

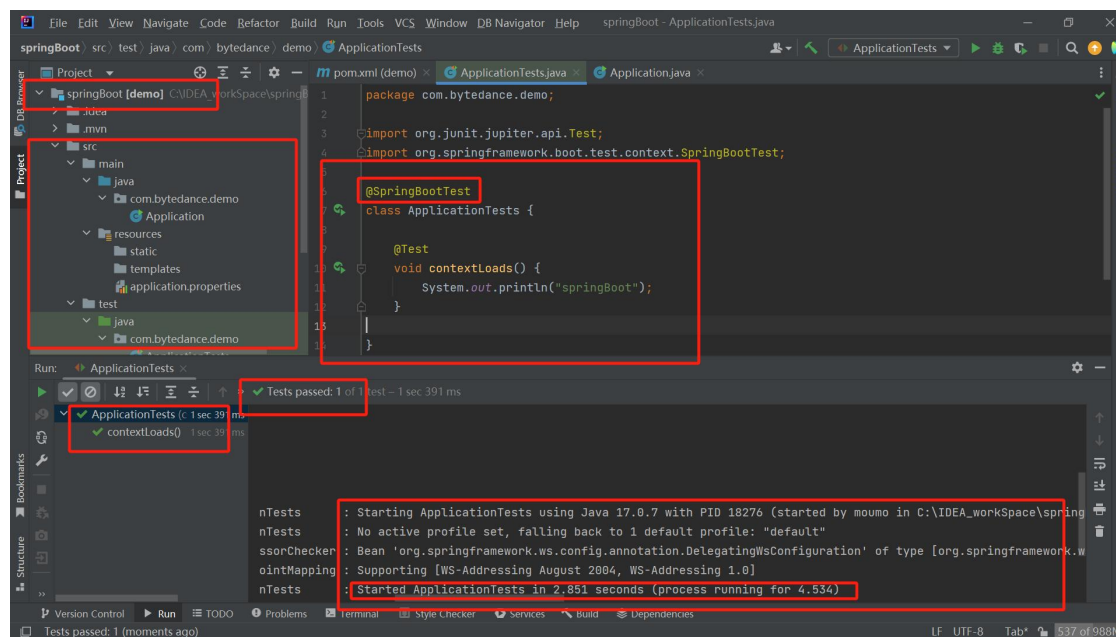
SpringBoot

任务要求1

能够成功启动springboot空项目

1. 启动连接/生成初始项目包: <https://start.spring.io/>
2. SpringBoot 教程: <https://www.cainiaojc.com/springboot/springboot-tutorial.html>
3. 启动空 springboot 空项目:

3.1. Output



任务2

认识Restful和CRUD

两个网络通信标准规范，一个面向前端，另一个面向数据库。

仿照<https://www.cainiaojc.com/springboot/springboot-rest-example.html>中的REST示例，写一个press的REST接口，用postman进行校验。

预先向数据库中存储一篇id为1的文章，做到使用get请求访问localhost:port/press/1的时候，能够以json格式返回文章数据。

1. RESTful 架构详解/...什么是 RESTful? .../... :

<https://www.runoob.com/w3cnote/restful-architecture.html>

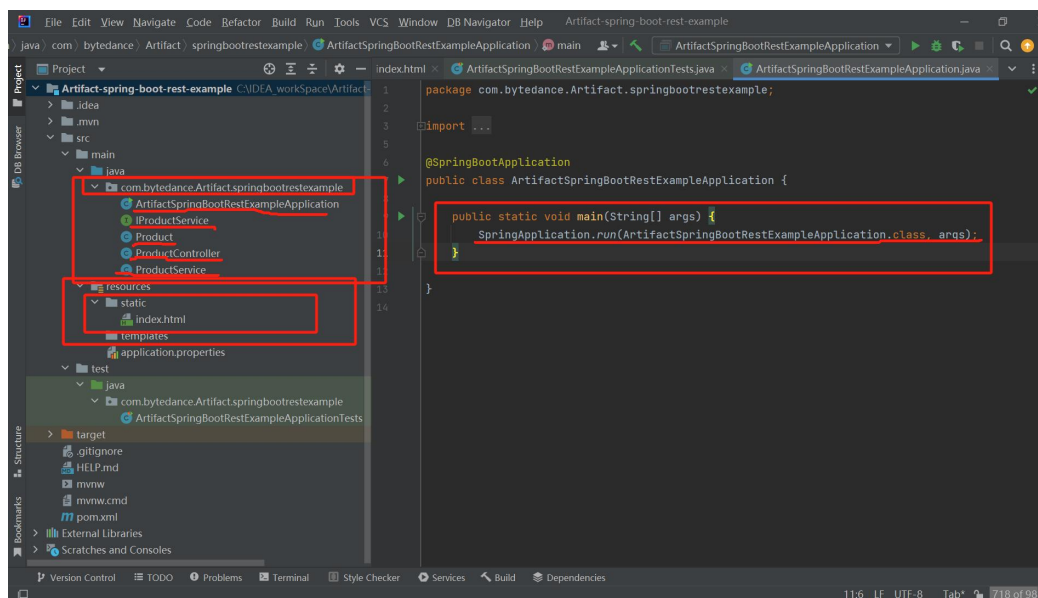
2. crud/CRUD - (Create, Read, Update, Delete):

<https://baike.baidu.com/item/crud/3606157?fr=aladdin>

3. 复现 “<https://www.cainiaojc.com/springboot/springboot-rest-example.html>” 中的 REST 示例：

(Using using an in-memory list instead of using a database like MySQL)

3.1. In 'C:\IDEA_workSpace\Artifact-spring-boot-rest-example':



3.2. Product.java:

```
Product.java × ProductController.java × IProductService.java × ProductService.java × index.html
1 package com.bytedance.Artifact.springbootrestexample;
2
3 12 usages
4 public class Product {
5     3 usages
6     private int id;
7     3 usages
8     private String pname;
9     3 usages
10    private String batchno;
11    3 usages
12    private double price;
13    3 usages
14    private int noofproduct;
15
16    //默认构造函数
17    no usages
18    public Product()
19    {
20    }
21
22    // Constructor
23    6 usages
24    public Product(int id, String pname, String batchno, double price, int noo
25        this.id = id;
```

3.3. ProductController.java:

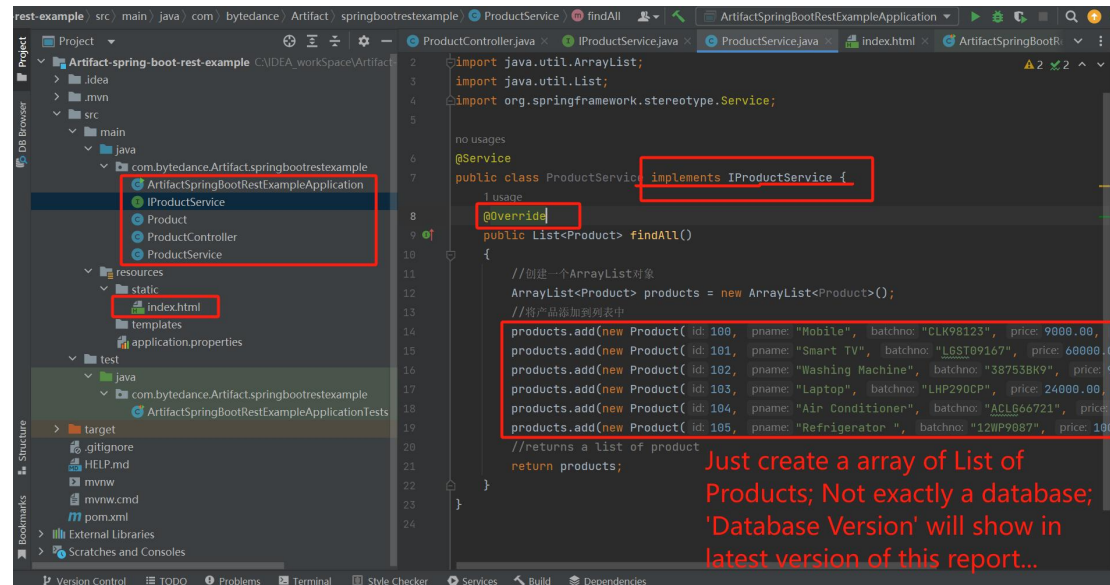
```
ProductController.java × IProductService.java × ProductService.java × index.html × ArtifactSpringB
1 package com.bytedance.Artifact.springbootrestexample;
2 import java.util.List;
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 no usages
8 @RestController
9 public class ProductController {
10     1 usage
11     @Autowired
12     private IProductService productService;
13     //将getProduct()方法映射到/product
14     no usages
15     @GetMapping(value = "/product")
16     public List<Product> getProduct()
17     {
18         //查找所有产品
19         List<Product> products = productService.findAll();
20         //返回产品列表
21         return products;
22     }
23 }
```

3.4. IProductService(Interface_Product_Service):

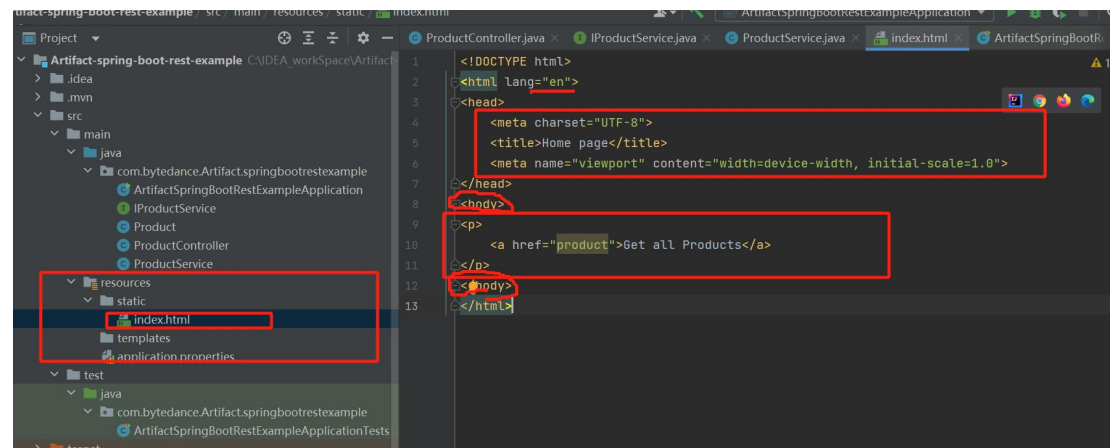
```
ProductController.java × IProductService.java × ProductService.java × index.html ×
1 package com.bytedance.Artifact.springbootrestexample;
2 import java.util.List;
3
4 2 usages 1 implementation
5 public interface IProductService {
6     1 usage 1 implementation
7     List<Product> findAll();
8 }
9 |
```

3.5. ProductService:

(Using using an in-memory list instead of using a database like MySQL)

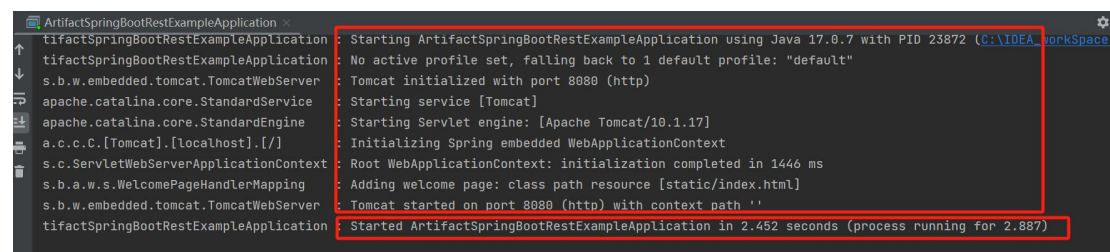


3.6. Index.html:



3.7. Output:

- Terminal



- 打开浏览器并调用 URL <http://localhost:8080/index.html>。它显示了 获取所有产品的链接，如下图所示：

```

1 | [
2 |   {
3 |     "id": 100,
4 |     "pname": "Mobile",
5 |     "batchno": "CLK98123",
6 |     "price": 9000,
7 |     "noofproduct": 6
8 |   },
9 |   {
10 |    "id": 101,
11 |    "pname": "Smart TV",
12 |    "batchno": "LGS709167",
13 |    "price": 60000,
14 |    "noofproduct": 3
15 |   },
16 |   {
17 |    "id": 102,
18 |    "pname": "Washing Machine",
19 |    "batchno": "387538K9",
20 |    "price": 9000,
21 |    "noofproduct": 7
22 |   },
23 |   {
24 |    "id": 103,
25 |    "pname": "Laptop",
26 |    "batchno": "LHP230CP",
27 |    "price": 24000,
28 |    "noofproduct": 1
29 |   },
30 |   {
31 |    "id": 104,
32 |    "pname": "Air Conditioner",
33 |    "batchno": "ACL066721",
34 |    "price": 30000,
35 |    "noofproduct": 5
36 |   },
37 |   {
38 |    "id": 105,
39 |    "pname": "Refrigerator",
40 |    "batchno": "12WF9087",
41 |    "price": 10000,
42 |    "noofproduct": 4
43 |   }
44 | ]

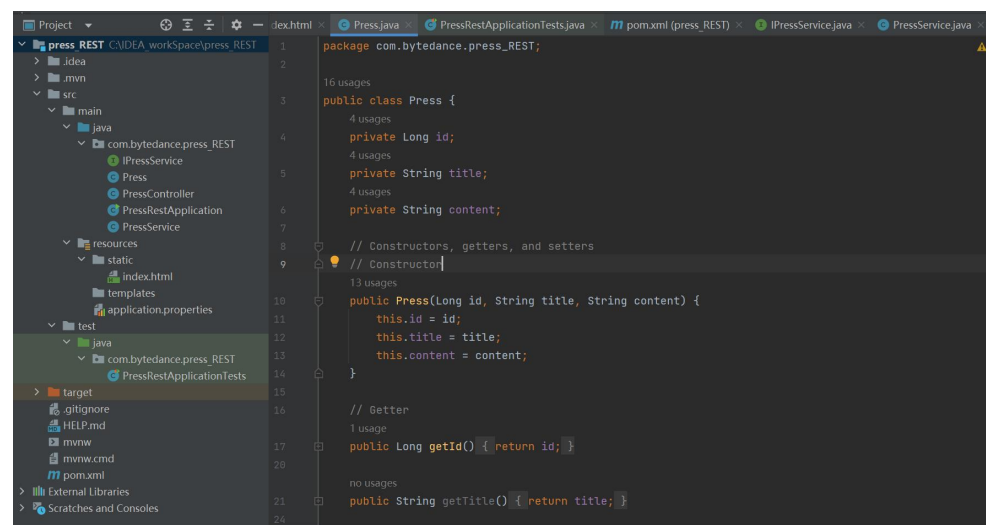
```

4. 仿照 <https://www.cainiaojc.com/springboot/springboot-rest-example.html> 中的 REST 示例，写一个 press 的 REST 接口，用 postman 进行校验(疑问：如何‘用 postman 进行校验？’，是类似于 curl...命令吗？)：

(Using using an in-memory list instead of using a database like MySQL)

“C:\IDEA_workSpace\press_REST”:

4.1. JavaBean creation: (Press.java)



...

- PressService.java:

(Using using an in-memory list instead of using a database like MySQL)

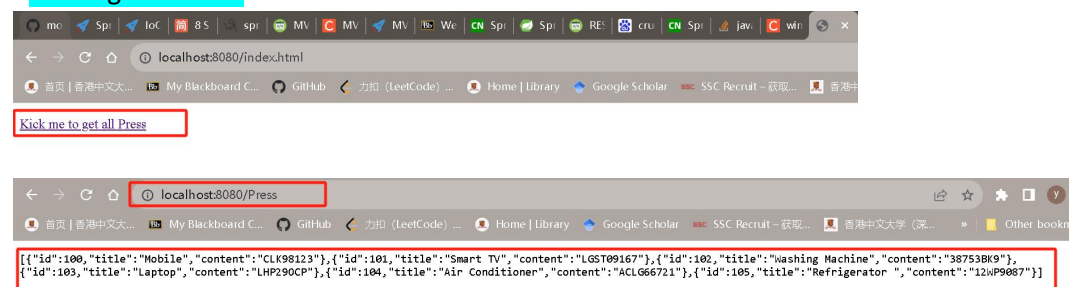
```

18  @Service
19  public class PressService implements IPressService {
20      // Usage
21      @Override
22      public List<Press> findAll() {
23          // 创建一个ArrayList对象
24          ArrayList<Press> presses = new ArrayList<>();
25          // 将产品添加到列表中
26          presses.add(new Press( id: 100L, title: "Mobile", content: "CLK98123"));
27          presses.add(new Press( id: 101L, title: "Smart TV", content: "LGS109167"));
28          presses.add(new Press( id: 102L, title: "Washing Machine", content: "38753BK9"));
29          presses.add(new Press( id: 103L, title: "Laptop", content: "LHP290CP"));
30          presses.add(new Press( id: 104L, title: "Air Conditioner", content: "ACL666721"));
31          presses.add(new Press( id: 105L, title: "Refrigerator ", content: "12WP9887"));
32          // returns a list of product
33          return presses;
34      }
35
36      2 usages
37      private final Map<Long, Press> pressMap = new HashMap<>();
38
39      no usages
40      public PressService() {
41          // Initialize with some sample data
42          Press press1 = new Press( id: 1L, title: "I am the title", content: "I am the content!");
43      }
44  }

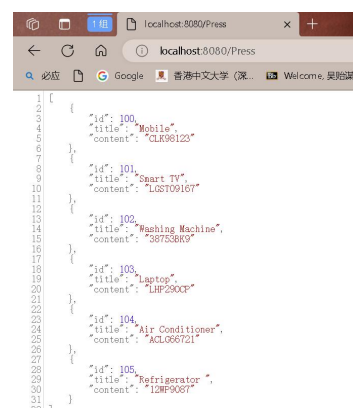
```

- 打开浏览器并调用 URL <http://localhost:8080/index.html> ('index.html' here is a folder under "C:\IDEA_workSpace\press-REST_databaseVersion\src\main\resources\static\index.html" !).
- 它显示了 获取所有产品的链接，如下图所示：

- In Google Chrome:



- In Microsoft Edge:

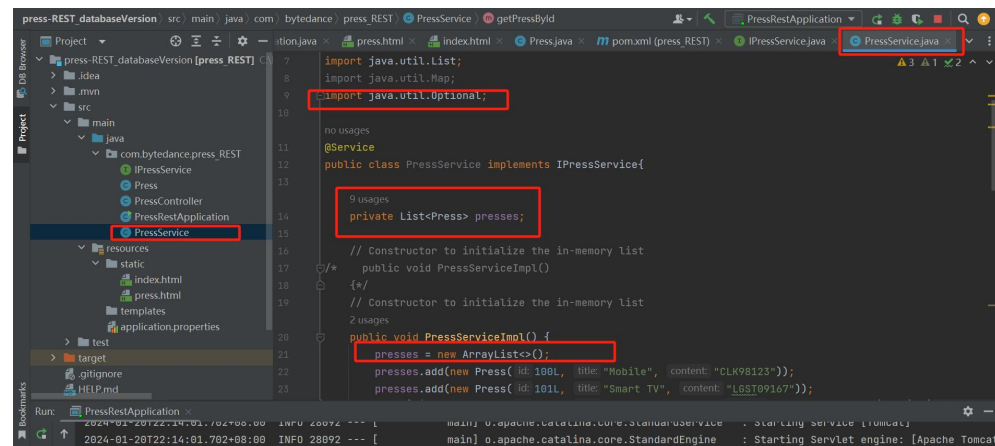


5. - 仿照 <https://www.cainiaojc.com/springboot/springboot-rest-example.html> 中的 REST 示例，写一个 press 的 REST 接口，用 postman 进行校验。

- 预先向数据库中存储一篇 id 为 1 的文章，做到使用 get 请求访问 `localhost:port/press/1` (`localhost:8080/press/1` here, since) 的时候，能够以 json 格式返回文章数据。

Modify parts:

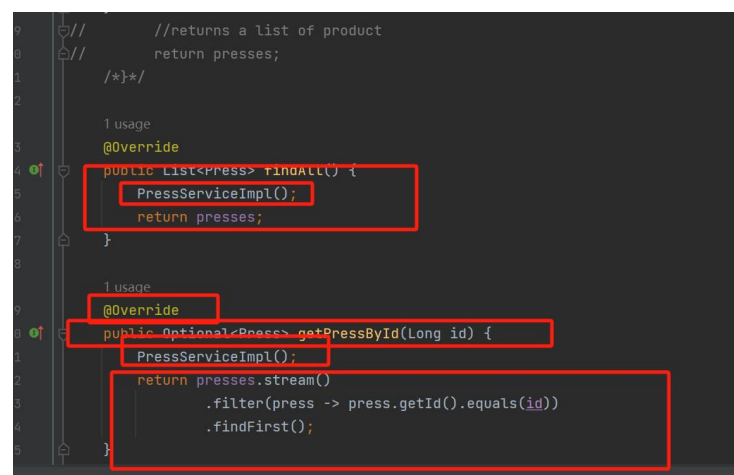
5.1. PressService.java:



```
import java.util.List;
import java.util.Map;
import java.util.Optional;

no usages
@Service
public class PressService implements IPressService {
    private List<Press> presses;

    // Constructor to initialize the in-memory list
    public void PressServiceImpl() {
        // Constructor to initialize the in-memory list
        presses = new ArrayList<>();
        presses.add(new Press(id: 100L, title: "Mobile", content: "DLK98123"));
        presses.add(new Press(id: 101L, title: "Smart TV", content: "L6ST09167"));
    }
}
```



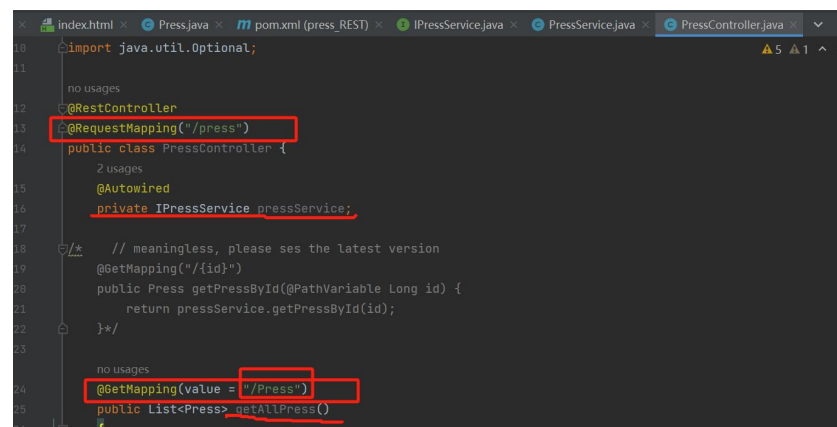
```
//returns a list of product
//return presses;

/* */

1 usage
@Override
public List<Press> findAll() {
    PressServiceImpl();
    return presses;
}

1 usage
@Override
public Optional<Press> getPressById(Long id) {
    PressServiceImpl();
    return presses.stream()
        .filter(press -> press.getId().equals(id))
        .findFirst();
}
```

5.2. PressController.java:



```
import java.util.Optional;

no usages
@RestController
@RequestMapping("/press")
public class PressController {
    @Autowired
    private IPressService pressService;

    // meaningless, please see the latest version
    @GetMapping("/{id}")
    public Press getPressById(@PathVariable Long id) {
        return pressService.getPressById(id);
    }

    no usages
    @GetMapping(value = "/press")
    public List<Press> getAllPress() {
        return pressService.findAll();
    }
}
```

```

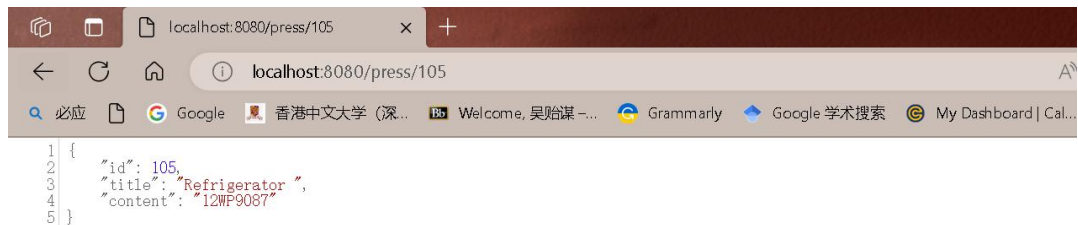
4  @GetMapping(value = "/Press")
5  public List<Press> getAllPress()
6  {
7      // 查找所有产品
8      List<Press> presses = pressService.findAll();
9
10     // 返回产品列表
11     return presses;
12 }

no usages
4  @GetMapping("/{id}")
5  public Press getPressById(@PathVariable Long id) {
6      Optional<Press> press = pressService.getPressById(id);
7      return press.orElse( other: null);
8  }
9  }

```

5.3. Output

"localhost:8080/press/105":

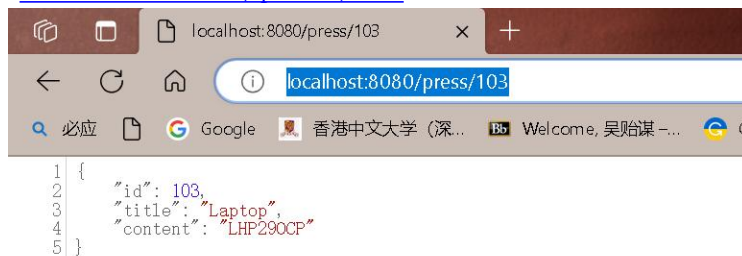


```

1  {
2      "id": 105,
3      "title": "Refrigerator",
4      "content": "12WFP9087"
5  }

```

"localhost:8080/press/103":



```

1  {
2      "id": 103,
3      "title": "Laptop",
4      "content": "LHP290CP"
5  }

```

"localhost:8080/press/100":



```

1  {
2      "id": 100,
3      "title": "Mobile",
4      "content": "CLK98123"
5  }

```


["localhost:8080/press/Press"](http://localhost:8080/press/Press):



The screenshot shows a web browser window with the address bar displaying `localhost:8080/press/Press`. The page content is a JSON array of five objects, each representing a product. The objects are: 1. `{ "id": 100, "title": "Mobile", "content": "CLK98123" }`, 2. `{ "id": 101, "title": "Smart TV", "content": "LGST09167" }`, 3. `{ "id": 102, "title": "Washing Machine", "content": "38753BK9" }`, 4. `{ "id": 103, "title": "Laptop", "content": "LHP290CP" }`, and 5. `{ "id": 104, "title": "Air Conditioner", "content": "ACLG66721" }`. The array is enclosed in square brackets `[` and `]`. The browser's address bar and the JSON content are highlighted with red boxes.

```
1 [
2   {
3     "id": 100,
4     "title": "Mobile",
5     "content": "CLK98123"
6   },
7   {
8     "id": 101,
9     "title": "Smart TV",
10    "content": "LGST09167"
11  },
12  {
13    "id": 102,
14    "title": "Washing Machine",
15    "content": "38753BK9"
16  },
17  {
18    "id": 103,
19    "title": "Laptop",
20    "content": "LHP290CP"
21  },
22  {
23    "id": 104,
24    "title": "Air Conditioner",
25    "content": "ACLG66721"
26  },
27  {
28    "id": 105,
29    "title": "Refrigerator ",
30    "content": "12WP9087"
31  }
32 ]
```

["Home page\(moumouta\)" / "localhost:8080/index.html"](http://localhost:8080/index.html)



After Kick the link "Kick me to get all Press":

Same as ["localhost:8080/press/Press"](http://localhost:8080/press/Press) page.

- 用 postman 进行校验

```
Terminal: Local ~ + -
PS C:\IDEA_workspace\press-REST_databaseVersion> curl http://localhost:8080/press/Press

StatusCode      : 200
StatusDescription:
Content         : [{"id":100,"title":"Mobile","content":"CLK98123"}, {"id":101,"title":"Smart TV","content":"L6ST09167"}, {"id":102,"title":"Washing Machine","content":"38753BK9"}, {"id":103,"title":"Laptop","content":"LH...
RawContent      : HTTP/1.1 200
                  Transfer-Encoding: chunked
                  Content-Type: application/json
                  Date: Sat, 20 Jan 2024 14:46:12 GMT

Forms           : {}
Headers         : [{"Transfer-Encoding, chunked}, {"Content-Type, application/json}, {"Date, Sat, 20 Jan 2024 14:46:12 GMT}]
```

- 缺点/未完成的地方:

只能实现 in-memory list 而不是 using a database like MySQL。

在网上查了如何（远程）连接（绑定）SpringBoot 和 MySQL 数据库，但实现过程中出现了较大阻碍。

请老师帮忙指点下~