

Kai Hedrick

Activity 3

CST-339

Instructor Mauger

7/3/24

Screenshots:

```
2024-07-03T16:33:24.507-07:00 INFO 115044 --- [t
Hello from the OrdersBusinessService
Hello from the OrdersBusinessService
Hello from the OrdersBusinessService
█
```

```
2024-07-03T17:04:27.804-07:00 INFO 115044 --- [topic22] [nio-8080-exec-1]
2024-07-03T17:04:27.864-07:00 INFO 115044 --- [topic22] [nio-8080-exec-1]
2024-07-03T17:04:27.865-07:00 INFO 115044 --- [topic22] [nio-8080-exec-1]
Hello from the AnotherOrdersBusinessService
█
```

```
2024-07-03T17:45:23.910-07:00 INFO 161320 --- [
2024-07-03T17:45:23.910-07:00 INFO 161320 --- [
Hello from the SecurityBusinessService
Hello from the SecurityBusinessService
█
```

localhost:8080/login/doLogin			
Order Number	Product Name	Price	Quantity
0000000001	Product 1	1.0	1
0000000002	Product 2	2.0	2
0000000003	Product 3	3.0	3
0000000004	Product 4	4.0	4
0000000005	Product 5	5.0	5

```

2024-07-03T22:58:24.743-07:00 INFO 329036 --- [topic22] [ resta
2024-07-03T22:58:24.745-07:00 INFO 329036 --- [topic22] [ resta
this is the init method
2024-07-03T22:58:25.198-07:00 INFO 329036 --- [topic22] [ resta
2024-07-03T22:58:25.221-07:00 INFO 329036 --- [topic22] [ resta
2024-07-03T22:58:25.228-07:00 INFO 329036 --- [topic22] [ resta
2024-07-03T22:58:29.883-07:00 INFO 329036 --- [topic22] [nio-808
2024-07-03T22:58:29.883-07:00 INFO 329036 --- [topic22] [nio-808
2024-07-03T22:58:29.884-07:00 INFO 329036 --- [topic22] [nio-808
Hello from the SecurityBusinessService

```

The `init()` method of `OrderBusinessService` would be called once during the initialization of the `ordersBusinessService` bean where the Spring application context starts up.

`@Configuration` indicates it contains bean definitions for the Spring application context. The `init()` method is called by the Spring container after the bean is instantiated and initialized, to receive the console message. The `destroy()` method is called before the bean is destroyed and when the application stops.

```

2024-07-04T14:13:58.165-07:00 INFO 51136 --- [
Hello from the SecurityBusinessService
this is the init method
Hello from the SecurityBusinessService
this is the init method

```

The `OrdersBusinessService` `init()` method is called each time a new instance of the bean is created because it is scoped as `prototype`. This means that the `init()` method is called every time the bean is requested, leading to multiple calls if the form is submitted multiple times, as seen in the console output.

```

2024-07-04T14:25:14.635-07:00 INFO 59948 ---
Hello from the SecurityBusinessService
this is the init method
This is the destroy method

```

The OrdersBusinessService init() method is called each time a new instance of the bean is created, which happens every time the login form page is requested because the bean is now scoped to the request. This means that the init() method is called every time the form is submitted, resulting in multiple init() calls as seen in the console output.

```

2024-07-04T15:27:05.259-07:00 INFO 59948 --
Hello from the SecurityBusinessService
this is the init method

```

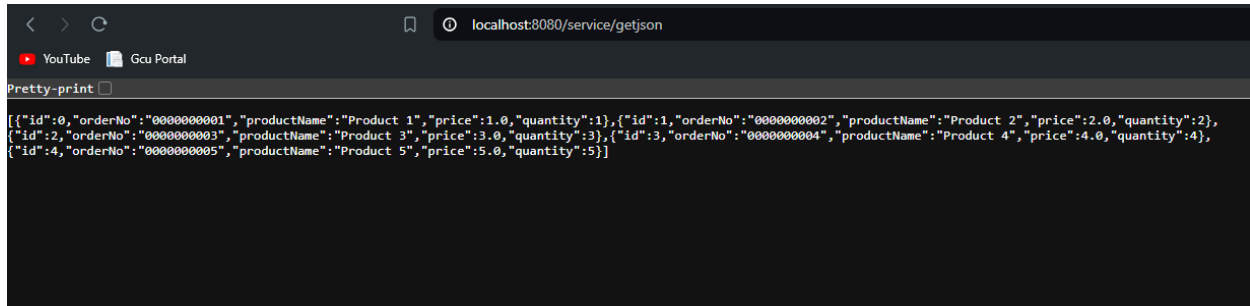
The OrdersBusinessService init() method is called once per unique browser session because the bean is now in session scope. This means that init() is called the first time the login form is requested in each new session, resulting in one call per browser session as seen in the console output.

```

2024-07-04T15:32:31.607-07:00 INFO 59948 --- [topi
2024-07-04T15:32:31.607-07:00 INFO 59948 --- [topi
2024-07-04T15:32:31.608-07:00 INFO 59948 --- [topi
Hello from the SecurityBusinessService

```

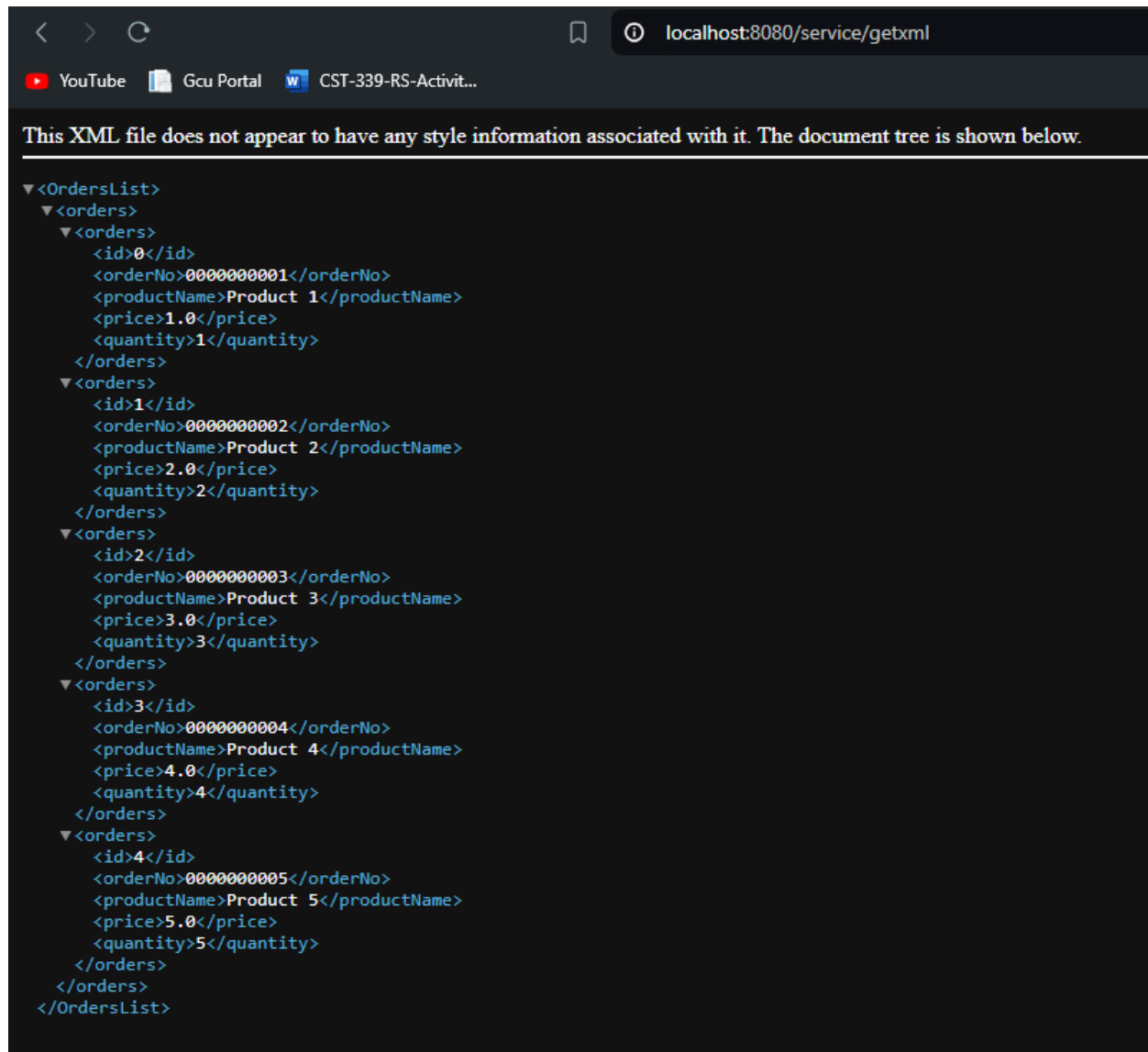
The OrdersBusinessService init() method is called only once when the application context is initialized because the bean is now in singleton scope. This means that the init() method is not called again with each form submission, resulting in a single call to init() as seen in the console output.



```
localhost:8080/service/getjson

Pretty-print ☐

[[{"id":0,"orderNo":"0000000001","productName":"Product 1","price":1.0,"quantity":1},{ "id":1,"orderNo":"0000000002","productName":"Product 2","price":2.0,"quantity":2}, {"id":2,"orderNo":"0000000003","productName":"Product 3","price":3.0,"quantity":3},{ "id":3,"orderNo":"0000000004","productName":"Product 4","price":4.0,"quantity":4}, {"id":4,"orderNo":"0000000005","productName":"Product 5","price":5.0,"quantity":5}]
```



The screenshot shows a web browser window with the address bar displaying `localhost:8080/service/getxml`. The browser's tab bar includes links to YouTube, Gcu Portal, and CST-339-RS-Activit... Below the address bar, a message states: "This XML file does not appear to have any style information associated with it. The document tree is shown below." The XML document tree is displayed in a dark-themed editor. The root element is `<OrdersList>`, which contains a single child element `<orders>`. This `<orders>` element contains five sub-elements, each representing an order. Each order element has the following structure: `<id>`, `<orderNo>`, `<productName>`, `<price>`, and `<quantity>`. The orders are numbered 1 through 5, with product names "Product 1" through "Product 5", and prices ranging from 1.0 to 5.0. The quantity for each order is equal to its ID.

```
<?xml version='1.0'>
<OrdersList>
  <orders>
    <orders>
      <id>0</id>
      <orderNo>0000000001</orderNo>
      <productName>Product 1</productName>
      <price>1.0</price>
      <quantity>1</quantity>
    </orders>
    <orders>
      <id>1</id>
      <orderNo>0000000002</orderNo>
      <productName>Product 2</productName>
      <price>2.0</price>
      <quantity>2</quantity>
    </orders>
    <orders>
      <id>2</id>
      <orderNo>0000000003</orderNo>
      <productName>Product 3</productName>
      <price>3.0</price>
      <quantity>3</quantity>
    </orders>
    <orders>
      <id>3</id>
      <orderNo>0000000004</orderNo>
      <productName>Product 4</productName>
      <price>4.0</price>
      <quantity>4</quantity>
    </orders>
    <orders>
      <id>4</id>
      <orderNo>0000000005</orderNo>
      <productName>Product 5</productName>
      <price>5.0</price>
      <quantity>5</quantity>
    </orders>
  </orders>
</OrdersList>
```

JSON from Postman:

```
[  
  {  
    "id": 0,  
    "orderNo": "0000000001",  
    "productName": "Product 1",  
    "price": 1.0,  
    "quantity": 1  
  },  
  {  
    "id": 1,  
    "orderNo": "0000000002",  
    "productName": "Product 2",  
    "price": 2.0,  
    "quantity": 2  
  },  
  {  
    "id": 2,  
    "orderNo": "0000000003",  
    "productName": "Product 3",  
    "price": 3.0,  
    "quantity": 3  
  },  
  {  
    "id": 3,  
    "orderNo": "0000000004",
```

```

    "productName": "Product 4",
    "price": 4.0,
    "quantity": 4
  },
  {
    "id": 4,
    "orderNo": "0000000005",
    "productName": "Product 5",
    "price": 5.0,
    "quantity": 5
  }
]

```

XML from Postman:

```

[{"id":0,"orderNo":"0000000001","productName":"Product
1","price":1.0,"quantity":1},{ "id":1,"orderNo":"0000000002","productName":"Product
2","price":2.0,"quantity":2},{ "id":2,"orderNo":"0000000003","productName":"Product
3","price":3.0,"quantity":3},{ "id":3,"orderNo":"0000000004","productName":"Product
4","price":4.0,"quantity":4},{ "id":4,"orderNo":"0000000005","productName":"Product
5","price":5.0,"quantity":5}]

```

No parameters required for both XML, and JSON. Get methods make up each of these responses.