C++发布订阅者模式：实现简单消息传递系统

作者：微信公众号：【架构师老卢】
12-10 15:44

概述：这个C++示例演示了发布者-订阅者模式的基本实现。通过 `Event` 类，发布者 `Publisher` 发送数据，而订阅者 `Subscriber` 订阅并处理数据。通过简单的回调机制，实现了组件间松散耦合的消息传递。

好的，我将为你提供一个简单的C++实例，演示如何使用发布者-订阅者模式。在这个例子中，我们将使用C++11的标准库中的 `<functional>` 头文件，以及线程支持。

首先，我们定义一个简单的事件类，该事件类将用于携带消息：

```cpp
// Event.h
#pragma once

#include <functional>

template <typename... Args>
class Event {
public:
    using Callback = std::function<void(Args...)>;

    void subscribe(Callback callback) {
        callbacks_.emplace_back(std::move(callback));
    }

    void notify(Args... args) const {
        for (const auto& callback : callbacks_) {
            callback(args...);
        }
    }

private:
    std::vector<Callback> callbacks_;
};
```

接下来，我们定义一个发布者类，它将包含一个事件对象，并提供一个方法来触发该事件：

```cpp
// Publisher.h
#pragma once

#include "Event.h"

class Publisher {
public:
    Event<int> onDataReceived;

    void sendData(int data) {
        // 假设在这里进行一些数据处理
        onDataReceived.notify(data);
    }
};
```

然后，我们定义一个订阅者类，它将订阅发布者的事件并定义处理函数：

```cpp
// Subscriber.h
#pragma once

#include "Event.h"
#include <iostream>

class Subscriber {
public:
    void processData(int data) {
        std::cout << "Received data: " << data << std::endl;
    }
};
```

最后，我们将创建一个主函数来演示发布者和订阅者的使用：

```cpp
// main.cpp
#include "Publisher.h"
#include "Subscriber.h"
#include <thread>

int main() {
    Publisher publisher;
    Subscriber subscriber;

    // 订阅者订阅发布者的事件
    publisher.onDataReceived.subscribe([&subscriber](int data) {
        subscriber.processData(data);
    });

    // 模拟数据发送
    for (int i = 1; i <= 5; ++i) {
        publisher.sendData(i);
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }

    return 0;
}
```

在这个例子中，我们创建了一个简单的发布者 `Publisher` 类，它包含一个 `Event` 对象，该对象具有整数参数类型。订阅者 `Subscriber` 类定义了一个处理函数 `processData`，该函数将在收到数据时被调用。在主函数中，我们创建了发布者和订阅者的实例，并通过调用 `onDataReceived.subscribe` 将订阅者订阅到发布者的事件。然后，我们通过调用 `sendData` 模拟发布者发送数据，订阅者的处理函数将被调用。

这只是一个简单的示例，实际应用中可能需要更复杂的实现，以处理多个事件和更多的数据类型。