

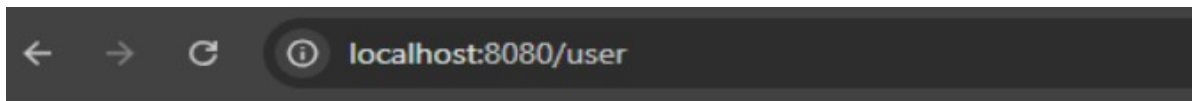
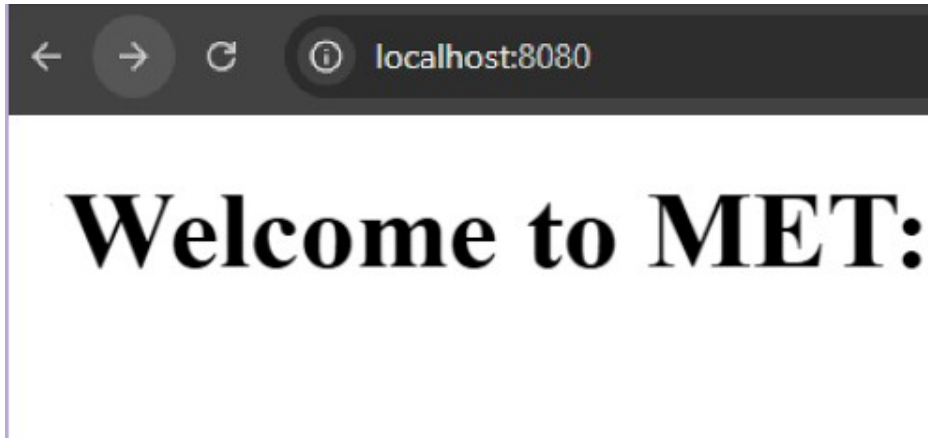
Program No:	24
Roll No :	1525
Title of Program :	Spring Boot and RESTful Web Services
Objective :	1. Write a program to create a simple Spring Boot application that prints a message. 2. Write a program to demonstrate Database Connection with spring boot.

Source Code:

GreetingController.java

```
package edu.met.p1;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class GreetingController {
    @GetMapping("/")
    public String welcomeMeassage()
    {
        return "<html><body>Welcom to MET</body></html>";
    }
    @GetMapping("/user")
    public String welcomeMeassage1()
    {
        return "<html><body>Welcom to MET:Bhairav </body></html>";
    }
}
```

OUTPUT:



Product.java

```
package edu.met.p1;
public class Product {
    int pid;
    String pname;
    int price;
    public int getPid() {
        return pid;
    }
    public void setPid(int pid) {
        this.pid = pid;
    }
    public String getPname() {
        return pname;
    }
}
```

```
public void setName(String pname) {
    this.pname = pname;
}
public int getPrice() {
    return price;
}
public void setPrice(int price) {
    this.price = price;
}
public Product(int pid, String pname, int price) {
    super();
    this.pid = pid;
    this.pname = pname;
    this.price = price;
}
public Product() {
    super();
    // TODO Auto-generated constructor stub
}
}
```

ProductRowMapper.java

```
package edu.met.p1;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;
public class ProductRowMapper implements RowMapper<Product> {
    @Override
    public Product mapRow(ResultSet rs, int rowNum) throws SQLException {
        // TODO Auto-generated method stub
        Product p1=new Product();
        p1.setPid(rs.getInt(1));
        p1.setName(rs.getString(2));
        p1.setPrice(rs.getInt(3));
        return p1;
    }
}
```

ProductController.java

```
package edu.met.p1;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RequestMethod;
@Controller
@RestController
public class ProductController {
    @Autowired
    ProductDao pd;
    @GetMapping("/product")
    public List<Product> getALLProducts(){

return pd.getAll();
    }
    @RequestMapping(value="/product/{id}",method = RequestMethod.GET)
    public @ResponseBody List<Product> getById(@PathVariable("id")String id)
    {
return pd.getById(id);
    }
    @RequestMapping(value="/product/del/{id}",method = RequestMethod.DELETE)
    public @ResponseBody int delById(@PathVariable("id")String id)
    {
return pd.delById(id);
    }
}
```

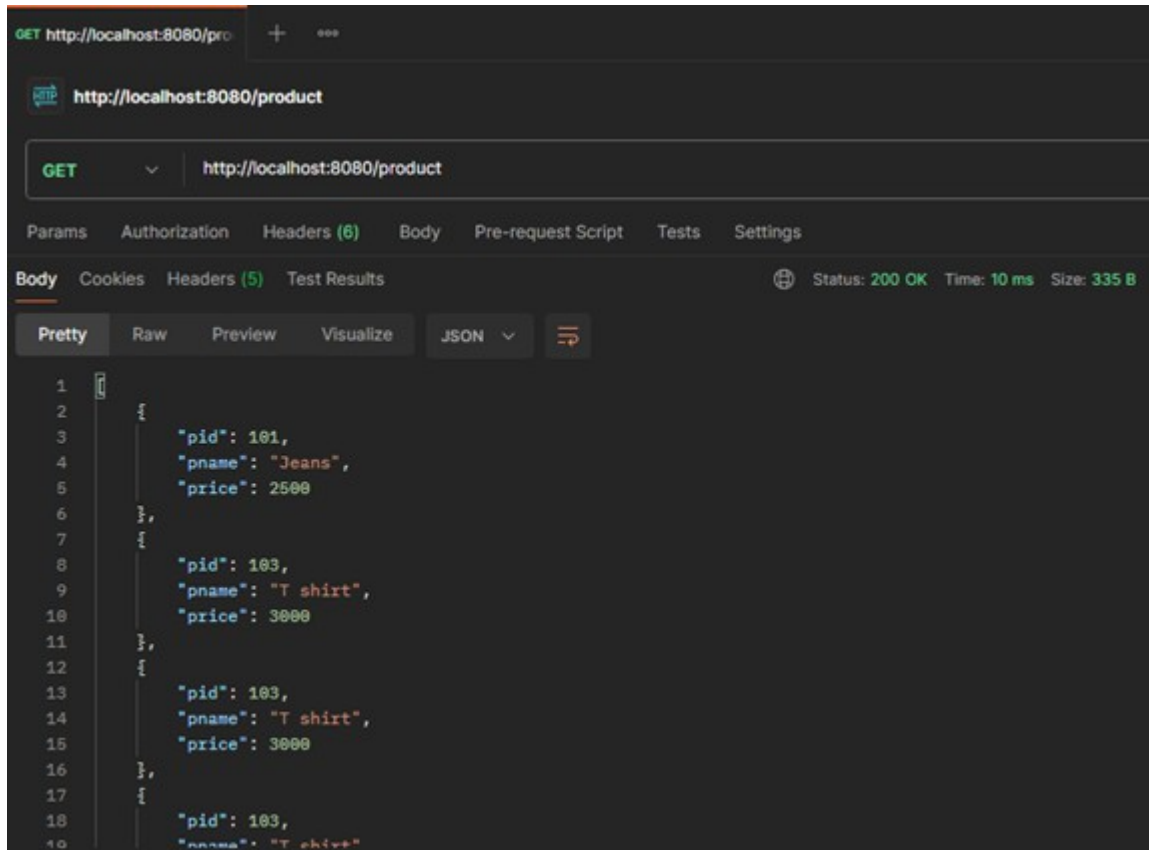
ProductDao.java

```
package edu.met.p1;
import java.util.List;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
@Repository
public class ProductDao {
    @Autowired
    JdbcTemplate jdbcT;
    //fetch all rows
    public List<Product> getAll()
    {
        String sql="select * from products";
        return jdbcT.query(sql, new ProductRowMapper());
    }
    public List<Product> getById(String id)
    {
        String sql="select * from products where pid="+Integer.parseInt(id);
        return jdbcT.query(sql, new ProductRowMapper());
    }

    public int delById(String id)
    {
        String sql="delete from products where pid="+Integer.parseInt(id);
        return jdbcT.update(sql);
    }
}
```

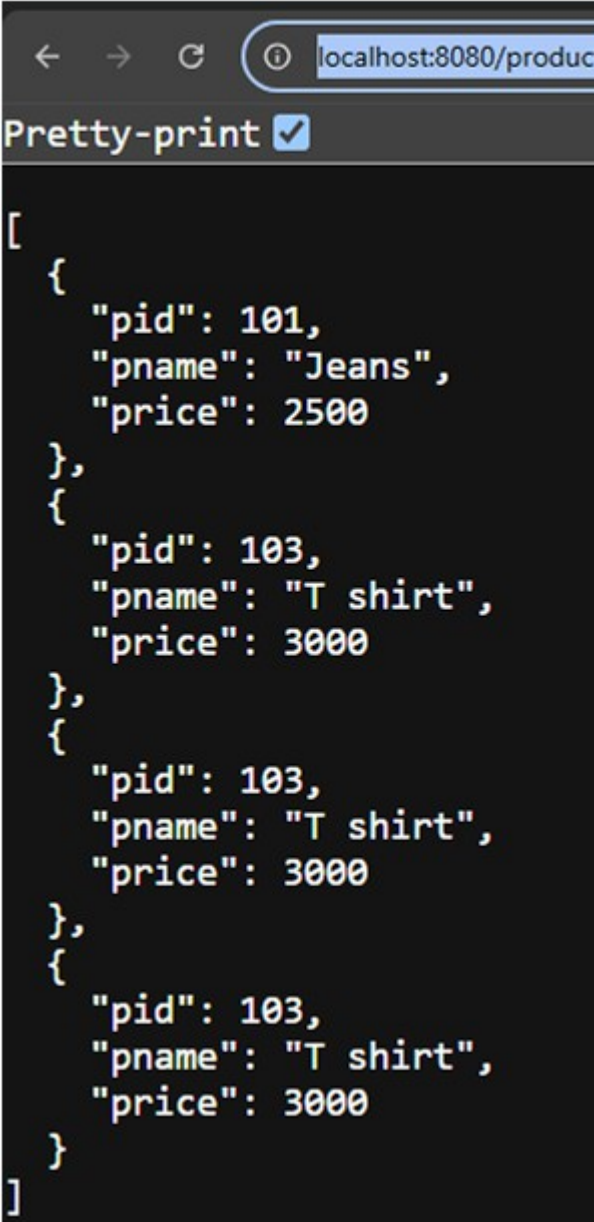
OUTPUT:



The screenshot shows a web browser's developer tools interface. At the top, a GET request to `http://localhost:8080/product` is displayed. Below the request bar, the 'Body' tab is selected, showing a JSON array of product objects. The status bar indicates a 200 OK response with a time of 10 ms and a size of 335 B. The JSON data is as follows:

```
[{"pid": 101, "pname": "Jeans", "price": 2500}, {"pid": 103, "pname": "T shirt", "price": 3000}, {"pid": 103, "pname": "T shirt", "price": 3000}, {"pid": 103, "pname": "T shirt", "price": 3000}]
```

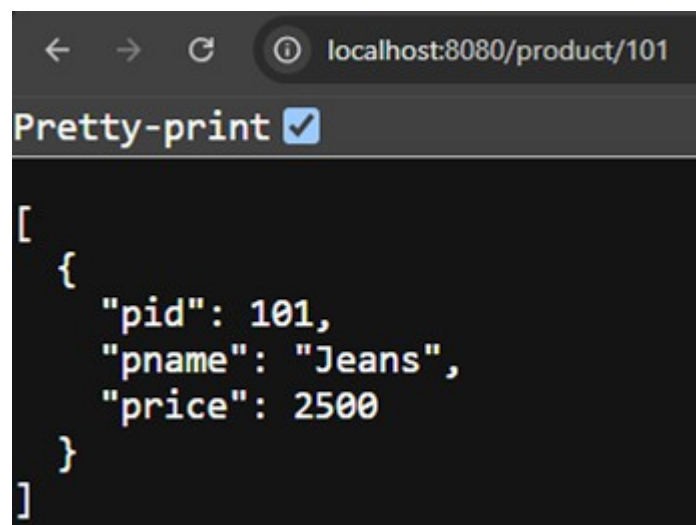
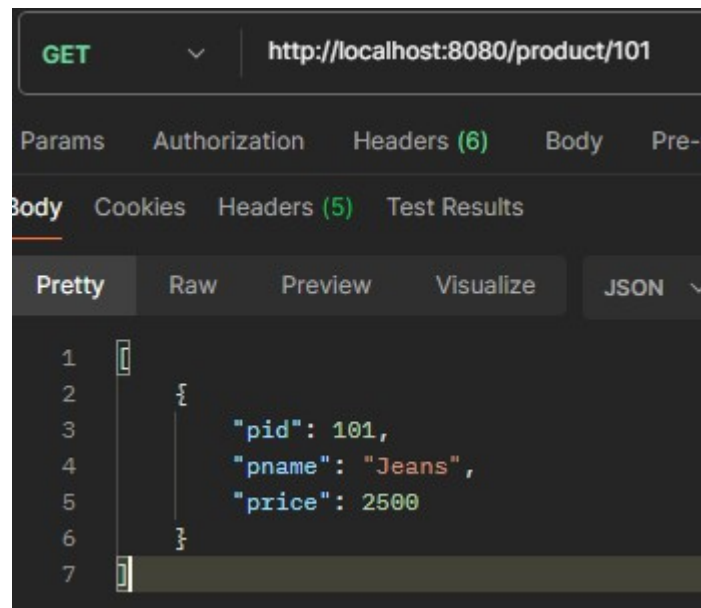
GETALL:



A screenshot of a web browser window. The address bar shows 'localhost:8080/produc'. Below the address bar, there is a 'Pretty-print' checkbox which is checked. The main content area displays a JSON array of three objects. The first object has 'pid': 101, 'pname': 'Jeans', and 'price': 2500. The second and third objects have 'pid': 103, 'pname': 'T shirt', and 'price': 3000.

```
[
  {
    "pid": 101,
    "pname": "Jeans",
    "price": 2500
  },
  {
    "pid": 103,
    "pname": "T shirt",
    "price": 3000
  },
  {
    "pid": 103,
    "pname": "T shirt",
    "price": 3000
  },
  {
    "pid": 103,
    "pname": "T shirt",
    "price": 3000
  }
]
```

GET BY ID



Delete By ID

