

```
1 package com.infy.eventregistration.api;
2
3 import java.util.List;
4
5 import javax.validation.Valid;
6 import javax.validation.constraints.Pattern;
7
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.core.env.Environment;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.validation.annotation.Validated;
13 import org.springframework.web.bind.annotation.GetMapping;
14 import org.springframework.web.bind.annotation.PathVariable;
15 import org.springframework.web.bind.annotation.PostMapping;
16 import org.springframework.web.bind.annotation.RequestBody;
17 import org.springframework.web.bind.annotation.RequestMapping;
18 import org.springframework.web.bind.annotation.RestController;
19
20 import com.infy.eventregistration.dto.ParticipantDTO;
21 import com.infy.eventregistration.exception.EventRegistrationException;
22 import com.infy.eventregistration.service.EventService;
23
24 @RestController
25 @Validated
26 @RequestMapping(value="/event-api")
27 public class EventAPI {
28
29     @Autowired
30     private EventService eventService;
31
32     @Autowired
33     private Environment environment;
34
35     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
36
37     @PostMapping(value="/events")
38     public ResponseEntity<String> registerParticipant(@Valid @RequestBody ParticipantDTO participantDTO) throws EventRegistrationException {
39         // your code goes here
40     }
41 }
```

- EventRegistration/src/main/java/com/infy/eventregistration/api/

File Source Refactor Navigate Search Project Run Window Help

MYSSSET3JAVA-066

ParticipantDTO.java EventAPI.java

Quick Access

```
122 import com.infy.eventregistration.service.EventService;
123
124 @RestController
125 @Validated
126 @RequestMapping(value="/event-api")
127 public class EventAPI {
128
129     @Autowired
130     private EventService eventService;
131
132     @Autowired
133     private Environment environment;
134
135     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
136
137     @PostMapping(value="/events")
138     public ResponseEntity<String> registerParticipant(@Valid @RequestBody ParticipantDTO participantDTO) throws EventRegistrationException {
139         // your code goes here
140         Integer participantId=eventService.registerParticipant(participantDTO);
141         String successMessage=environment.getProperty("API.REGISTRATION_SUCCESS")+participantId;
142         return new ResponseEntity<>(successMessage,HttpStatus.CREATED);
143     }
144
145 }
146
147 // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
148
149     @GetMapping(value="/events/{venue}")
150     public ResponseEntity<List<ParticipantDTO>> getParticipantsByEventVenue(@PathVariable @Pattern(regexp="[A-Z][0-9][-](Hall)",message="{event.venue.invalid}") String venue)
151     // your code goes here
152
153     List<ParticipantDTO> participantDTO=eventService.getParticipantsByEventVenue(venue);
154
155     return new ResponseEntity<>(participantDTO,HttpStatus.OK);
156
157 }
158
159 }
```

Writable Smart Insert 60 : 1 : 2194

Search

16:25 07-12-2020 ENG

A screenshot of an IDE interface showing the code for `ParticipantDTO.java`. The code defines a class with various fields and annotations for validation. The IDE has a toolbar at the top, a left sidebar with icons, and a status bar at the bottom.

```
1 package com.infy.eventregistration.dto;
2
3 import java.time.LocalDate;
4
5 import javax.validation.Valid;
6 import javax.validation.constraints.NotNull;
7 import javax.validation.constraints.Pattern;
8
9 public class ParticipantDTO {
10
11     private Integer participantId;
12
13     @NotNull(message="{participant.name.notpresent}")
14     @Pattern(regexp="([A-Z][a-z]+)",message="{participant.name.invalid}")
15     private String name;
16
17     @NotNull(message="{participant.emailid.notpresent}")
18     @Pattern(regexp="[A-Za-z]+[0-9]+[@](infy.com)",message="{participant.emailid.invalid}")
19     private String emailId;
20
21     @NotNull(message="{participant.gender.notpresent}")
22     @Pattern(regexp="Female|Male|Others",message="{participant.gender.invalid}")
23     private String gender;
24
25     @NotNull(message="{participant.registrationdate.notpresent}")
26
27     private LocalDate registrationDate;
28
29     @NotNull(message="{participant.event.notpresent}")
30     @Valid
31     private EventDTO eventDTO;
32
33     public Integer getParticipantId() {
34         return participantId;
35     }
36
37     public void setParticipantId(Integer participantId) {
38         this.participantId = participantId;
39     }
40 }
```

The screenshot shows a Java IDE interface with the following details:

- Project Structure:** The left sidebar shows a project named "EventRegistration" with the following structure:
 - src/main/java:
 - com.infy.eventregistration.entity (selected)
 - com.infy.eventregistration.exception
 - com.infy.eventregistration.repository
 - com.infy.eventregistration.service
 - src/main/resources
 - src/test/java
 - Maven Dependencies
 - JRE System
 - log
- Code Editor:** The main window displays the `Event.java` file content. The code defines a class `Event` with fields `eventId`, `name`, `eventDate`, `venue`, and `maxCount`. It includes getters and setters for these fields.
- Console:** At the bottom, the console output shows "Finished after 16.516 seconds".

```
1 package com.infy.eventregistration.entity;
2
3+ import java.time.LocalDate;
4
5
6 @Entity
7 public class Event {
8
9     @Id
10    private Integer eventId;
11    private String name;
12    private LocalDate eventDate;
13    private String venue;
14    private Integer maxCount;
15
16    public Integer getEventId() {
17        return eventId;
18    }
19    public void setEventId(Integer eventId) {
20        this.eventId = eventId;
21    }
22    public String getName() {
23        return name;
24    }
25    public void setName(String name) {
26        this.name = name;
27    }
28    public LocalDate getEventDate() {
29        return eventDate;
30    }
31    public void setEventDate(LocalDate eventDate) {
32        this.eventDate = eventDate;
33    }
34    public String getVenue() {
35
36
37}
```

eventRegistration/src/main/java/com/infy/eventregistration/entity

source Refactor Navigate Search Project Run Window Help

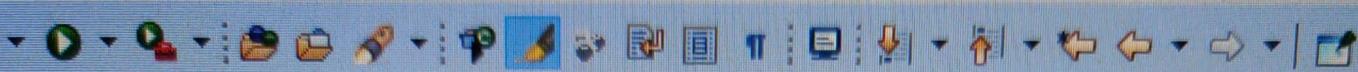
Event.java Participant.java

```
1 package com.infy.eventregistration.entity;
2
3+ import java.time.LocalDate;
4
5 @Entity
6 public class Participant {
7     @Id
8     @GeneratedValue(strategy=GenerationType.IDENTITY)
9     private Integer participantId;
10    private String name;
11    private String emailId;
12    private String gender;
13    private LocalDate registrationDate;
14
15    @ManyToOne(cascade=CascadeType.ALL)
16    @JoinColumn(name="event_id")
17    private Event event;
18
19    public Integer getParticipantId() {
20        return participantId;
21    }
22    public void setParticipantId(Integer participantId) {
23        this.participantId = participantId;
24    }
25    public String getName() {
26        return name;
27    }
28    public void setName(String name) {
29        this.name = name;
30    }
31    public String getEmailId() {
32        return emailId;
33    }
34
35}
```

Console JUnit

Finished after 16.516 seconds

Refactor Navigate Search Project Run Window Help



Event.java

EventRepository.java

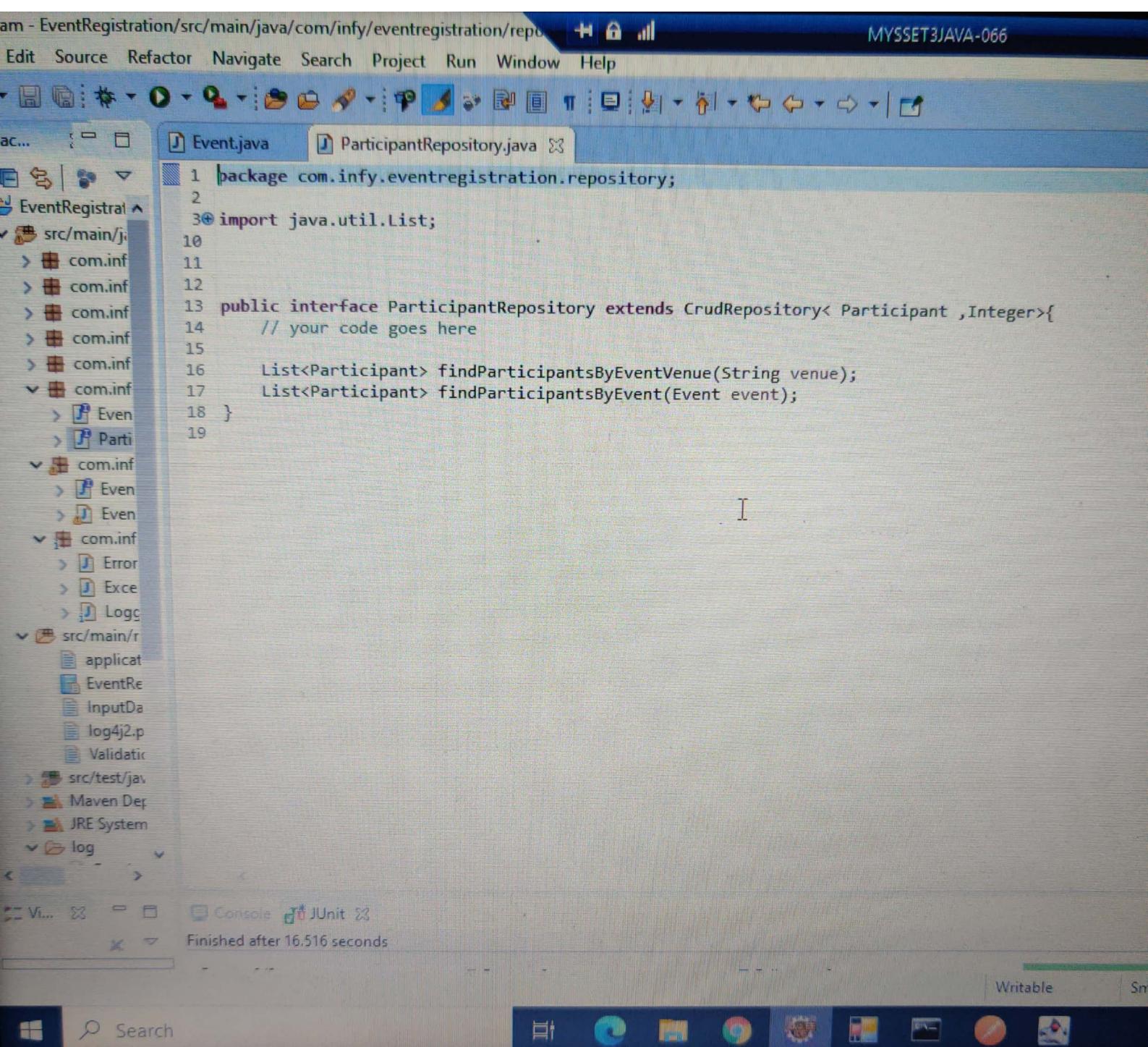
```
1 package com.infy.eventregistration.repository;
2
3+ import org.springframework.data.repository.CrudRepository;
4
5 public interface EventRepository extends CrudRepository< Event ,Integer> {
6     // your code goes here
7
8     Event findByName(String name);
9
10 }
11
12 }
```

Console JUnit
Finished after 16.516 seconds

Writable

Search





The screenshot shows an IDE interface with the following details:

- Title Bar:** exam - EventRegistration/src/main/java/com/infy/eventregistration/service
- Toolbar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Project Explorer:** Shows the project structure under "exam".
 - src/main/java:
 - com.infy.eventregistration.service (selected)
 - com.infy.eventregistration.dto
 - com.infy.eventregistration.entity
 - com.infy.eventregistration.exception
 - com.infy.eventregistration.repository
 - src/main/resources
 - src/test/java
 - Maven Dependencies
 - JRE System
 - log
- Code Editor:** The "EventServiceImpl.java" file is open.

```
1 package com.infy.eventregistration.service;
2
3
4 import java.util.ArrayList;
5 import java.util.List;
6
7
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.stereotype.Service;
10 import org.springframework.transaction.annotation.Transactional;
11
12 import com.infy.eventregistration.dto.EventDTO;
13 import com.infy.eventregistration.dto.ParticipantDTO;
14 import com.infy.eventregistration.entity.Event;
15 import com.infy.eventregistration.entity.Participant;
16 import com.infy.eventregistration.exception.EventRegistrationException;
17 import com.infy.eventregistration.repository.EventRepository;
18 import com.infy.eventregistration.repository.ParticipantRepository;
19
20 @Service(value="eventService")
21 @Transactional
22 public class EventServiceImpl implements EventService {
23
24     @Autowired
25     private EventRepository eventRepository;
26
27     @Autowired
28     private ParticipantRepository participantRepository;
29
30     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
31     @Override
32     public Integer registerParticipant(ParticipantDTO participantDTO) throws EventRegistrationException {
33         // your code goes here
34         Event event=eventRepository.findByName(participantDTO.getEventDTO().getName());
35     }
36 }
```
- Console Tab:** JUnit 32, Finished after 16.516 seconds.
- Status Bar:** Writable, Smart Insert, 3:1:47.

exam - EventRegistration/src/main/java/com/infy/eventregistration/service

File Edit Source Refactor Navigate Search Project Run Window Help

MYSSET3JAVA-066

Pac...

Event.java ParticipantRepository.java EventServiceImpl.java

```
31  @Override
32  public Integer registerParticipant(ParticipantDTO participantDTO) throws EventRegistrationException {
33      // your code goes here
34      Event event=participantRepository.findByName(participantDTO.getEventDTO().getName());
35      if(event==null)
36          throw new EventRegistrationException("Service.EVENT_UNAVAILABLE");
37      List<Participant> participantList=participantRepository.findParticipantsByEvent(event);
38      if(participantList.size()>event.getMaxCount())
39          throw new EventRegistrationException("Service.REGISTRATION_CLOSED");
40      for(Participant p:participantList)
41      {
42          if(p.getRegistrationDate().isAfter(event.getEventDate().minusDays(2)))
43              throw new EventRegistrationException("Service.REGISTRATION_CLOSED");
44      }
45      Participant participant=new Participant();
46      participant.setEmailId(participantDTO.getEmailId());
47      participant.setGender(participantDTO.getGender());
48      participant.setName(participantDTO.getName());
49      participant.setRegistrationDate(participantDTO.getRegistrationDate());
50      participant.setEvent(event);
51      participantRepository.save(participant);
52
53      return participant.getParticipantId();
54  }
55
56  // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
57  @Override
58  public List<ParticipantDTO> getParticipantsByEventVenue(String venue) throws EventRegistrationException {
59      // your code goes here
60      List<Participant> participantList=participantRepository.findParticipantsByEventVenue(venue);
61      if(participantList.isEmpty())
62          throw new EventRegistrationException("Service.PARTICIPANTS_UNAVAILABLE");
63      List<ParticipantDTO> participantDTOList=new ArrayList<>();
64  }
```

Console JUnit

EventRegistrationApplication [Java Application] C:\Program Files\Zulu\zulu-11\bin\javaw.exe (07-Dec-2020, 2:19:06 pm)

Writable Smart Insert 64:1:2554

Search

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** Event.java, ParticipantRepository.java, EventServiceImpl.java
- Left Sidebar (Project Explorer):** Shows the project structure under "EventRegistration". It includes packages like com.inf.event.registration, com.inf.event.registration.error, com.inf.event.registration.inputdata, and log. Under "src/main/java", there are subfolders "com.inf.event.registration" containing "Event.java", "ParticipantRepository.java", and "EventServiceImpl.java", and "src/main/resources" containing "application.properties", "EventRegistration.properties", "InputData.properties", "log4j2.xml", and "Validations.properties".
- Central Area (Code Editor):** Displays the content of `Event.java`. The code implements the `getParticipantsByEventVenue` method, which retrieves participants from a repository, converts them to DTOs, and sorts them by registration date.
- Bottom Status Bar:** Shows the console output: "EventRegistrationApplication [Java Application] C:\Program Files\Zulu\zulu-11\bin\javaw.exe (07-Dec-2020, 2:19:06 pm)". It also displays the current line (91), total lines (1359), and the status "Writable" and "Smart Insert".

```
58     @Override
59     public List<ParticipantDTO> getParticipantsByEventVenue(String venue) throws EventRegistrationException {
60         // your code goes here
61         List<Participant> participantList=participantRepository.findParticipantsByEventVenue(venue);
62         if(participantList.isEmpty())
63             throw new EventRegistrationException("Service.PARTICIPANTS_UNAVAILABLE");
64         List<ParticipantDTO> participantDTOList=new ArrayList<>();
65         for(Participant participant:participantList)
66         {
67             ParticipantDTO participantDTO=new ParticipantDTO();
68             participantDTO.setEmailId(participant.getEmailId());
69             participantDTO.setGender(participant.getGender());
70             participantDTO.setName(participant.getName());
71             participantDTO.setParticipantId(participant.getParticipantId());
72             participantDTO.setRegistrationDate(participant.getRegistrationDate());
73
74             EventDTO eventDTO=new EventDTO();
75             eventDTO.setDate(participant.getEvent().getEventDate());
76             eventDTO.setId(participant.getEvent().getEventId());
77             eventDTO.setMaxCount(participant.getEvent().getMaxCount());
78             eventDTO.setName(participant.getEvent().getName());
79             eventDTO.setVenue(venue);
80
81             participantDTO.setEventDTO(eventDTO);
82             participantDTOList.add(participantDTO);
83
84         }
85
86         participantDTOList.sort((p1,p2)->p2.getRegistrationDate().compareTo(p1.getRegistrationDate()));
87
88         return participantDTOList;
89     }
90
91 }
```

The screenshot shows a Java IDE interface with a code editor containing the following Java code:

```
2 import java.util.stream.Collectors;
3 import javax.validation.ConstraintViolation;
4 import javax.validation.ConstraintViolationException;
5
6 import org.apache.commons.logging.Log;
7 import org.apache.commons.logging.LogFactory;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.core.env.Environment;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.validation.ObjectError;
13 import org.springframework.web.bind.MethodArgumentNotValidException;
14 import org.springframework.web.bind.annotation.ExceptionHandler;
15 import org.springframework.web.bind.annotation.RestControllerAdvice;
16
17 import com.infy.eventregistration.exception.EventRegistrationException;
18
19 @RestControllerAdvice
20 public class ExceptionControllerAdvice {
21
22     private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);
23
24     @Autowired
25     private Environment environment;
26
27     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
28
29     @ExceptionHandler(EventRegistrationException.class)
30     public ResponseEntity<ErrorInfo> eventRegistrationExceptionHandler(EventRegistrationException exception) {
31
32
33
34 }
```

The code implements a `@RestControllerAdvice` annotation to handle exceptions. It uses `LogFactory` to get a logger named `LOGGER`. It also injects the `Environment` object. The `eventRegistrationExceptionHandler` method is annotated with `@ExceptionHandler(EventRegistrationException.class)` to handle instances of `EventRegistrationException`.

```
26
27
28     @Autowired
29     private Environment environment;
30
31     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
32
33     @ExceptionHandler(EventRegistrationException.class)
34     public ResponseEntity<ErrorInfo> eventRegistrationExceptionHandler(EventRegistrationException exception) {
35         LOGGER.error(exception.getMessage(), exception);
36         ErrorInfo errorInfo = new ErrorInfo();
37         errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
38         errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
39         return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
40     }
41
42     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
43     @ExceptionHandler(Exception.class)
44     public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
45         LOGGER.error(exception.getMessage(), exception);
46         ErrorInfo errorInfo = new ErrorInfo();
47         errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
48         errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
49         return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
50     }
51
52     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
53
54     @ExceptionHandler({MethodArgumentNotValidException.class, ConstraintViolationException .class})
55     public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
56         LOGGER.error(exception.getMessage(), exception);
57         String errorMsg;
58         if (exception instanceof MethodArgumentNotValidException) {
59             MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
```

The screenshot shows a Java IDE interface with the following details:

- Title Bar:** The title bar displays several tabs: Event.java, ParticipantRepository.java, EventServiceImpl.java, ExceptionControllerAdvice.java, and LoggingAspect.java.
- Code Editor:** The main window contains the source code for the LoggingAspect.java file. The code defines a class LoggingAspect with an @Aspect annotation. It includes a private static final Log field named LOGGER. A method logServiceException is annotated with @AfterThrowing, specifying a pointcut for methods in the com.infy.eventregistration.service.EventServiceImpl class that throw exceptions. The method logs the exception message using the LOGGER.error method.
- Toolbars and Menus:** Standard Java IDE toolbars and menus are visible at the top.
- Bottom Status Bar:** The status bar at the bottom shows "Writable", "Smart Insert", and the current line number "21 : 75 : 748".
- Console Tab:** The "Console" tab is open, showing the application name "EventRegistrationApplication [Java Application]" and the command line "C:\Program Files\Zulu\zulu-11\bin\javaw.exe (07-Dec-2020, 2:19:06 pm)".

File Edit Source Refactor Navigate Search Project Run Window Help

Event.java ParticipantRepository.java EventServiceImpl.java ExceptionControllerAdvice.java LoggingAspect.java EventRegistrationApplicationTests.java

```
1 package com.infy.eventregistration;
2
3 import java.time.LocalDate;
4
5
6 import org.junit.jupiter.api.Assertions;
7 import org.junit.jupiter.api.Test;
8 import org.mockito.InjectMocks;
9 import org.mockito.Mock;
10 import org.mockito.Mockito;
11 import org.springframework.boot.test.context.SpringBootTest;
12
13 import com.infy.eventregistration.dto.EventDTO;
14 import com.infy.eventregistration.dto.ParticipantDTO;
15 import com.infy.eventregistration.entity.Event;
16
17 import com.infy.eventregistration.exception.EventRegistrationException;
18 import com.infy.eventregistration.repository.EventRepository;
19 import com.infy.eventregistration.repository.ParticipantRepository;
20 import com.infy.eventregistration.service.EventService;
21 import com.infy.eventregistration.service.EventServiceImpl;
22
23
24 @SpringBootTest
25 public class EventRegistrationApplicationTests {
26
27     @Mock
28     private EventRepository eventRepository;
29
30     @Mock
31     private ParticipantRepository participantRepository;
32
33
34     @InjectMocks
35     private EventService eventService = new EventServiceImpl();
36
37     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
38     @Test
39     public void getParticipantsByEventVenueNoParticipantsFoundTest() {
40
41         Assertions.assertThrows(EventRegistrationException.class,
42             () -> eventService.getParticipantsByEventVenueNoParticipantsFound("abc"));
43     }
44 }
```

Writable Smart Insert 16 : 1 : 440

Search

exam - EventRegistration/src/test/java/com/infy/eventregistration/EventServiceTest.java MYSSET3JAVA-066

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

EventRegistrationApplicationTests.java EventServiceImpl.java

```
24
25 @SpringBootTest
26 public class EventRegistrationApplicationTests {
27
28     @Mock
29     private EventRepository eventRepository;
30
31     @Mock
32     private ParticipantRepository participantRepository;
33
34
35     @InjectMocks
36     private EventService eventService = new EventServiceImpl();
37
38     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
39     @Test
40     public void getParticipantsByEventVenueNoParticipantsFoundTest() {
41         // your code goes here
42
43         Event event=new Event();
44         event.setName("Ethical Hacking");
45         //event.setVenue("A6-HALL");
46
47         List<Participant> list=new ArrayList<>();
48         String venue="A6-HALL";
49
50         Mockito.when(participantRepository.findParticipantsByEventVenue(Mockito.anyString())).thenReturn(list);
51         EventRegistrationException exception=Assertions.assertThrows(EventRegistrationException.class, ()->eventService.getParticipantsByEventVenue(venue));
52         Assertions.assertEquals("Service.PARTICIPANTS_UNAVAILABLE", exception.getMessage());
53     }
54
55     // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD
56
57     @Test
58     public void registerParticipantNoEventFoundTest() {
59         // your code goes here
60
61         Event event=new Event();
62     }
}
```

Writable Smart Insert 45 : 37 : 1387

Search

Windows Taskbar icons: File Explorer, Edge, Google Chrome, File Manager, Task View, Taskbar Icons, Taskbar Icons, Taskbar Icons.

The screenshot shows an IDE interface with multiple tabs open at the top. The active tab is `EventRegistrationApplicationTests.java`. The code in the editor is a Java test class containing a single test method. The code uses Mockito for testing a repository and an event service.

```
50 }  
51 // DO NOT CHANGE METHOD SIGNATURE AND DELETE/COMMENT METHOD  
52  
53 @Test  
54 public void registerParticipantNoEventFoundTest() {  
55     // your code goes here  
56  
57     Event e=new Event();  
58     e.setEventDate(LocalDate.now().plusWeeks(2));  
59     e.setEventId(1234);  
60     e.setMaxCount(4);  
61     e.setName("Xsss");  
62     e.setVenue("A6-Hall");  
63  
64     ParticipantDTO participantDTO=new ParticipantDTO();  
65     participantDTO.setEmailId("abc.infy.com");  
66     participantDTO.setGender("Female");  
67     participantDTO.setName("Absc");  
68     participantDTO.setRegistrationDate(LocalDate.of(2020, 9, 21));  
69  
70     EventDTO event=new EventDTO();  
71     event.setEventDate(e.getEventDate());  
72     event.setEventId(e.getEventId());  
73     event.setMaxCount(e.getMaxCount());  
74     event.setName(e.getName());  
75     event.setVenue(e.getVenue());  
76  
77     participantDTO.setEventDTO(event);  
78  
79  
80     Mockito.when(eventRepository.findByName(Mockito.anyString())).thenReturn(null);  
81     EventRegistrationException exception=Assertions.assertThrows(EventRegistrationException.class, ()->eventService.registerParticipant(participantDTO));  
82     Assertions.assertEquals("Service.EVENT_UNAVAILABLE", exception.getMessage());  
83  
84  
85  
86 }  
87 }  
88 }
```

The status bar at the bottom right shows the time as 88:1:2770. The taskbar at the bottom has icons for File, Undo, Redo, Cut, Copy, Paste, Find, Replace, and others.