// Copyright 2013 The Flutter Authors. All rights reserved.

// Use of this source code is governed by a BSD-style license that can be

// found in the LICENSE file.

#ifndef FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_EVENT\_STREAM\_HANDLER\_FUNCTIONS\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_EVENT\_STREAM\_HANDLER\_FUNCTIONS\_H\_

#include <memory>

#include "event\_sink.h"

#include "event\_stream\_handler.h"

namespace flutter {

class EncodableValue;

// Handler types for each of the StreamHandler setup and teardown

// requests.

template <typename T>

using StreamHandlerListen =

std::function<std::unique\_ptr<StreamHandlerError<T>>(

const T\* arguments,

std::unique\_ptr<EventSink<T>>&& events)>;

template <typename T>

using StreamHandlerCancel =

std::function<std::unique\_ptr<StreamHandlerError<T>>(const T\* arguments)>;

// An implementation of StreamHandler that pass calls through to

// provided function objects.

template <typename T = EncodableValue>

class StreamHandlerFunctions : public StreamHandler<T> {

public:

// Creates a handler object that calls the provided functions

// for the corresponding StreamHandler outcomes.

StreamHandlerFunctions(StreamHandlerListen<T> on\_listen,

StreamHandlerCancel<T> on\_cancel)

: on\_listen\_(on\_listen), on\_cancel\_(on\_cancel) {}

virtual ~StreamHandlerFunctions() = default;

// Prevent copying.

StreamHandlerFunctions(StreamHandlerFunctions const&) = delete;

StreamHandlerFunctions& operator=(StreamHandlerFunctions const&) = delete;

protected:

// |flutter::StreamHandler|

std::unique\_ptr<StreamHandlerError<T>> OnListenInternal(

const T\* arguments,

std::unique\_ptr<EventSink<T>>&& events) override {

if (on\_listen\_) {

return on\_listen\_(arguments, std::move(events));

}

auto error = std::make\_unique<StreamHandlerError<T>>(

"error", "No OnListen handler set", nullptr);

return std::move(error);

}

// |flutter::StreamHandler|

std::unique\_ptr<StreamHandlerError<T>> OnCancelInternal(

const T\* arguments) override {

if (on\_cancel\_) {

return on\_cancel\_(arguments);

}

auto error = std::make\_unique<StreamHandlerError<T>>(

"error", "No OnCancel handler set", nullptr);

return std::move(error);

}

StreamHandlerListen<T> on\_listen\_;

StreamHandlerCancel<T> on\_cancel\_;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_EVENT\_STREAM\_HANDLER\_FUNCTIONS\_H\_