

OBJECTIVE QUESTIONS

Intro To Spring

1

What is a "Bean" in Spring?

- Component
- Object
- Class
- Container

Reset

Save

Question 1

shreyas.bedagkar
2-JAN-2023

00 : 14 : 59

Hours Minutes Seconds

OBJECTIVE QUESTIONS

Spring IoC

1 2 3

Question 1

Consider a CustomerService class in the package com.infosys with appropriate properties and
Which of the following is the CORRECT option to define a bean of CustomerService class in the
configuration class?

- @Component
- @Configuration
- @Bean
- @Service

Reset

Save

Spring IoC

1

2

3

Question 2

A Spring standalone application has followed Java configuration to configure its beans. It is loaded as follows assuming AppConfig.java as the configuration file.

```
ApplicationContext applicationContext = new AnnotationConfigApplicationContext();
```

Which of the following statements is/are FALSE with respect to the above-given code?

- a. Replace AnnotationConfigApplicationContext with ApplicationContext to load the configuration file.
- b. For Java configuration to work correctly, replace AnnotationConfigApplicationContext with AnnotationConfigApplicationContext(AppConfig.java)

- Only a
- Only b
- Both a and b
- Neither a nor b

ResetSave

Spring IoC

1 2 3

Question 3

Identify which among the following statements is/are TRUE with respect to S

- a. The Spring IoC Container creates the Java objects that needs to be injected
- b. The Spring IoC Container takes only the metadata file as input to generate
- c. The Spring IoC Container takes only the POJO classes as input to generate
- d. The Spring IoC Container takes both the metadata file and POJO classes as input to generate

- Only a) and d)
- Only a) and b)
- Only a)
- Only d)

Reset

Save

Dependency Injection

1

2

}

The main method of the application includes the following statements:

```
AbstractApplicationContext context = new AnnotationConfigApplicationContext();
//Line2
bookService.setBookName("Java 9");
System.out.println(bookService.getBookName());
```

Which of the following given option/s can be used in place of Line2 in the code?

- a. BookService bookService = (BookService)context.getBean("bookService");
- b. BookService bookService = context.getBean("bookService");
- c. BookService bookService = context.getBean("bookService", BookService.class);

- Only a)
- Only a) and b)
- Only a) and c)
- Only b)

Reset

Save

Dependency Injection

1

2

Question 2

Which of the following is/are the advantages of Dependency Injection(DI)?

- a. Loose coupling
- b. Re-use of code, ease of testing and maintainable
- c. Separation of responsibility by keeping code and configuration separately.

- Only a)
- Only b)
- Only c)
- All the given options

Reset

Save

Autowiring

1

2

Question

Analyze the code below. Assume class B and a bean of class B exists in the

public class A{

 private B b;

 public A(){ }

 @Autowired

 public void setB(B b){

 this.b=b;

}

}

Which of the following Spring feature is observed in the above given code?

- Spring AOP
- Setter injection
- Constructor Injection
- Spring Auto-scanning

Autowiring

1

2

```
        return addressService;
    }

    @Bean(name = "addr2")
    public AddressService addressService() {
        AddressService addressService = new AddressService();
        addressService.setCity("Mysore");
        return addressService;
    }

    @Bean(value = "customer")
    public CustomerService customerService() {
        return new CustomerService();
    }
}
```

The main method of the application includes the following statements:
AbstractApplicationContext context = new AnnotationConfigApplicationContext();
CustomerService customer = (CustomerService)context.getBean("customer");
System.out.println(customer.getAddressService().getCity());

What will be the output, when the above code is executed?

- Bangalore
- Mysore
- BeanInstantiationException: Could not instantiate customer bean
- Compilation error: Application cannot use both annotation and xml based configuration

John finds the @Scope annotation usage in the code during his code walk behavior of @Scope. Help him by identifying the RIGHT options from the

- a. @Scope can be used as a method-level annotation
- b. @Scope can be used as a class level annotation
- c. By default, @Scope attribute is set to prototype
- d. @Scope has to be used mandatorily while defining bean using @Bear

- Only a)
- Only b)
- Only a) and b)
- Only d)

Reset

Save

Question 1

Autoscan

1

2

Which of the below-given options is/are CORRECT with respect to the Auto scan?

- a. @ComponentScan tag is not required when beans are configured using an XML file.
- b. @ComponentScan tag is not required when beans are configured using XML file.
- c. @ComponentScan annotation can be used in Java configuration for the autoscanning of beans.

- Only b
- Only c
- Only b and c
- All a,b and c

Reset

Save

Autoscan

1 2

```
-----  
}  
  
package com.mypack.demo2;  
  
@Component  
  
public class Address{  
  
-----  
}
```

Which of the below given statements is/are VALID in xml configuration to

- a) <context:component-scan base-package="com.mypack.demo1" />
- b) <context:component-scan base-package="com.mypack.demo2" />
- c) <context:component-scan base-package="com" />
- d) <context:component-scan base-package="com.mypack" />

- Only c) and d)
- Only a)
- Only b)
- Only c)

Reset

Save

Java Based Configuration

1

Consider the below Java configuration file:

```
@Configuration  
public class ConfigClass {  
    @Bean  
    public AddressService addrBean() {  
        AddressService addressService = new AddressService("B:  
        return addressService;  
    }  
}
```

What is the default bean name of the AddressService class in the above

- AddressService
- addressService
- addrBean
- addr

Reset

Save



Spring AOP

1

2

3

Below given is the Spring configuration in AppConfig.java :

```
@Configuration  
@ComponentScan("com.infy")  
//Line2  
public class AppConfig {  
}
```

The main method of the application includes the following statements:
AbstractApplicationContext context = new AnnotationConfigApplicationContext();
StudentService stu=(StudentService)context.getBean("studentService");
System.out.println(stu.getName());
Identify the CORRECT annotations to be used at Line1 and Line2 of the above code.

Before execution of the method
Jack

- @Component at Line1 and @EnableAspectJAutoProxy at Line2
- @EnableAspectJAutoProxy at Line1 and @Component at Line2
- @ComponentScan at Line1 and @Component at Line2
- @Configuration at Line1 and @EnableAspectJAutoProxy at Line2

Reset

Save

Spring AOP

1

2

3

```
if (n1==0) {  
    System.out.println(" '0' is not valid, c  
    n1 = 10;  
}  
if (n2==0){  
    System.out.println(" '0' is not valid,  
    n2=10;  
}  
return joinPoint.proceed(new Object[] { n1,  
});  
}
```

Assume the required configuration is already present in SpringConfig statements:

```
AbstractApplicationContext context = new AnnotationConfigAppl  
MyInterface sum = (MyInterface) context.getBean("calculate");  
int result = sum.divide(100, 0);  
System.out.println("Output is : = " + result);
```

What will be the output, when the above code is executed?

- Output is : = 0
- Runtime exception: Invalid poincut expression
- '0' is not valid, changing number to 10
- Output is : = 10
- Output is : = 10

Spring AOP

2

3

}

Assume the required configuration is already present in SpringConfiguration.java. The main statements:

```
AbstractApplicationContext context = new AnnotationConfigApplicationContext(SpringConfiguration.class);
Calculate sum = (Calculate) context.getBean("calculate");
System.out.println("Output is : = " + sum.add(100, 10));
System.out.println("Output is : = " + sum.sub(5.5, 2.0));
```

What will be the output, when the above code is executed?

- Output is : = 110
 Output is : = -3.5

Before Calling the Method.

- Output is : = 110
 Output is : = -3.5

Before Calling the Method.

- Output is : = 110
 Before Calling the Method.
Output is : = -3.5

Output is : = 110

- Before Calling the Method.
Output is : = -3.5

Reset

Save

Refactor Navigate Search Project Run Window Help

Quick Access

```

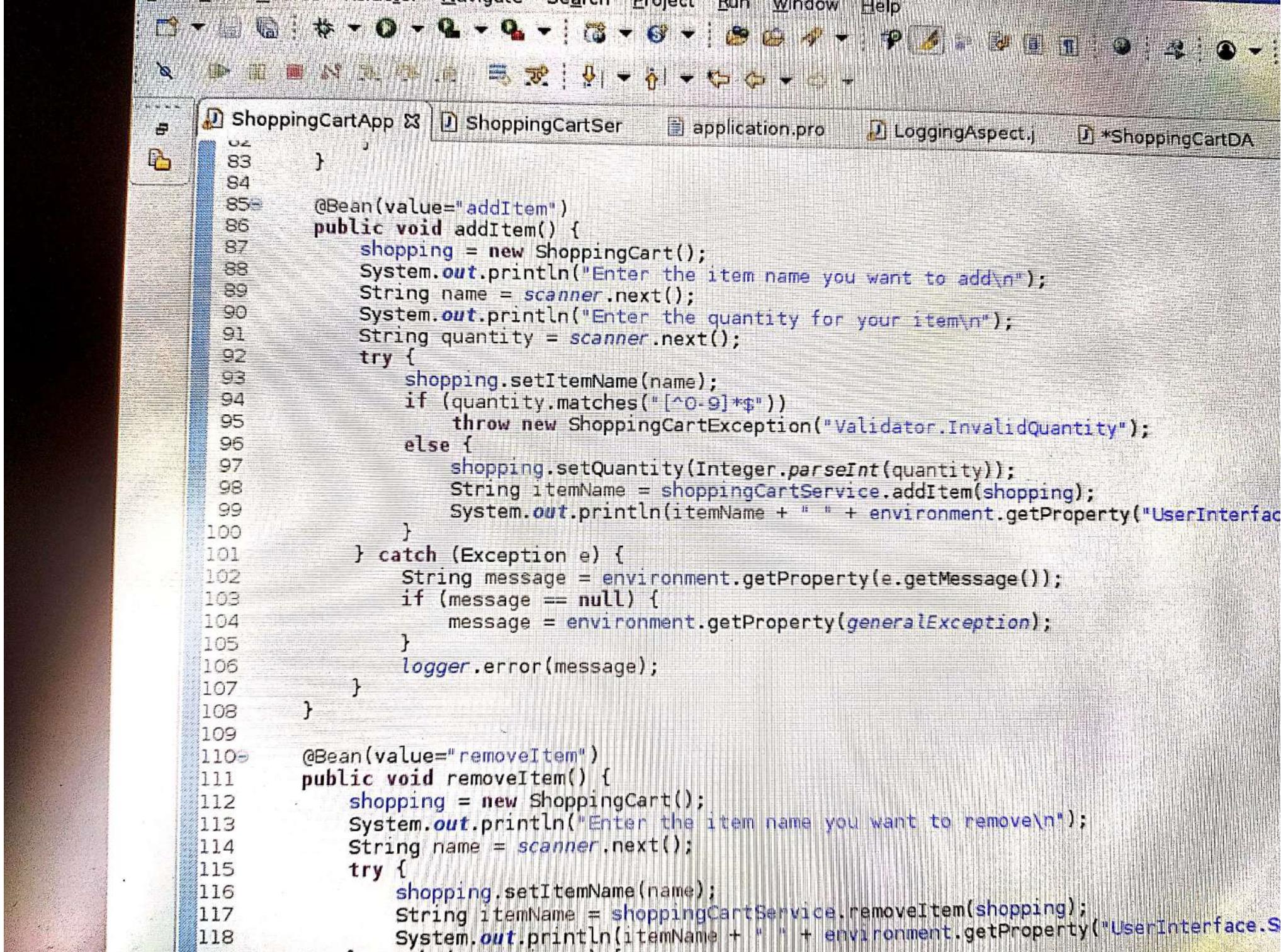
1 package com.infosys;
2
3+import java.util.Map;
4
5 //*****Do the required Spring configuration.*****
6 //*****Do not add any other extra instance variable or method.*****
7 @SpringBootApplication
8 @ComponentScan(basePackages="com.infosys")
9 @PropertySource("messages.properties")
10 public class ShoppingCartApplication implements CommandLineRunner {
11
12     @Autowired
13     private Environment environment;
14
15     @Autowired
16     private ShoppingCartService shoppingCartService;
17
18     private static String generalException = "General.Exception";
19
20     private static Logger logger = LoggerFactory.getLogger(ShoppingCartApplication.
21
22     private static Scanner scanner;
23
24     private ShoppingCart shopping;
25
26     public static void main(String[] args) {
27         SpringApplication.run(ShoppingCartApplication.class, args);
28     }

```

Markers Properties Servers Data Source Explorer Snippets

2 errors, 4 warnings, 0 others

Description	Resource	Path	Location	Type
Java Problems (4 items)				
Maven Configuration Problem (1 item)				



```
125      }
126  }
127
128  @Bean(value="updateDetails")
129  public void updateDetails() {
130      shopping = new ShoppingCart();
131      System.out.println("Enter the item name for which you want to update\n");
132      String name = scanner.next();
133      shopping.setItemName(name);
134      try {
135          System.out.println("Enter the quantity for the item\n");
136          String quantity = scanner.next();
137          if (quantity.matches("[^0-9]*$"))
138              throw new ShoppingCartException("Validator.InvalidQuantity");
139          else {
140              shopping.setQuantity(Integer.parseInt(quantity));
141              String n1 = shoppingCartService.updateDetails(shopping);
142              System.out.println(n1 + " " + environment.getProperty("UserInterface.UpdateCart"
143                                + shopping.getQuantity() + "\n");
144          }
145      } catch (Exception e) {
146          String message = environment.getProperty(e.getMessage());
147          if (message == null)
148              message = environment.getProperty(generalException);
149      }
150      logger.error(message);
151  }
152
153
154  @Bean(value="retrieveCart")
155  public void retrieveCart() {
156      shopping = new ShoppingCart();
157      try {
158          Map<String, Integer> cartDetails = shoppingCartService.retrieveCartDetails();
159          Set<String> cartDetailsSet = cartDetails.keySet();
160          System.out.println(environment.getProperty("UserInterface.RetrieveDetails"));
161          System.out.println("*****");
162          for (String s : cartDetailsSet) {
163              System.out.println(s + " " + cartDetails.get(s));
164          }
165      }
```



Quick Access

```
15 // do not add any other extra instances variables or methods.
16
17 @Service("shoppingCartService")
18 @Component
19 @Configuration
20 public class ShoppingCartServiceImpl implements ShoppingCartService {
21
22     @Autowired
23     private ShoppingCartDAO shoppingCartDao;
24
25     // *****Do not modify setShoppingCartDAO() method.*****
26     @Override
27     public void setShoppingCartDAO(ShoppingCartDAO shoppingCartDao) {
28         this.shoppingCartDao = shoppingCartDao;
29     }
30
31     @Override
32     @Bean
33     public String addItem(ShoppingCart shopping) throws ShoppingCartException {
34         Validator.isValidItemName(shopping.getItemName());
35         Validator.isValidQuantity(shopping.getQuantity());
36         if((Validator.isValidItemName(shopping.getItemName()))
37             && Validator.isValidQuantity(shopping.getQuantity())==true) {
38             shoppingCartDao.addItem(shopping);
39         }
40         return shopping.getItemName();
41     }
42
43     @Override
44     @Bean
45     public String removeItem(ShoppingCart shopping) throws ShoppingCartException {
46         Validator.isValidItemName(shopping.getItemName());
47         if(Validator.isValidItemName(shopping.getItemName())==false)
48             throw new ShoppingCartException("Service.CartError");
49         if(Validator.isValidItemName(shopping.getItemName())==true)
50             shoppingCartDao.removeItem(shopping.getItemName());
51         return shopping.getItemName();
52     }
53
54     @Override
55     @Bean
56     public String updateDetails(ShoppingCart shopping) throws ShoppingCartException {
```

```
38             shoppingCartDao.addItem(shopping);
39         }
40         return shopping.getItemName();
41     }
42
43     @Override
44     @Bean
45     public String removeItem(ShoppingCart shopping) throws ShoppingCartException {
46         Validator.isValidItemName(shopping.getItemName());
47         if(Validator.isValidItemName(shopping.getItemName())==false)
48             throw new ShoppingCartException("Service.CartError");
49         if(Validator.isValidItemName(shopping.getItemName())==true)
50             shoppingCartDao.removeItem(shopping.getItemName());
51         return shopping.getItemName();
52     }
53
54     @Override
55     @Bean
56     public String updateDetails(ShoppingCart shopping) throws ShoppingCartException {
57         Validator.isValidItemName(shopping.getItemName());
58         Validator.isValidQuantity(shopping.getQuantity());
59         if((Validator.isValidItemName(shopping.getItemName()))
60             && Validator.isValidQuantity(shopping.getQuantity())==false) {
61             throw new ShoppingCartException("Service.CartError");
62         }
63         if((Validator.isValidItemName(shopping.getItemName()))
64             && Validator.isValidQuantity(shopping.getQuantity())==true)
65             shoppingCartDao.updateDetails(shopping);
66         return shopping.getItemName();
67     }
68
69     @Override
70     @Bean
71     public Map<String, Integer> retrieveCartDetails() throws ShoppingCartException {
72         Map<String, Integer> shoppingCart=shoppingCartDao.retrieveCartDetails();
73         if(shoppingCart.isEmpty()) {
74             throw new ShoppingCartException("Service.NoProductsAvailable");
75         }
76         return shoppingCart;
77     }
```

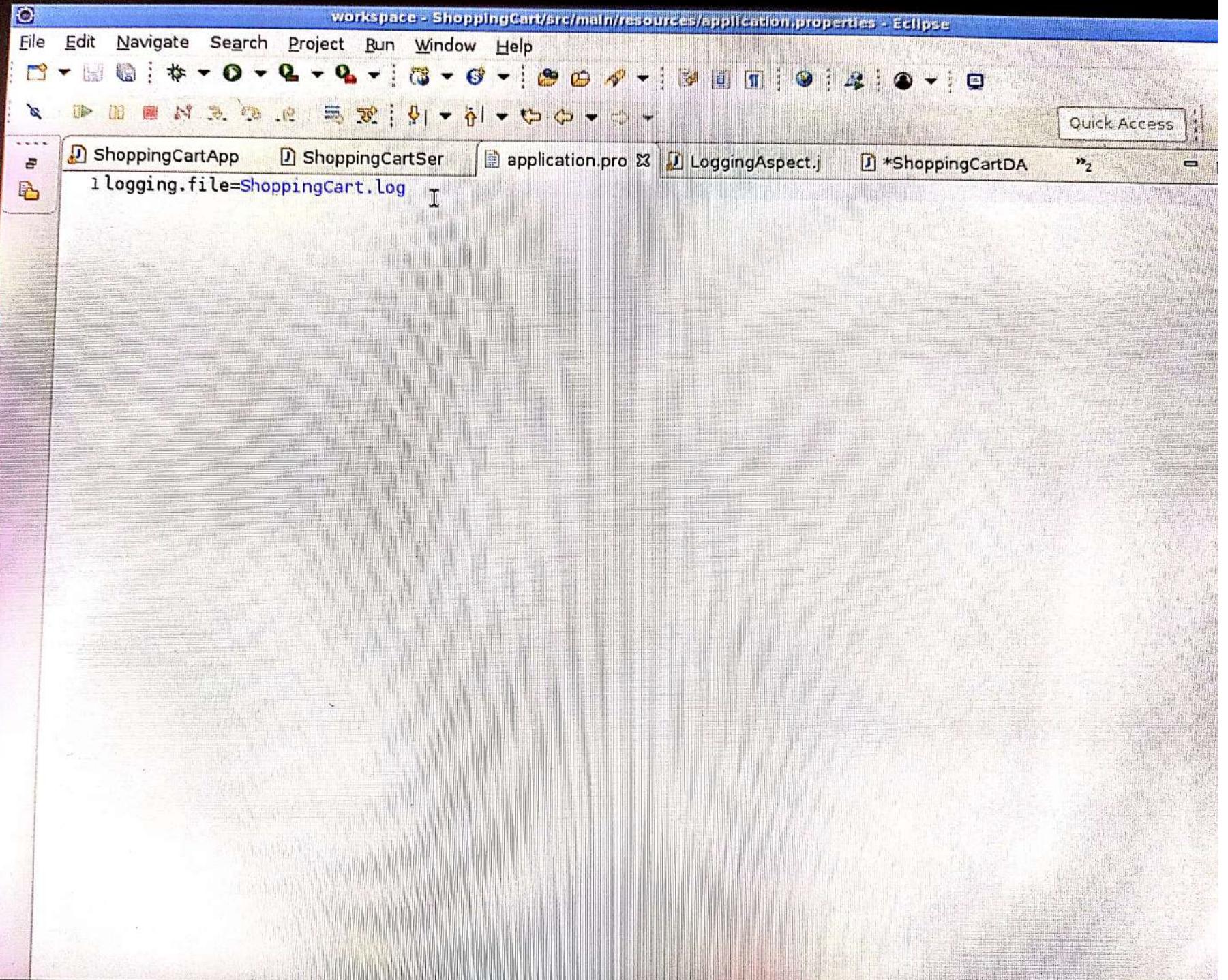
workspace - ShoppingCart/src/main/java/eta infosys/utility/LoggingAspect.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

ShoppingCartApp ShoppingCartSer application.pro LoggingAspect.j *ShoppingCartDA "2

```
1 package eta.infosys.utility;
2
3 import java.time.LocalDateTime;
4
5 @Component
6 @Aspect
7 public class LoggingAspect {
8     private static Logger logger = LoggerFactory.getLogger(LoggingAspect.class);
9
10    @Before("execution(* com.infosys.service.ShoppingCartServiceImpl.addItem(..))")
11    public void logBeforeAdvice() {
12        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm");
13        LocalDateTime now = LocalDateTime.now();
14        logger.info(now.toString(),formatter.toString());
15    }
16}
```



Result

Last modified on 2/0/2023 at 13:38:16

Module	Success	Failed	Total
ShoppingCartServiceTest	12	3	15
ShoppingCartConfigTest	10	2	12
Total	22	5	27

Close

```
42     private static Scanner scanner;
43
44     private ShoppingCart shopping;
45
46     public static void main(String[] args) {
47         SpringApplication.run(ShoppingCartApplication.class, args);
48     }
49
50     @Override
51     public void run(String... args) throws Exception {
52         scanner = new Scanner(System.in);
53         System.out.println("*****WELCOME TO INFYCART*****");
54     }
55 }
```

