Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №8

Обмен сообщениями в Spring
тема

Преподаватель _____ А.С. Черниговский
подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031830504 _____ Е.В. Железкин
номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2020

1 Цель работы

Познакомиться с механизмом JMS в Spring.

2 Задача работы

Взять практическое задание №7 и добавить следующий функционал:

1) Настроить очередь (Для ActiveMQ или любого другого брокера сообщений JMS) приема сообщений для администратора.

2) При выполнении операций добавления, удаления или редактирования ресурса через REST API создавать соответствующие уведомления и отправлять их в очередь.

3) Любым удобным способом (можно через консоль) продемонстрировать извлечение административных сообщений о выполненных операциях (из пункта 2).

3 Ход работы

1) К проекту подключены новые зависимости(рисунок 1):

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-activemq</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jms</artifactId>
</dependency>
```

Рисунок 1 – Зависимости для JMS, pom.xml

2) Добавлены отправители уведомлений для очереди (рисунки 2-3):

```
public BooksController(@Qualifier("bookService") BookServiceImpl bookService, JmsTemplate jmsTemplate)
{
    this.bookService = bookService;
    this.jmsTemplate = jmsTemplate;
}
```

Рисунок 2 – Конфигурация JMS Template, BooksController.java

```
@PatchMapping("/{id}")
public String update(@ModelAttribute("book") @Valid Book book, BindingResult bindingResult, @PathVariable("id") int id)
{
    if (bindingResult.hasErrors())
        return "books/edit";
    bookService.updateById(id, book);
    jmsTemplate.convertAndSend( destinationName: "Edit Notification",  message: "Book " + id + " updated: " + book.toString());
    return "redirect:/books";
}

@DeleteMapping("/{id}")
public String delete(@PathVariable("id") int id)
{
    bookService.removeById(id);
    jmsTemplate.convertAndSend( destinationName: "Delete Notification",  message: "Book " + id + " deleted!");
    return "redirect:/books";
}
```

Рисунок 3 – Отправка уведомлений, BooksController.java

3) Взаимодействие с очередями сообщений(рисунки 4-8):

```
jvm 1    |  INFO | Connector ws started
jvm 1    |  INFO | Apache ActiveMQ 5.16.0 (localhost, ID:Head-56012-1608813503574-0:1) started
jvm 1    |  INFO | For help or more information please see: http://activemq.apache.org
jvm 1    |  INFO | ActiveMQ WebConsole available at http://127.0.0.1:8161/
jvm 1    |  INFO | ActiveMQ Jolokia REST API available at http://127.0.0.1:8161/api/jolokia/
```

Рисунок 4 – Запуск ActiveMQ сервера

Queues:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
Add Notification	0	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
Delete Notification	0	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
Edit Notification	0	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause

Рисунок 5 – Список очередей, описанных в spring приложении

```java
@Component
public class ActiveMQListener
{
    private static final Logger log = LoggerFactory.getLogger(WebApp.class);

    @JmsListener(destination = "Add Notification")
    public void addListener(String message)
    {
        log.info("Add notification: \"" + message + "\"");
    }

    @JmsListener(destination = "Edit Notification")
    public void editListener(String message) { log.info("Edit notification: \"" + message + "\""); }

    @JmsListener(destination = "Delete Notification")
    public void deleteListener(String message) { log.info("Delete notification: \"" + message + "\""); }
}
```

Рисунок 6 – Получение уведомлений с помощью @JmsListener, ActiveMQListener.java

```
INFO 14528 --- [enerContainer-1] ru.xorsiphus.WebApp                : Add notification: "New book added: Book{id=10, name='JmsTest', author='ttt', sizeInPages=555, printEdition='yyy'}"
INFO 14528 --- [enerContainer-1] ru.xorsiphus.WebApp                : Delete notification: "Book 9 deleted!"
INFO 14528 --- [enerContainer-1] ru.xorsiphus.WebApp                : Edit notification: "Book 4 updated: Book{id=4, name='Test', author='123', sizeInPages=123, printEdition='123'}"
```

Рисунок 7 – Вывод уведомлений

Queues:

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
Add Notification	0	1	2	2	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
Delete Notification	0	1	1	1	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause
Edit Notification	0	1	1	1	Browse Active Consumers Active Producers atom rss	Send To Purge Delete Pause

Рисунок 8 – Статистика очередей

4 Вывод

В ходе данной лабораторной работы было проведено ознакомление с механизмом JMS в Spring.