Spring Boot 入门（十一）：集成 WebSocket, 实时显示系统日志

以前面的博客为基础，最近一篇为Spring Boot 入门（十）：集成Redis哨兵模式、实现Mybatis二级缓存。本篇博客主要介绍了Spring Boot集成 Web Socket进行日志的推送，并实时显示在页面上。

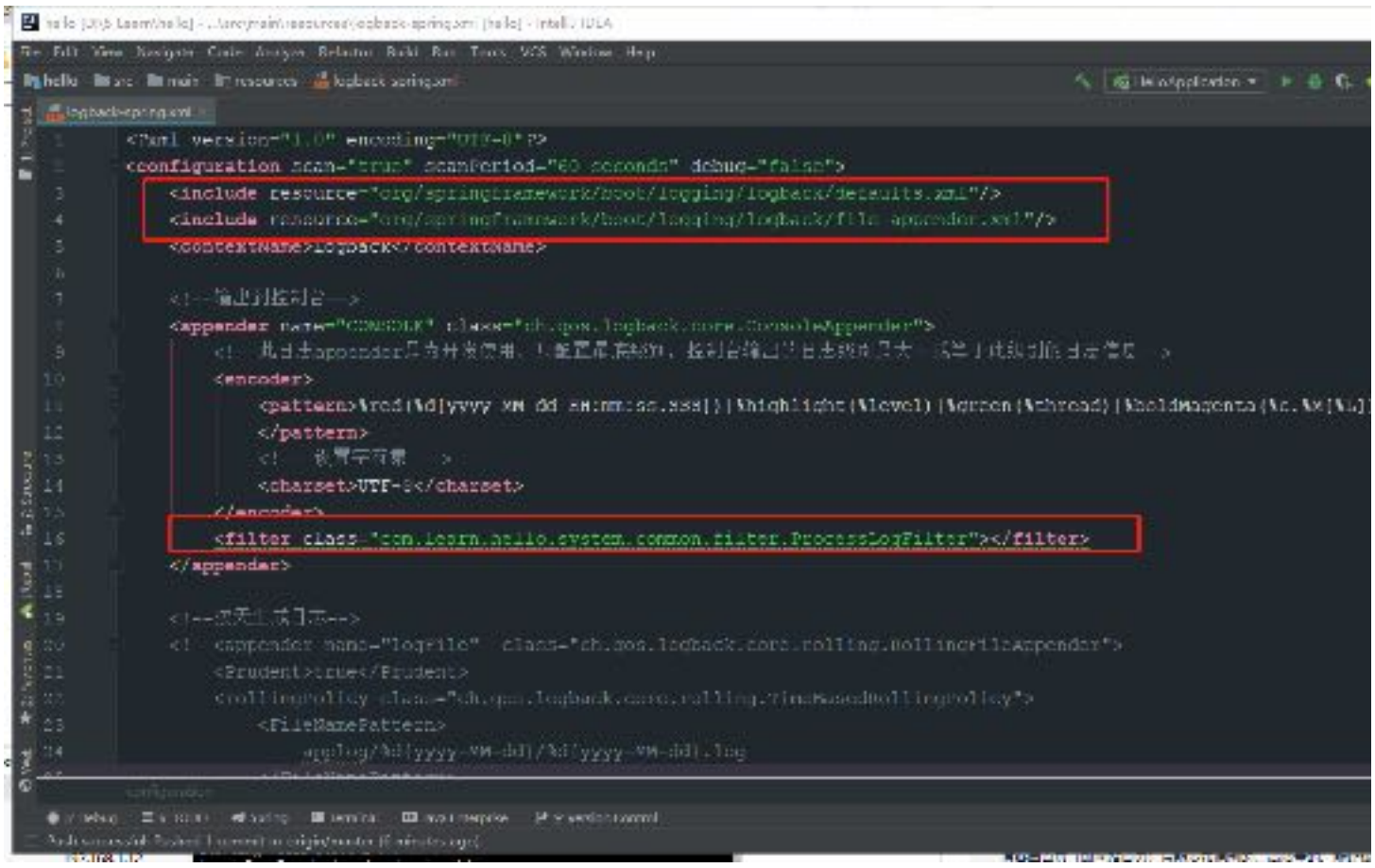
1.导入jar包

第一个jar包是websocket的，第二个jar包是关于环形队列的jar包，本案例是通过本地队列存储日志。有条件的话，最好通过中间件存储（eg：redis，mq……）。通过本地队列存储日志会存在日志丢失的情况，且日志量太大，会把页面卡死。

```
<!--begin web socket-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-websocket</artifactId>
        </dependency>
        <dependency>
            <groupId>com.lmax</groupId>
            <artifactId>disruptor</artifactId>
            <version>3.4.2</version>
        </dependency>
        <!--end web socket-->
```

2.增加监听器

（1）.在logback中增加监听器



并根据logback编写相应的监听器ProcessLogFilter

```
@Service
public class ProcessLogFilter extends Filter<ILoggingEvent> {

    @Override
    public FilterReply decide(ILoggingEvent event) {
        LoggerMessage loggerMessage = new LoggerMessage(
                event.getMessage()
                , DateFormat.getDateTimeInstance().format(new Date(event.getTimeStamp())),
                event.getThreadName(),
                event.getLoggerName(),
                event.getLevel().levelStr
        );
        LoggerDisruptorQueue.publishEvent(loggerMessage);
        return FilterReply.ACCEPT;
    }
}
```

该监听器将监听的日志消息推送到本地消息队列中，然后页面通过 Web Socket 去此队列获取日志信息，从而在页面显示

（2）.编写日志处理器

```
//进程日志事件内容载体
@Data
@NoArgsConstructor
@AllArgsConstructor
public class LoggerEvent {
    private LoggerMessage log;
}
```

```
/**
 * Content :进程日志事件工厂类
 */
public class LoggerEventFactory implements EventFactory<LoggerEvent> {
    @Override
    public LoggerEvent newInstance() {
        return new LoggerEvent();
    }
}
```

```
/**
 * Content :进程日志事件处理器
 */
@Component
public class LoggerEventHandler implements EventHandler<LoggerEvent> {

    @Autowired
    private SimpMessagingTemplate messagingTemplate;

    @Override
    public void onEvent(LoggerEvent stringEvent, long l, boolean b) {
        messagingTemplate.convertAndSend("/topic/pullLogger", stringEvent.getLog());
    }
}
```

日志事件处理器的作用是监听本地环形队列中的消息，如果有消息，就会将这些消息推送到 Socket 管道中

（3）.编写页面

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>欢迎页</title>
    <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no"
name="viewport">
    <script src="plugins/jQuery/jquery-2.2.3.min.js"></script>
    <script src="js/websocket/sockjs.min.js"></script>
    <script src="js/websocket/stomp.min.js"></script>
</head>
<body>
<div class="panel panel-default">
    <h1>jvm进程内的日志</h1>
    <button onclick="openSocket()">开启日志</button>
    <button onclick="closeSocket()">关闭日志</button>
    <div id="log-container" style="height: 600px; overflow-y: scroll; background: #333; color:
#aaa; padding: 10px;">
        <div></div>
    </div>
</div>
<script>
    var stompClient = null;
    $(document).ready(function () {
        openSocket();
    });

    function openSocket() {
        if (stompClient == null) {
            var socket = new SockJS('http://localhost:8080/websocket?token=kl');
            stompClient = Stomp.over(socket);
```

```
            stompClient.connect({token: "kl"}, function (frame) {
                stompClient.subscribe('/topic/pullLogger', function (event) {
                    var content = JSON.parse(event.body);
                    $("#log-container div").append("<font color='red'>" + content.timestamp + "
</font>|<font color='highlight'>" + content.level + "</font> |<font color='green'>" +
content.threadName + "</font>| <font color='boldMagenta'>" + content.className + "</font>|<font
color='cyan'>" + content.body + "</font>").append("<br/>");
                    $("#log-container").scrollTop($("#log-container div").height() - $("#log-
container").height());
                }, {
                    token: "kltoen"
                });
            });
        }
    }

    function closeSocket() {
        if (stompClient != null) {
            stompClient.disconnect();
            stompClient = null;
        }
    }
</script>
</body>
</html>
```

页面链接web Socket服务器，如果有消息，就能获取

（4）.其他辅助类

环形本地队列类

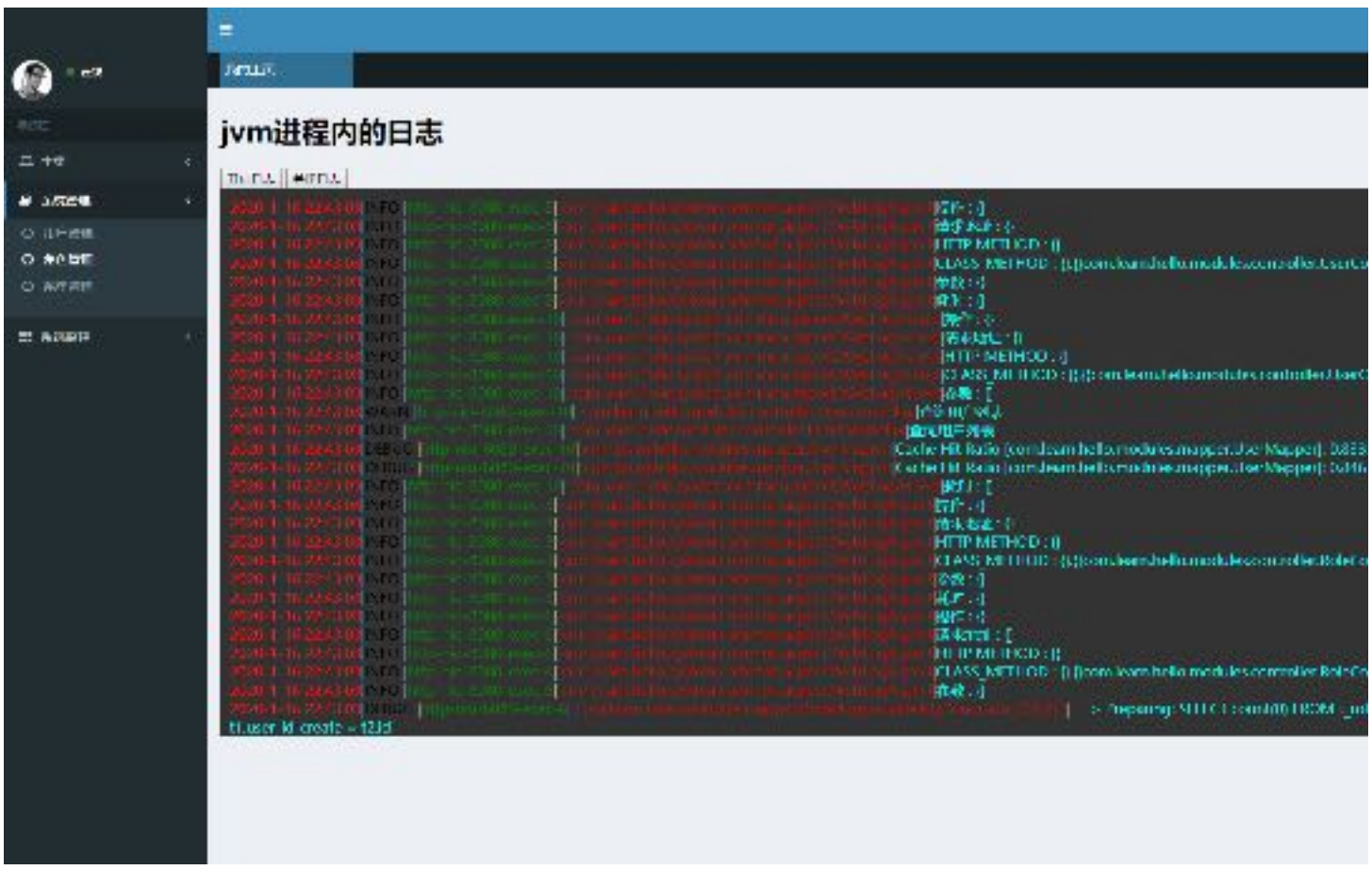
```
package com.learn.hello.system.common.queue;

import com.learn.hello.modules.entity.LoggerMessage;
import com.learn.hello.system.common.event.LoggerEvent;
import com.learn.hello.system.common.event.LoggerEventFactory;
import com.learn.hello.system.common.event.LoggerEventHandler;
import com.lmax.disruptor.RingBuffer;
import com.lmax.disruptor.dsl.Disruptor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.util.concurrent.Executor;
import java.util.concurrent.Executors;

/**
 * Content :Disruptor 环形队列
 */
@Component
public class LoggerDisruptorQueue {

    private Executor executor = Executors.newCachedThreadPool();

    // The factory for the event
    private LoggerEventFactory factory = new LoggerEventFactory();


    // Specify the size of the ring buffer, must be power of 2.
    private int bufferSize = 2 * 1024;

    // Construct the Disruptor
    private Disruptor<LoggerEvent> disruptor = new Disruptor<>(factory, bufferSize, executor);
    ;


    private static RingBuffer<LoggerEvent> ringBuffer;


    @Autowired
    LoggerDisruptorQueue(LoggerEventHandler eventHandler) {
        disruptor.handleEventsWith(eventHandler);
        this.ringBuffer = disruptor.getRingBuffer();
        disruptor.start();
    }

    public static void publishEvent(LoggerMessage log) {
        long sequence = ringBuffer.next();  // Grab the next sequence
        try {
            LoggerEvent event = ringBuffer.get(sequence); // Get the entry in the Disruptor
            // for the sequence
```

```
50              event.setLog(log);  // Fill with data
51          } finally {
52              ringBuffer.publish(sequence);
53          }
54      }
55
56  }
```

消息实体类

```
 1  package com.learn.hello.modules.entity;
 2
 3  import lombok.AllArgsConstructor;
 4  import lombok.Data;
 5  import lombok.NoArgsConstructor;
 6
 7  // 日志实体类
 8  @Data
 9  @AllArgsConstructor
10  @NoArgsConstructor
11  public class LoggerMessage {
12      private String body;
13      private String timestamp;
14      private String threadName;
15      private String className;
16      private String level;
17  }
```

3.效果



页面中的颜色可以自行设置

.

分类: Spring Boot

标签: web socket, 日志, spring boot

« 上一篇: Spring Boot 入门（十）：集成Redis哨兵模式，实现Mybatis二级缓存

posted @ 2020-01-16 23:19 光头才能强 阅读(82) 评论(0) 编辑 收藏