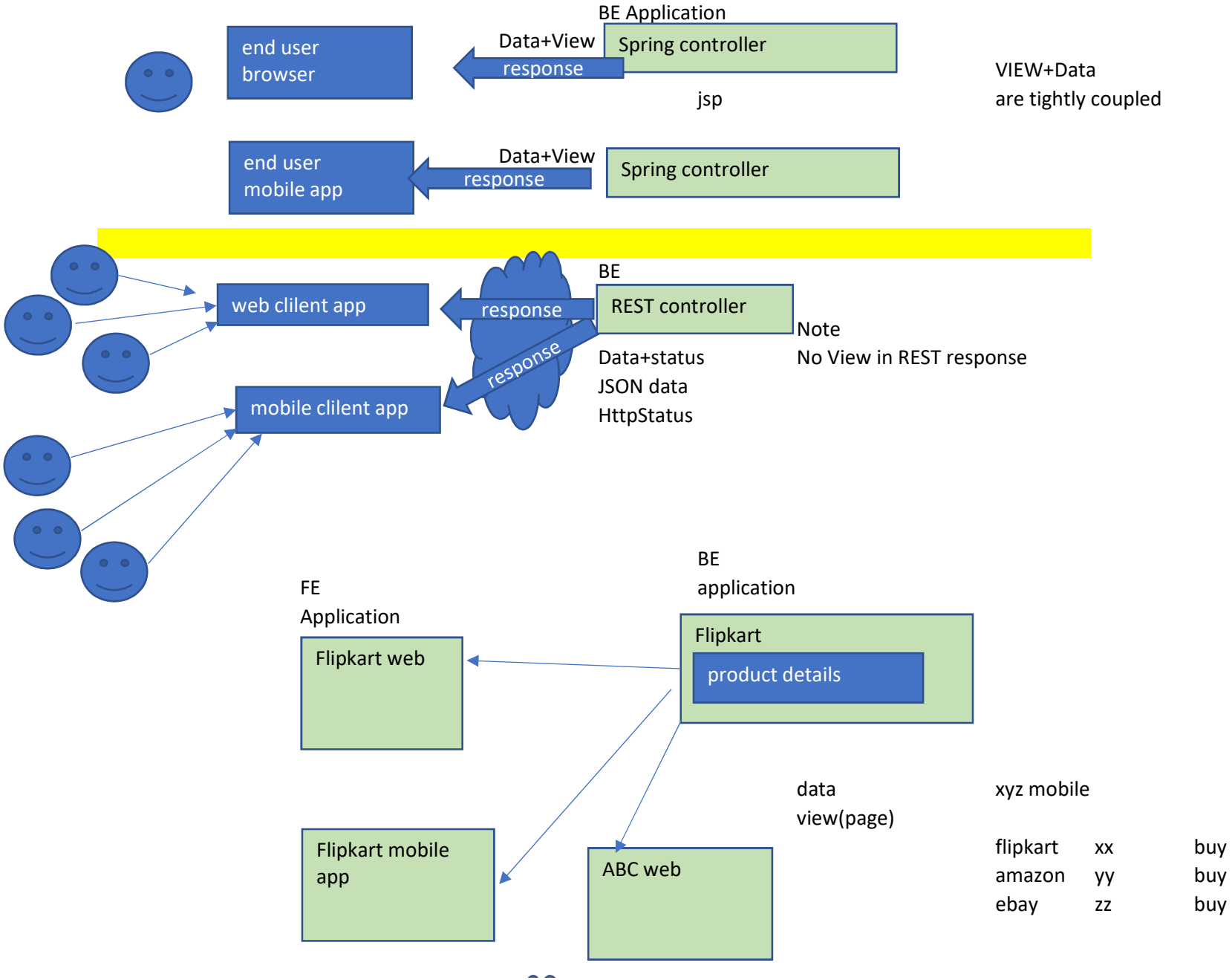
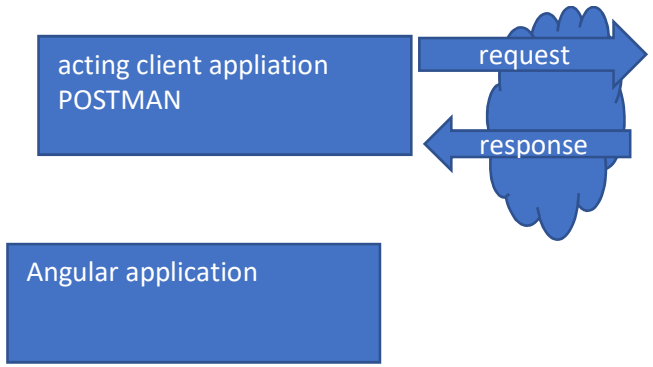

BE Application
end user browser
Data+View response
Spring controller
jsp
VIEW+Data are tightly coupled
end user mobile app
Data+View response
Spring controller
web clilent app
mobile clilent app
response
response
BE
REST controller
Data+status
JSON data
HttpStatus
Note
No View in REST response
FE Application
Flipkart web
Flipkart mobile app
ABC web
BE application
Flipkart
product details
data
view(page)
xyz mobile
flipkart xx buy
amazon yy buy
ebay zz buy



Spring controller

```
@Controller
public class HomeController {
	@RequestMapping("/data")
	public String displayHome(Model m){

		m.addAttribute("data",1234);
		return "hompage";
	}
}
```

Above request handler returns view with data

CrudReposiory

JPARepository

MongoRepository

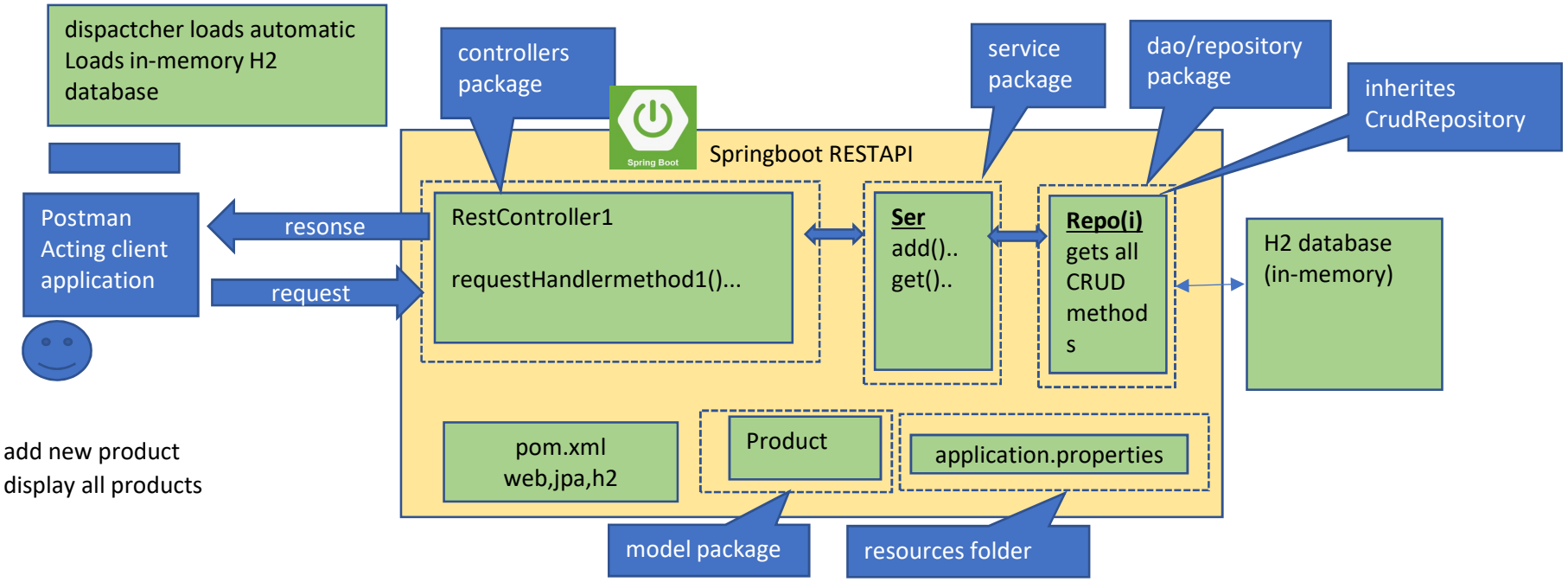
RestController

```
class MyTime{
}
```

```
@RestController
public class TimeController{
	@GetMapping("/time")
	public ResponseEntity<?> getTime(){
		MyTime t = new MyTime(1,2,3);
		return new ResponseEntity<>(t,HttpStatus.OK);
	}
}
```

ResponseEntity holds data + status

Demo
SpringBoot application
RESTful API



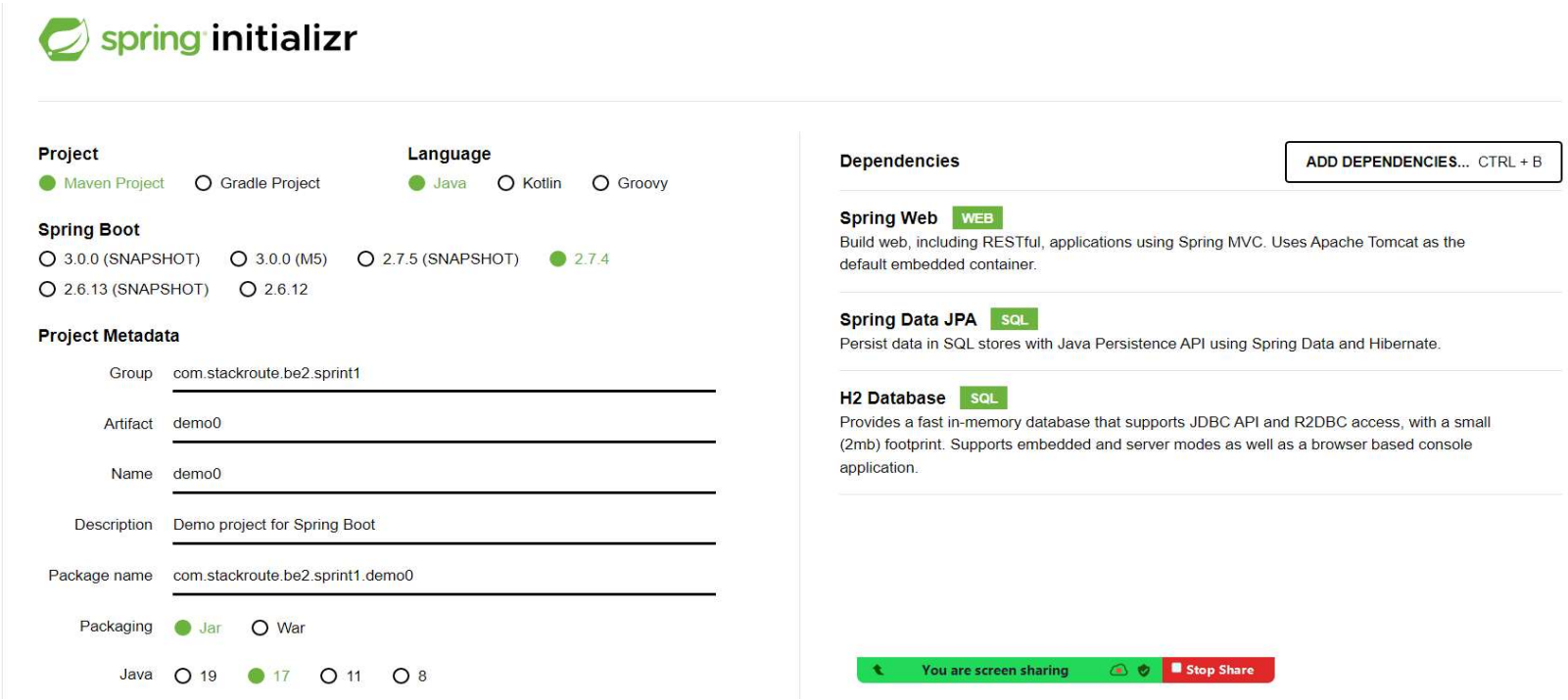
add new product
display all products

Step 1 Create springboot application by adding required dependencies

web

data

h2



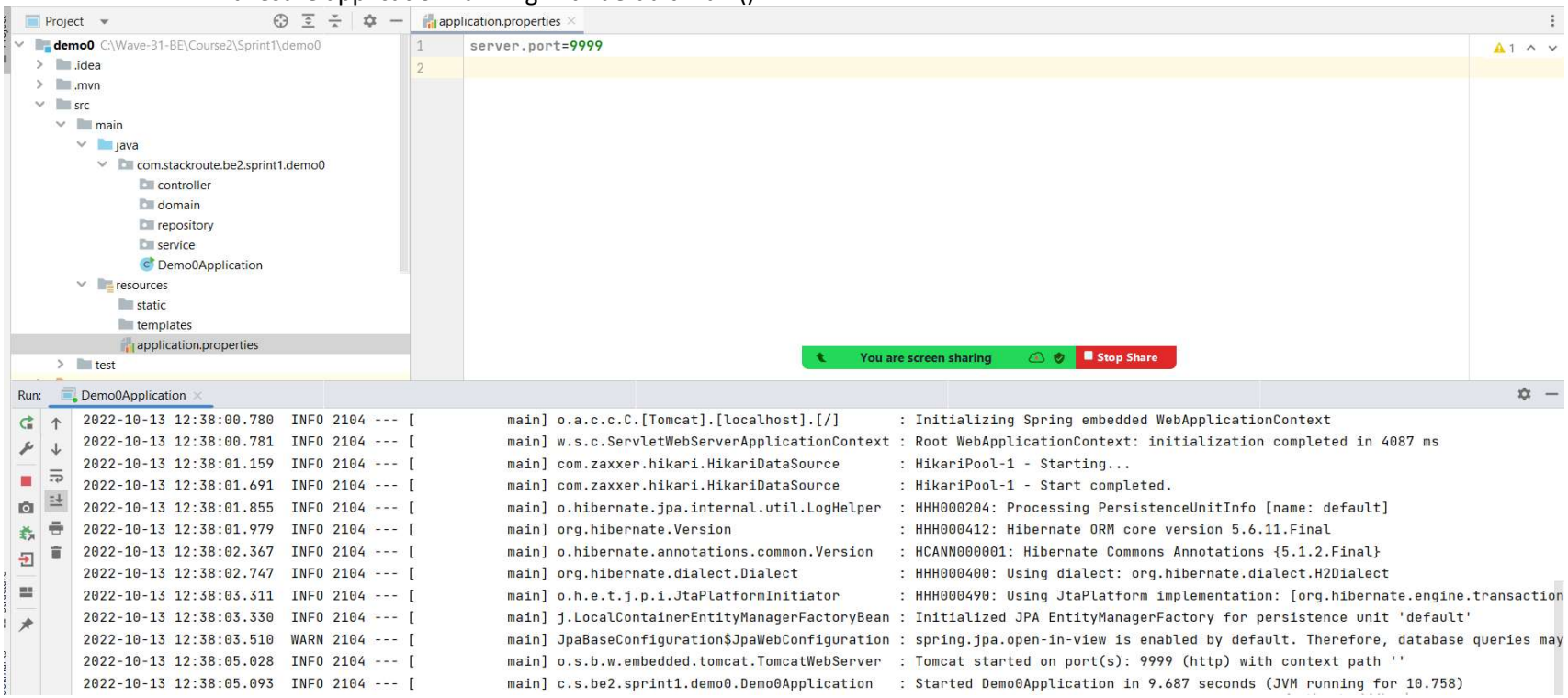
download the created project

Step 2 Extract downloaded project

open in intellij

create required packages

Makesure application running with default main()



Step 3 Define model class

```
6   @Entity
7   public class Product {
        4 usages
8       @Id
9       private String productId;
        4 usages
10      private String category,name;
        4 usages
11      private double price, rating;
12
13      public Product() {
```

Step 4 Create repository interface by providing domain entity and identifier type in domain entity

model class, primary key datatype in model class

model class/domain entity Product

identifier type in domain entity String

```java
package com.stackroute.be2.sprint1.demo0.repository;

import com.stackroute.be2.sprint1.demo0.domain.Product;
import org.springframework.data.repository.CrudRepository;

// @Repository (not required)
public interface ProductRepository extends CrudRepository <Product, String>{
        /*
            findAll()
            findById()
            save();
            delete();
            ..
            ..
            .
            .
         */
}

// Spring IOC creates an IMPL class in spring container for above
// repository with all definitions for abstract methods
/*
class ProductRepositoryImpl implements ProductRepository{

findAll() {  ... get all products ... }
save(product) {  .. to save product... }
}
 */
```

insert/save/add()

return object

Step 5 Define service layer with two methods

addProduct()

getProducts()

declare interface

define impl class

```java
public interface ProductService {
    public abstract Product addProduct(Product product);
    public abstract List<Product> getProducts();
}
```

```java
@Service
public class ProductServiceImpl implements ProductService {

    2 usages
    @Autowired
    private ProductRepository productRepository;

    @Override
    public Product addProduct(Product product) {
        return productRepository.save(product);
    }

    @Override
    public List<Product> getProducts() {
        return (List<Product>)productRepository.findAll();
    }
}
```

Step 6 Define controller

```java
@RequestMapping("/productapp/v1")
@RestController
public class ProductController {
    1 usage
    @Autowired
    private ProductService productService;

    @GetMapping("/product")
    public ResponseEntity<?> getProductdata(){
        // return RE object with productdata+status
        List<Product> products = productService.getProducts();
        return new ResponseEntity<>(products, HttpStatus.OK);
    }
}
```

Step 7 enable inmemory h2

application.properties

spring.h2.console.enabled=true

```
server.port=9999
spring.h2.console.enabled=true
```

How to login to in memory h2 database

```
main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 3175 ms
main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
main] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:bb650b1c-9eae-4cac-bb0e-05afae7baa57'
main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.6.11.Final
main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.
main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may
```

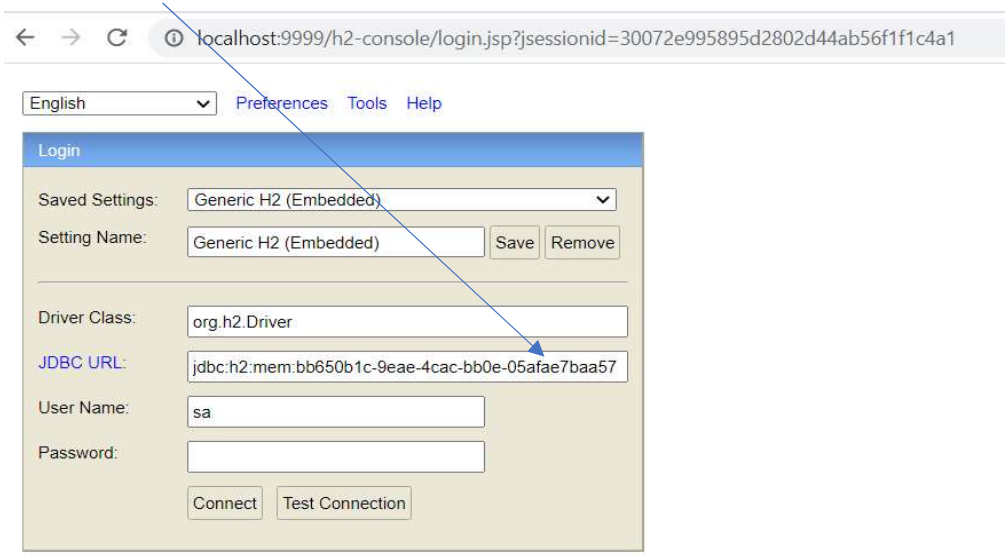
changes for every execution

if port number is xxxx

then H2 URL is

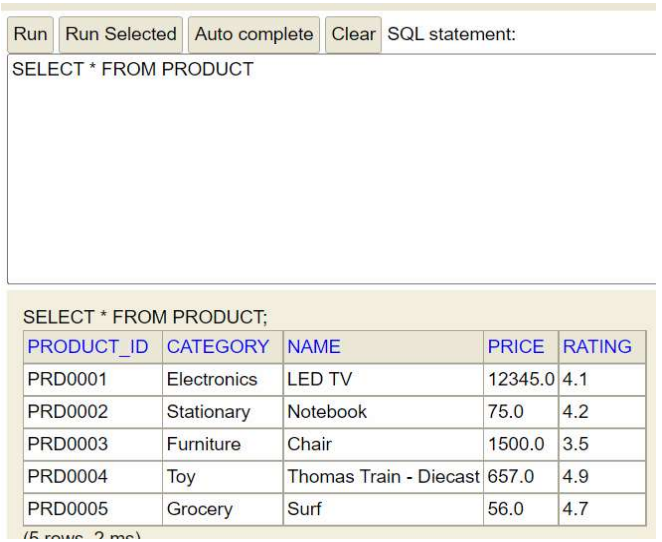
http://localhost:xxxx/h2-console

JDBC URL



insert few records

```
insert into product values('PRD0001','Electronics','LED TV',12345,4.1);
insert into product values('PRD0002','Stationary','Notebook',75,4.2);
insert into product values('PRD0003','Furniture','Chair',1500,3.5);
insert into product values('PRD0004','Toy','Thomas Train - Diecast',657,4.9);
insert into product values('PRD0005','Grocery','Surf',56,4.7);
```



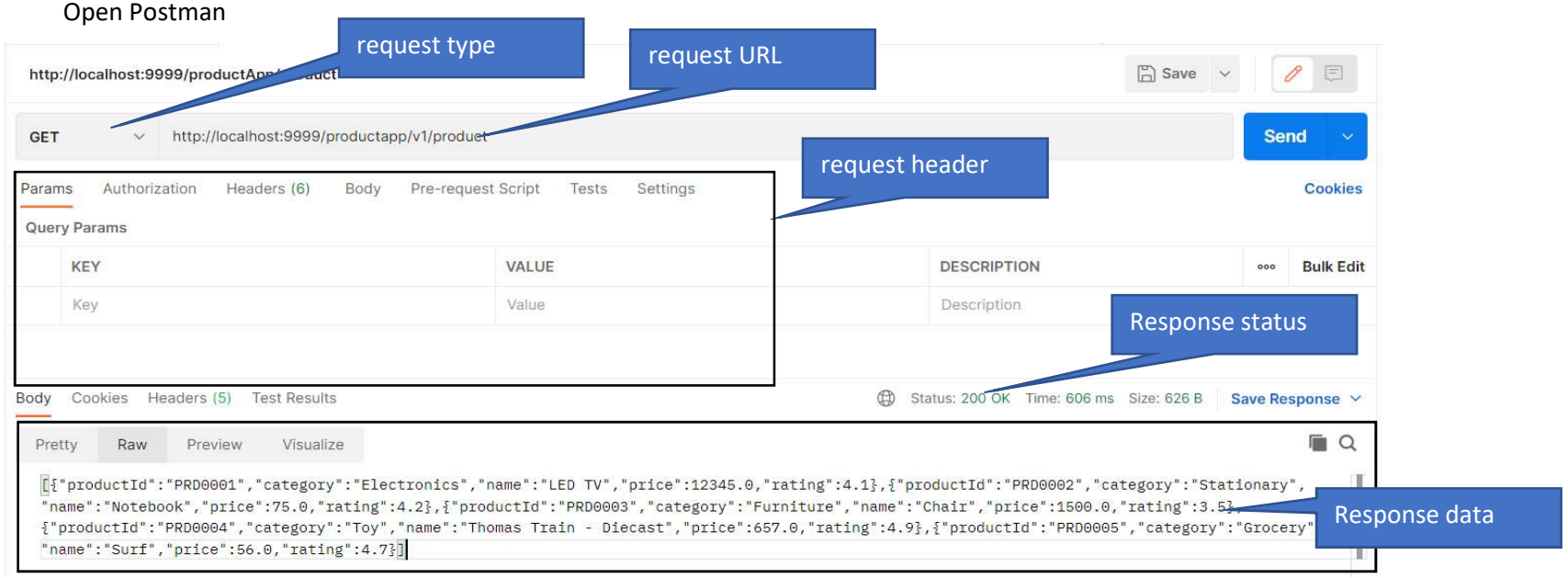
PRODUCT_ID	CATEGORY	NAME	PRICE	RATING
PRD0001	Electronics	LED TV	12345.0	4.1
PRD0002	Stationary	Notebook	75.0	4.2
PRD0003	Furniture	Chair	1500.0	3.5
PRD0004	Toy	Thomas Train - Diecast	657.0	4.9
PRD0005	Grocery	Surf	56.0	4.7

(5 rows, 2 ms)

How to check application

make sure inmemory h2 running and few records inserted

Open Postman

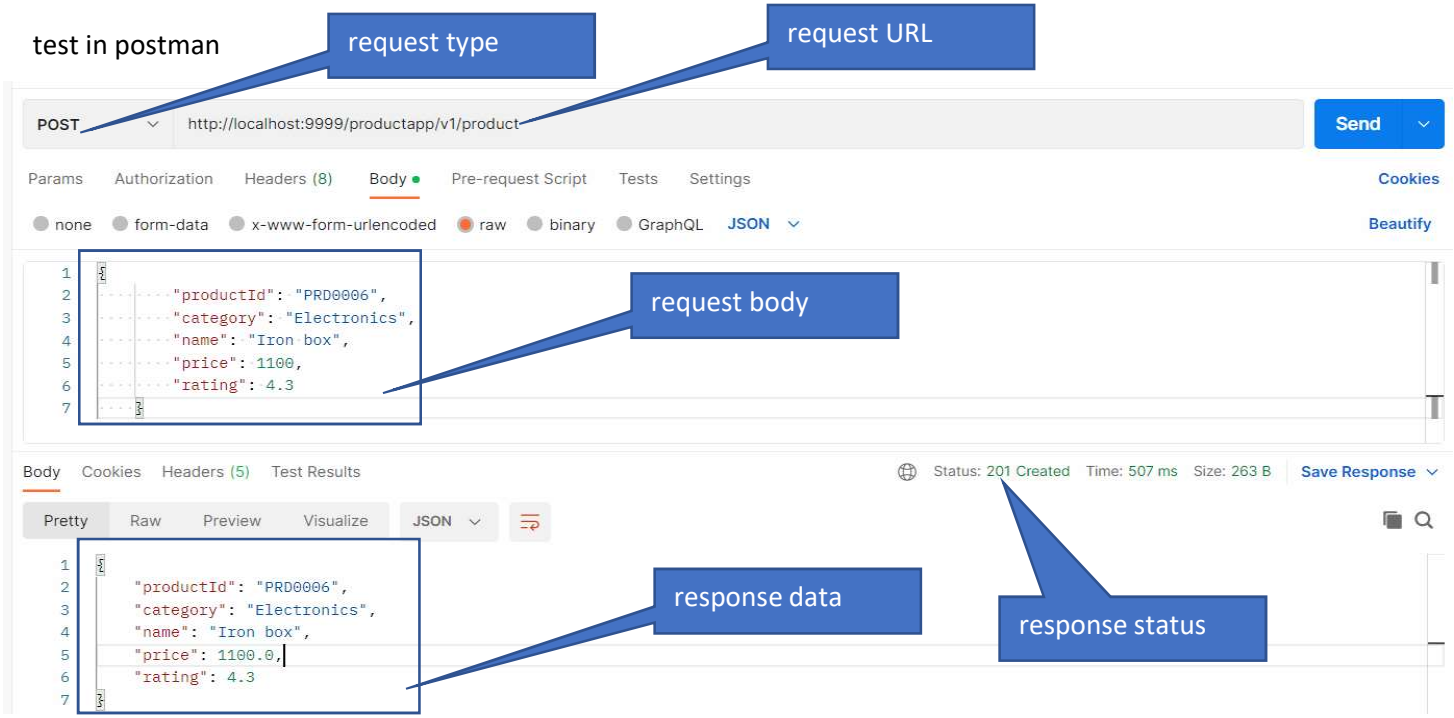


How to add record

define request handler in controller

```
26      // http://localhost:9999/productapp/v1/product   [POST]
27      @PostMapping("/product")
28      public ResponseEntity<?> addProduct(@RequestBody Product prd){
29          Product result = productService.addProduct(prd);
30          return new ResponseEntity<>(result,HttpStatus.CREATED);
31      }
```

test in postman



What client application DO

UI

send request (post/get/put/...)
send request with data
receive response

Postman is doing same without UI

BASE_URL

http://localhost:9999/productapp/v1

to get all product records

GET http://localhost:9999/productapp/v1/product

```
// http://localhost:9999/productapp/v1/product  [GET]
@GetMapping("/product")
public ResponseEntity<?> getProductdata(){
    // return RE object with productdata+status
    List<Product> products = productService.getProducts();
    return new ResponseEntity<>(products, HttpStatus.OK);
}
```

```
@Override
public List<Product> getProducts() {
    return (List<Product>)productRepository.findAll();
}
```

```
7  public interface ProductRepository extends CrudRepository <Product, String> {
8  }
```

to add new product

POST http://localhost:9999/productapp/v1/product

```
// http://localhost:9999/productapp/v1/product   [POST]
@PostMapping("/product")
public ResponseEntity<?> addProduct(@RequestBody Product prd){
    Product result = productService.addProduct(prd);
    return new ResponseEntity<>(result,HttpStatus.CREATED);
}
```

```
@Override
public Product addProduct(Product product) {
    return productRepository.save(product);
}
```

```
7  public interface ProductRepository extends CrudRepository <Product, String> {
8  }
```

to update product

PUT http://localhost:9999/productapp/v1/product

```
// http://localhost:9999/productapp/v1/product   [PUT]
@PutMapping("/product")
public ResponseEntity<?> updateProduct(@RequestBody Product prd){
    Product result = productService.updateProduct(prd);
    return new ResponseEntity<>(result,HttpStatus.OK);
}
```

```
@Override
public Product updateProduct(Product p) {
    return productRepository.save(p);
}
```

DELETE

to delete product

http://localhost:9999/productapp/v1/product /PRD0001

Template variable / Path varable

```
// http://localhost:9999/productapp/v1/product/PRD0001   [DELETE]
@DeleteMapping("/product/{productId}")
public ResponseEntity<?> deleteProduct(@PathVariable String productId){
    productService.deleteProduct(productId);
    return new ResponseEntity<>( body: "deleted",HttpStatus.OK );
}
```

```
@Override
public void deleteProduct(String pid) {
    productRepository.deleteById(pid);
}
```

GET

to get product by id

http://localhost:9999/productapp/v1/get-product-by-id/PRD0001

Template variable / Path varable

```
// http://localhost:9999/productapp/v1/get-product-by-id/PRD0001   [GET]
@GetMapping("/get-product-by-id/{productId}")
public ResponseEntity<?> getProductById(@PathVariable String productId){
    Product result = productService.getProductById(productId);
    return new ResponseEntity<>(result,HttpStatus.OK );
}
```

```
@Override
public Product getProductById(String pid) {
    if(productRepository.findById(pid).isPresent())
    {
        return productRepository.findById(pid).get();
    }
    else {
        return null;
    }
}
```

Iterable<>	findAll()	gives all records
Object	save()	saves / updates record
void	deleteById(id)	deletes record
Optional	findById(id)	