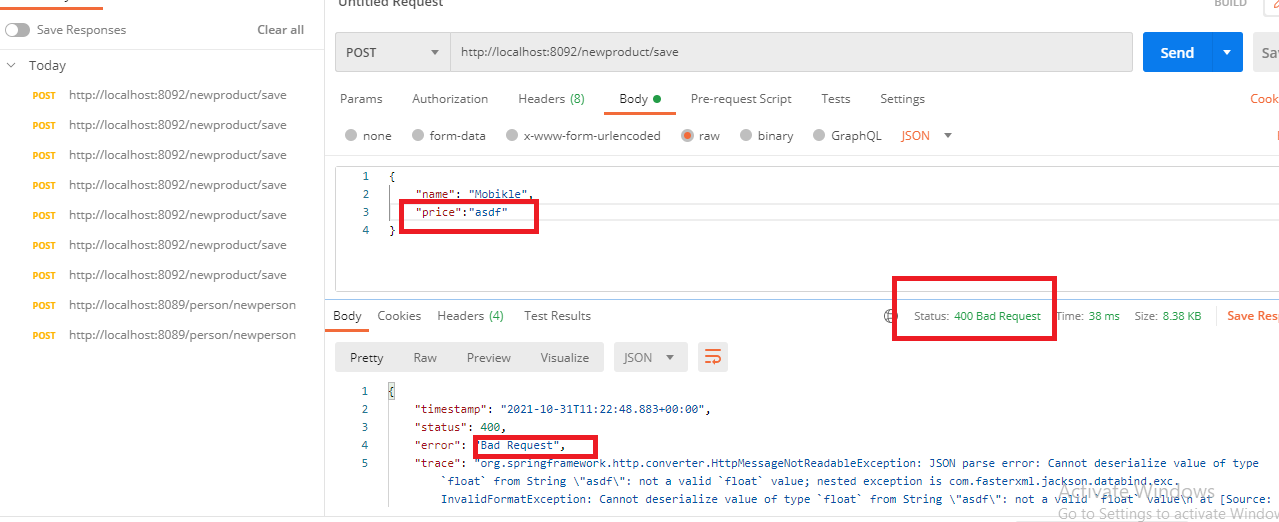
}

}



Status is ok and return newly created id 7 and details.

If we provide wrong data

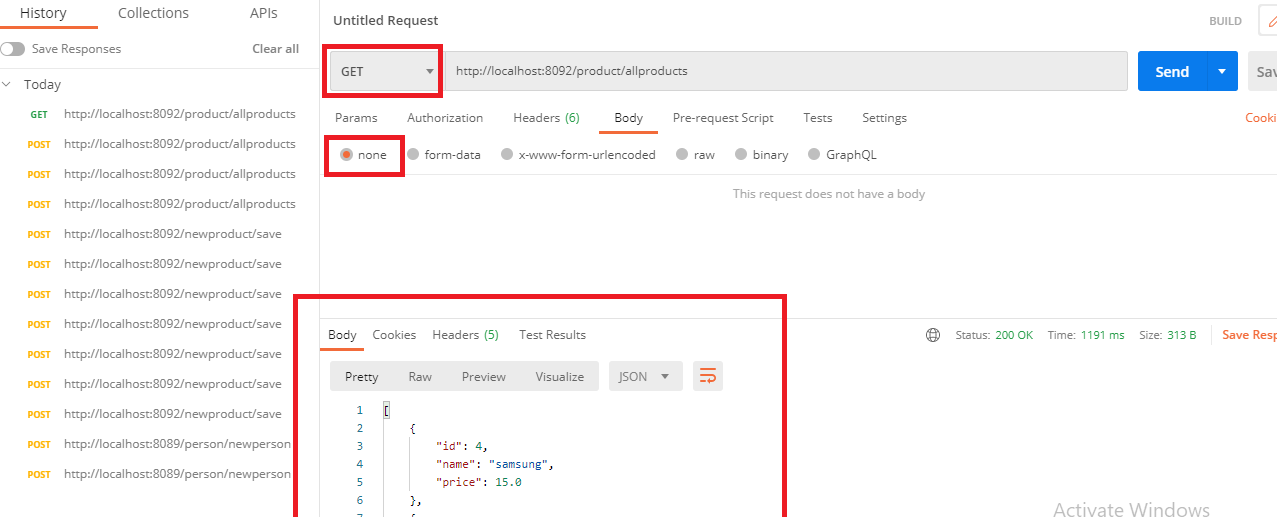


200 status is means ->ok

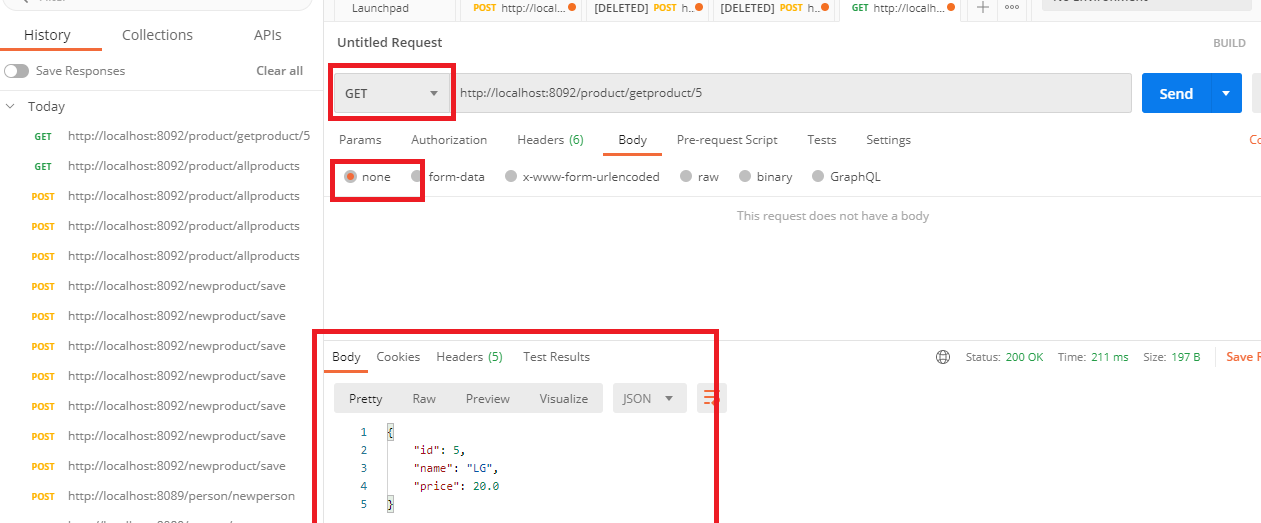
400 status is means🡪Bad request

2) Now add list and update , delete

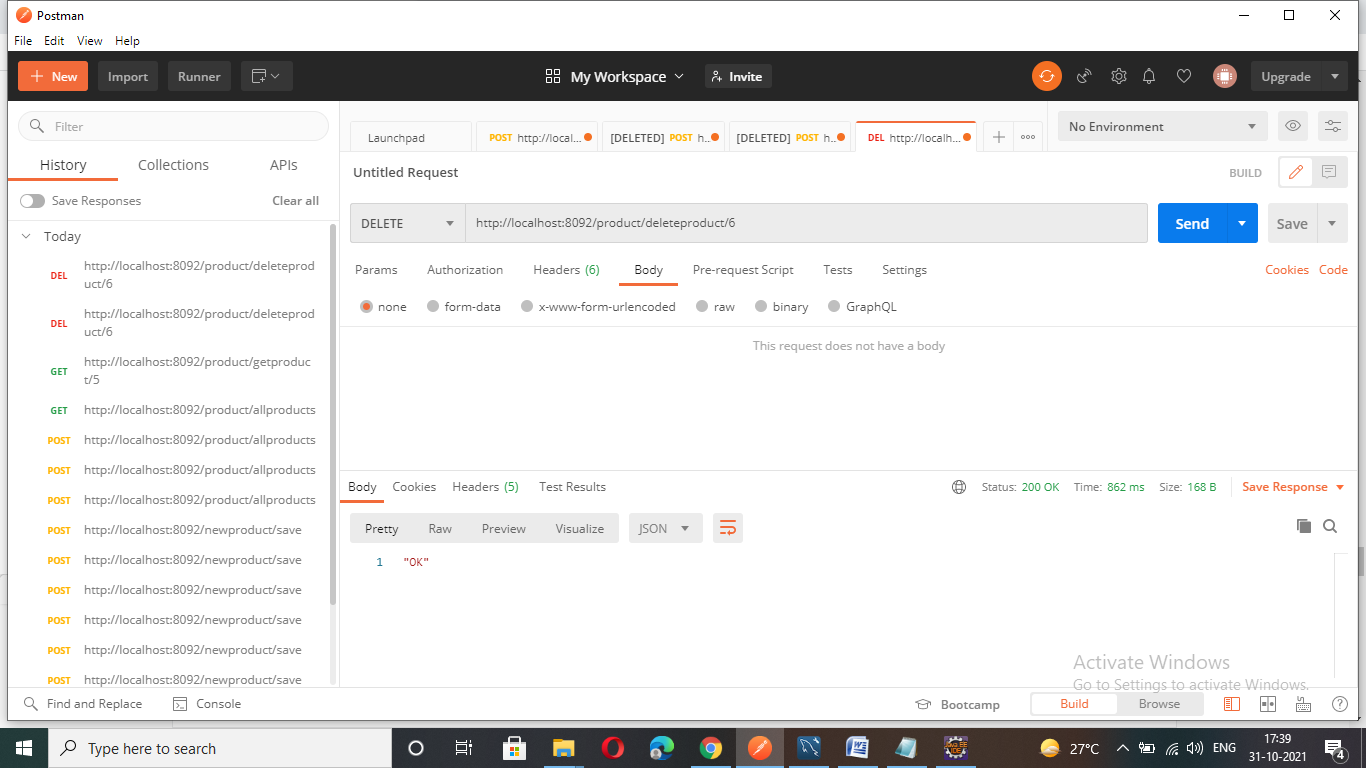
List:



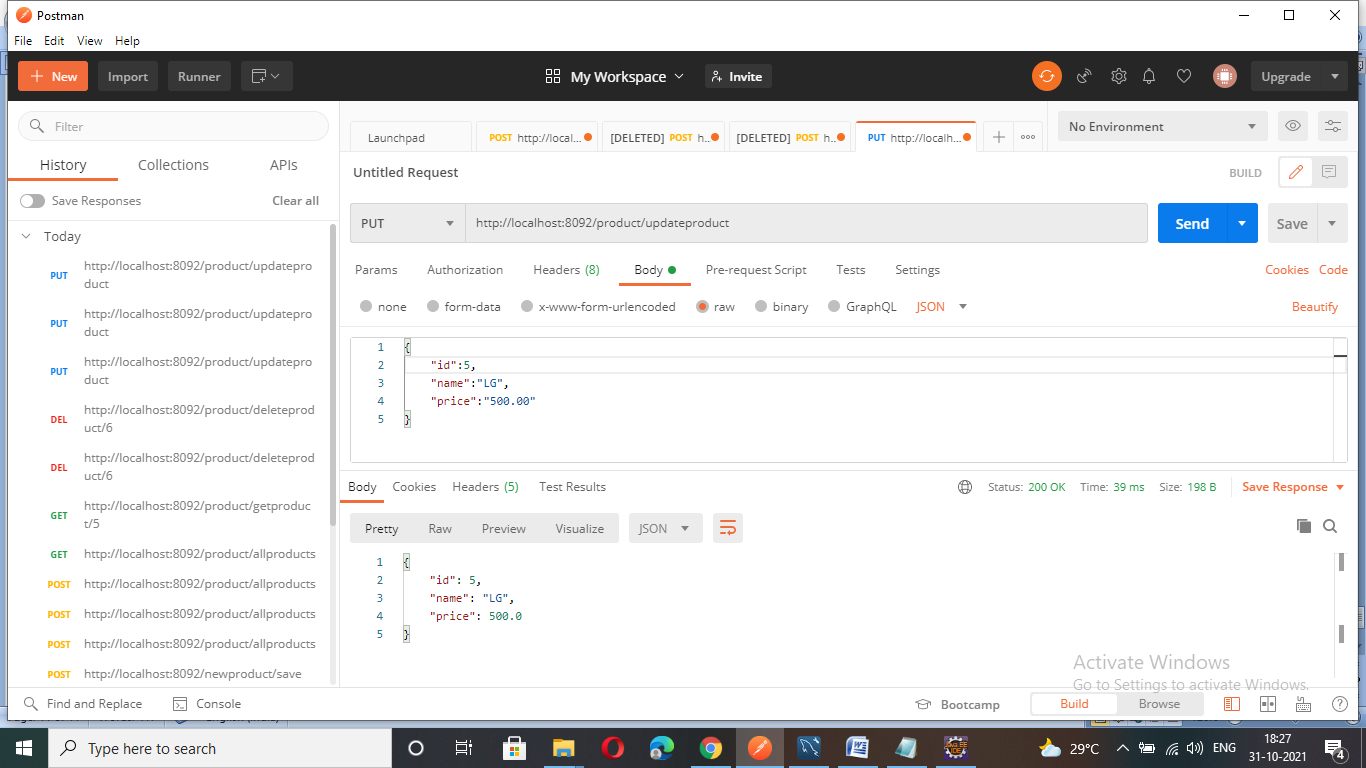
Edit:



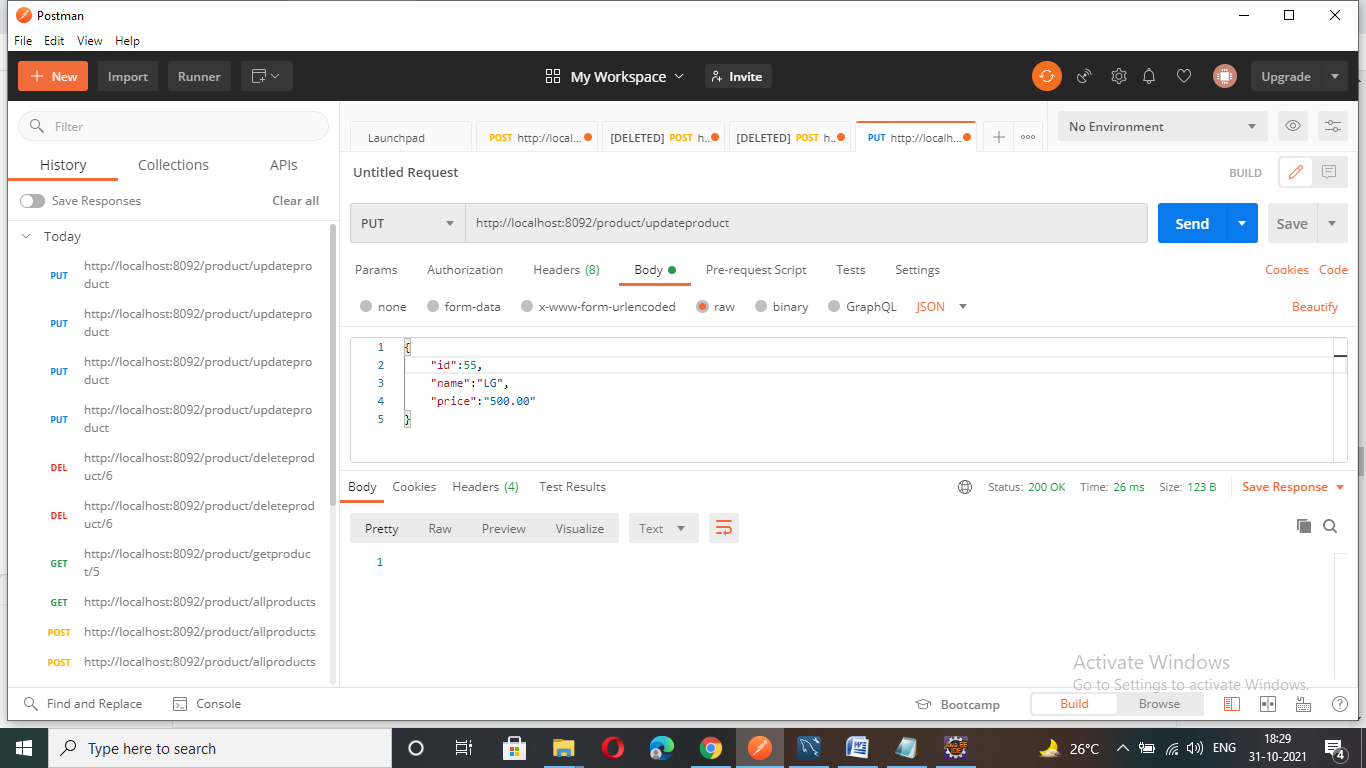
Delete :

****

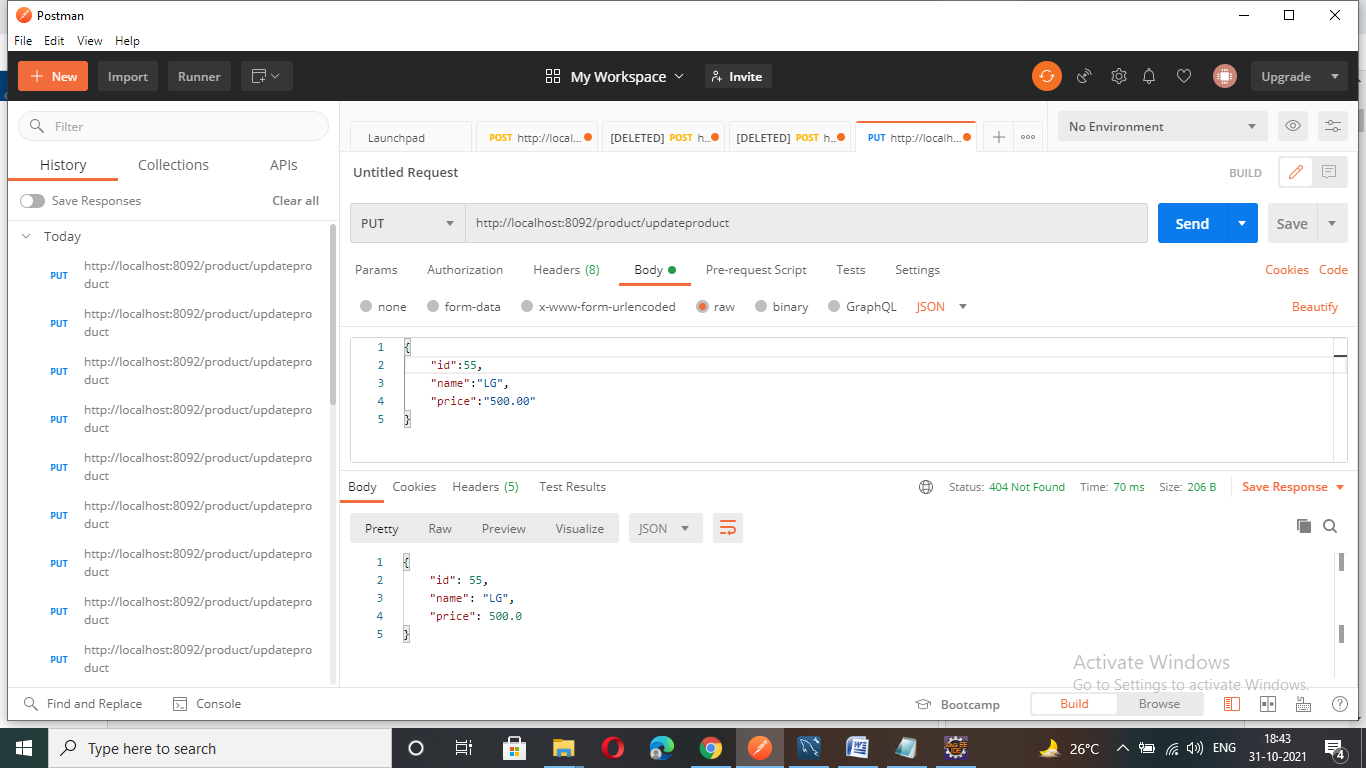
**Update:**

****

If i give wrong id : No error message display. To avoid that issue , write below logic.



Status code error :



In code :

@PutMapping("/product/updateproduct")

**public** ResponseEntity<Product> updateProductById(@RequestBody Product product) {

**try** {

**return** ResponseEntity.*ok*().body(**this**.service.update(product));

} **catch** (RelationNotFoundException e) {

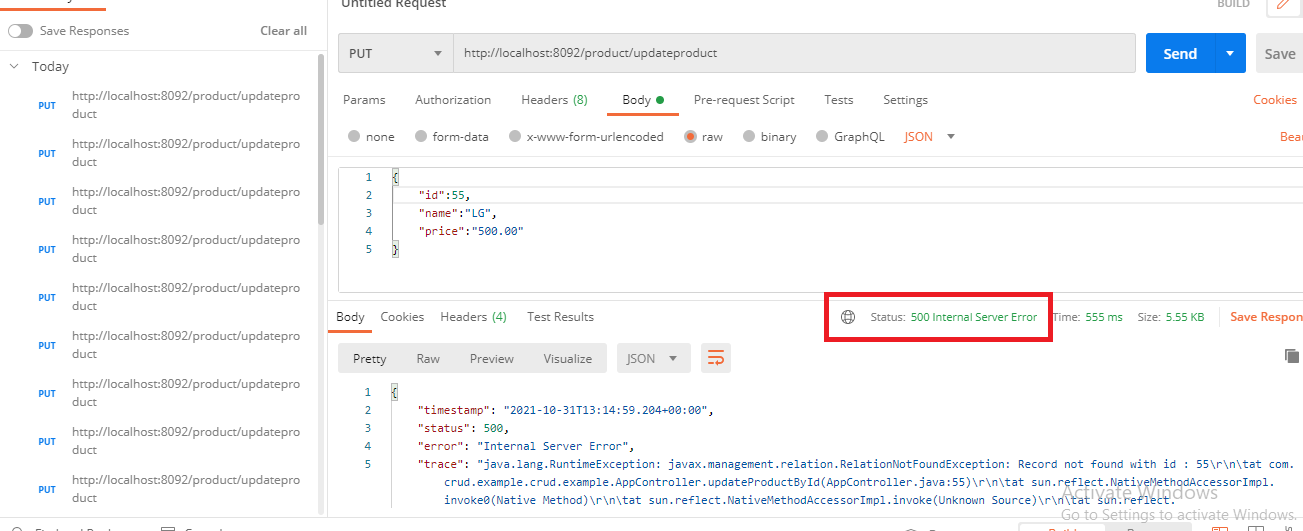
**return** ResponseEntity

.*status*(HttpStatus.***NOT\_FOUND***)

.body(product);

}

Other approach :



Code :

@PutMapping("/product/updateproduct")

**public** ResponseEntity<Product> updateProductById(@RequestBody Product product) {

**try** {

**return** ResponseEntity.*ok*().body(**this**.service.update(product));

} **catch** (RelationNotFoundException e) {

**throw** **new** RuntimeException(e);

}

}

Finale code :

**package** com.crud.example.crud.example;

**import** java.util.List;

**import** javax.management.relation.RelationNotFoundException;

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

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

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

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

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

**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

**public** **class** AppController {

@Autowired

**private** ProductService service;

@PostMapping("/newproduct/save")

**public** ResponseEntity<Product> saveProduct(@RequestBody Product product) {

**return** ResponseEntity.*ok*().body(**this**.service.save(product));

}

@GetMapping("/product/allproducts")

**public** ResponseEntity<List<Product>> getAllProducts() {

**return** ResponseEntity.*ok*().body(**this**.service.listAll());

}

@GetMapping("/product/getproduct/{id}")

**public** ResponseEntity<Product> getProductById(@PathVariable **long** id) {

**return** ResponseEntity.*ok*().body(**this**.service.get(id));

}

@DeleteMapping("/product/deleteproduct/{id}")

**public** HttpStatus deleteProductById(@PathVariable **long** id) {

**this**.service.delete(id);

**return** HttpStatus.***OK***;

}

@PutMapping("/product/updateproduct")

**public** ResponseEntity<Product> updateProductById(@RequestBody Product product) {

**try** {

**return** ResponseEntity.*ok*().body(**this**.service.update(product));

} **catch** (RelationNotFoundException e) {

**throw** **new** RuntimeException(e);

}

}

}

Service :

**package** com.crud.example.crud.example;

**import** java.util.List;

**import** java.util.Optional;

**import** javax.management.relation.RelationNotFoundException;

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

**import** org.springframework.stereotype.Service;

@Service

**public** **class** ProductService {

@Autowired

**private** ProductRepository repo;

**public** List<Product> listAll() {

**return** repo.findAll();

}

**public** Product save(Product product) {

**return** repo.save(product);

}

**public** Product get(**long** id) {

**return** repo.findById(id).get();

}

**public** **void** delete(**long** id) {

repo.deleteById(id);

}

**public** Product update(Product product) **throws** RelationNotFoundException {

Optional < Product > productDb = repo.findById(product.getId());

**if** (productDb.isPresent()) {

Product productUpdate = productDb.get();

productUpdate.setId(product.getId());

productUpdate.setName(product.getName());

productUpdate.setPrice(product.getPrice());

repo.save(productUpdate);

**return** productUpdate;

} **else** {

**throw** **new** RelationNotFoundException("Record not found with id : " + product.getId());

}

}

}

Exception class:

**package** com.crud.example.crud.example;

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

@ResponseStatus

**public** **class** ResourceNotFoundException **extends** RuntimeException {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** ResourceNotFoundException(String message) {

**super**(message);

}

**public** ResourceNotFoundException(String message, Throwable throwable) {

**super**(message, throwable);

}

}

Only few methods allow by JPA :

Save,getById,delete,update allow .

If we want to fetch by name JPA not allow.

Write User defined methods:

Controller:

@GetMapping("/product/getproductByName/{name}")

**public** ResponseEntity<List<Product>> getProductByName(@PathVariable String name) {

**return** ResponseEntity.*ok*().body(**this**.service.getByName(name));

}

Service:

**public** List<Product> getByName(String name) {

**return** repo.findByName(name);

}

Interface:

**package** com.crud.example.crud.example;

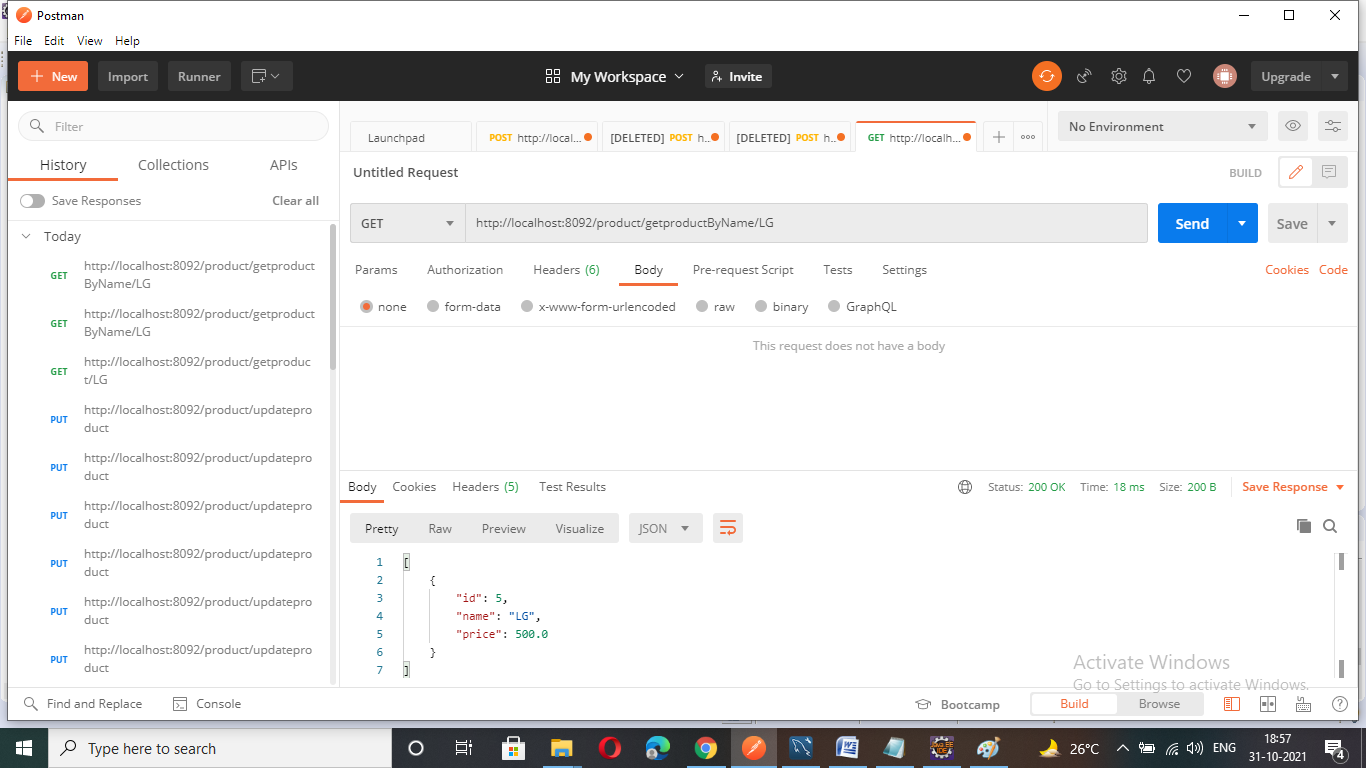
**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

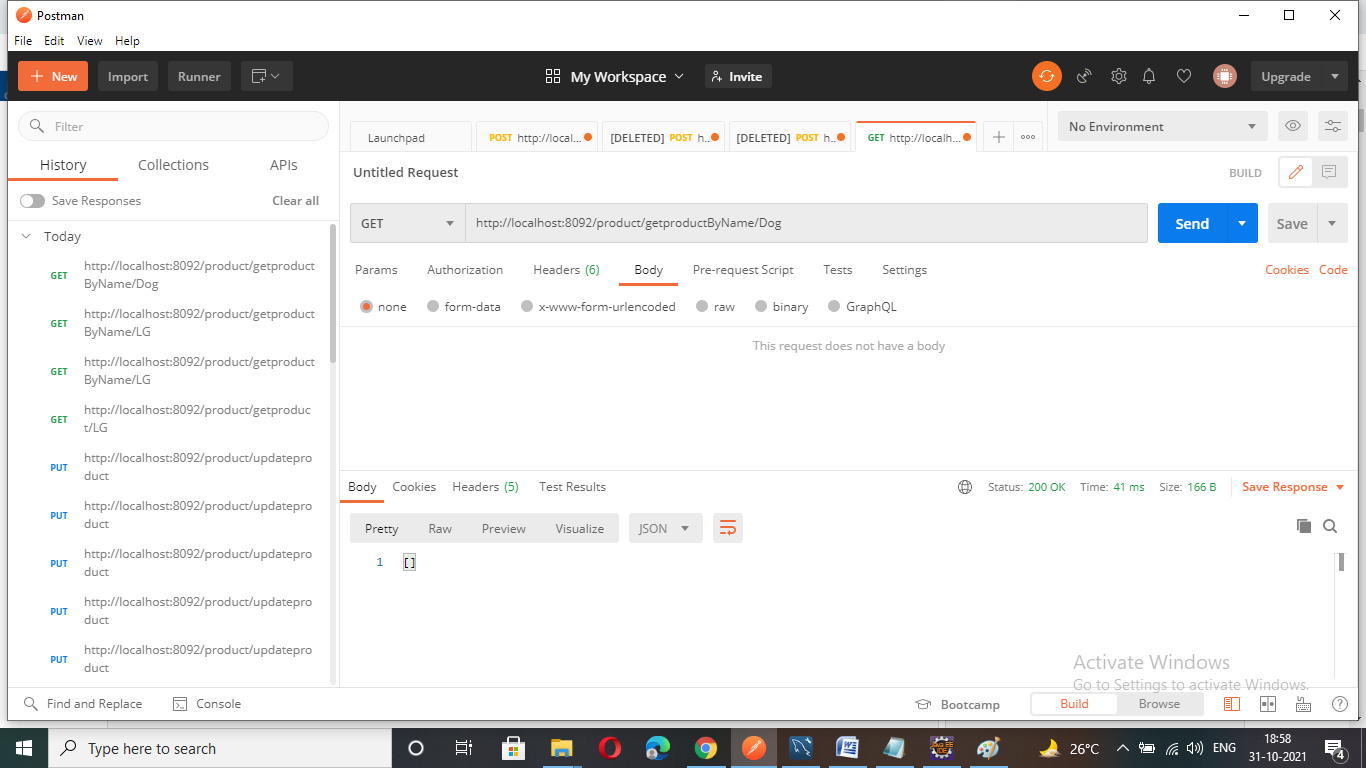
**public** **interface** ProductRepository **extends** JpaRepository<Product, Long> {

**public** List<Product> findByName(String name);

}



If we pass the wrong name , http status showing ok , it is not correct . we are showing error message .



Controller:

@GetMapping("/product/getproductByName/{name}")

**public** ResponseEntity<List<Product>> getProductByName(@PathVariable String name) **throws** RelationNotFoundException {

List<Product> list = **this**.service.getByName(name);

**if**(list.size()!=0) {

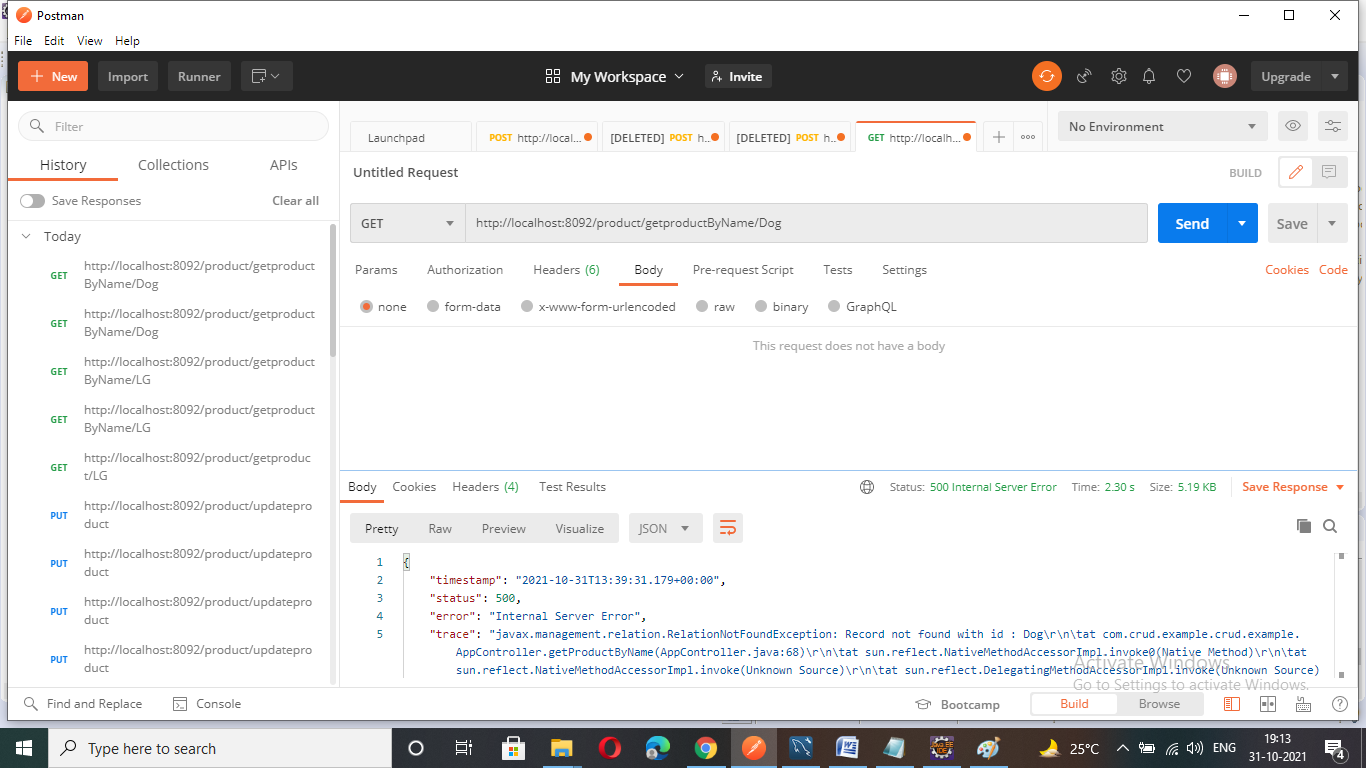
**return** ResponseEntity.*ok*().body(**this**.service.getByName(name));

}**else** {

**throw** **new** RelationNotFoundException("Record not found with id : " + name);

}

}



Final classes :

**package** com.crud.example.crud.example;

**import** java.util.List;

**import** javax.management.relation.RelationNotFoundException;

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

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

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

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

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

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

**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

**public** **class** AppController {

@Autowired

**private** ProductService service;

@PostMapping("/newproduct/save")

**public** ResponseEntity<Product> saveProduct(@RequestBody Product product) {

**return** ResponseEntity.*ok*().body(**this**.service.save(product));

}

@GetMapping("/product/allproducts")

**public** ResponseEntity<List<Product>> getAllProducts() {

**return** ResponseEntity.*ok*().body(**this**.service.listAll());

}

@GetMapping("/product/getproduct/{id}")

**public** ResponseEntity<Product> getProductById(@PathVariable **long** id) {

**return** ResponseEntity.*ok*().body(**this**.service.get(id));

}

@DeleteMapping("/product/deleteproduct/{id}")

**public** HttpStatus deleteProductById(@PathVariable **long** id) {

**this**.service.delete(id);

**return** HttpStatus.***OK***;

}

@PutMapping("/product/updateproduct")

**public** ResponseEntity<Product> updateProductById(@RequestBody Product product) {

**try** {

**return** ResponseEntity.*ok*().body(**this**.service.update(product));

} **catch** (RelationNotFoundException e) {

**throw** **new** RuntimeException(e);

}

}

@GetMapping("/product/getproductByName/{name}")

**public** ResponseEntity<List<Product>> getProductByName(@PathVariable String name) **throws** RelationNotFoundException {

List<Product> list = **this**.service.getByName(name);

**if**(list.size()!=0) {

**return** ResponseEntity.*ok*().body(**this**.service.getByName(name));

}**else** {

**throw** **new** RelationNotFoundException("Record not found with id : " + name);

}

}

}

Serive :

**package** com.crud.example.crud.example;

**import** java.util.List;

**import** java.util.Optional;

**import** javax.management.relation.RelationNotFoundException;

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

**import** org.springframework.stereotype.Service;

@Service

**public** **class** ProductService {

@Autowired

**private** ProductRepository repo;

**public** List<Product> listAll() {

**return** repo.findAll();

}

**public** Product save(Product product) {

**return** repo.save(product);

}

**public** Product get(**long** id) {

**return** repo.findById(id).get();

}

**public** List<Product> getByName(String name) {

**return** repo.findByName(name);

}

**public** **void** delete(**long** id) {

repo.deleteById(id);

}

**public** Product update(Product product) **throws** RelationNotFoundException {

Optional < Product > productDb = repo.findById(product.getId());

**if** (productDb.isPresent()) {

Product productUpdate = productDb.get();

productUpdate.setId(product.getId());

productUpdate.setName(product.getName());

productUpdate.setPrice(product.getPrice());

repo.save(productUpdate);

**return** productUpdate;

} **else** {

**throw** **new** RelationNotFoundException("Record not found with id : " + product.getId());

}

}

}

Interface:

**package** com.crud.example.crud.example;

**import** java.util.List;

**import** org.springframework.data.jpa.repository.JpaRepository;

**public** **interface** ProductRepository **extends** JpaRepository<Product, Long> {

**public** List<Product> findByName(String name);

}

**package** com.crud.example.crud.example;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name="product")

**public** **class** Product {

@Id

@GeneratedValue(strategy = GenerationType.***SEQUENCE***)

**private** Long id;

@Column(name="name")

**private** String name;

@Column(name="price")

**private** **float** price;

**protected** Product() {

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getPrice() {

**return** price;

}

**public** **void** setPrice(**float** price) {

**this**.price = price;

}

}

server.port=8092

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/db\_example

spring.datasource.username=springuser

spring.datasource.password=ThePassword

spring.datasource.testWHileIdle=true

spring.datasource.validationQuery=SELECT 1

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.hibernate.naming-strategy=org.hibernate.cfg.ImprovedNamingStrategy

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

spring.main.allow-bean-definition-overriding=true