// 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\_ENGINE\_METHOD\_RESULT\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_ENGINE\_METHOD\_RESULT\_H\_

#include <memory>

#include <string>

#include <vector>

#include "binary\_messenger.h"

#include "method\_codec.h"

#include "method\_result.h"

namespace flutter {

namespace internal {

// Manages the one-time sending of response data. This is an internal helper

// class for EngineMethodResult, separated out since the implementation doesn't

// vary based on the template type.

class ReplyManager {

public:

explicit ReplyManager(BinaryReply reply\_handler\_);

~ReplyManager();

// Prevent copying.

ReplyManager(ReplyManager const&) = delete;

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

// Sends the given response data (which must either be nullptr, which

// indicates an unhandled method, or a response serialized with |codec\_|) to

// the engine.

void SendResponseData(const std::vector<uint8\_t>\* data);

private:

BinaryReply reply\_handler\_;

};

} // namespace internal

// Implemention of MethodResult that sends a response to the Flutter engine

// exactly once, encoded using a given codec.

template <typename T>

class EngineMethodResult : public MethodResult<T> {

public:

// Creates a result object that will send results to |reply\_handler|, encoded

// using |codec|. The |codec| pointer must remain valid for as long as this

// object exists.

EngineMethodResult(BinaryReply reply\_handler, const MethodCodec<T>\* codec)

: reply\_manager\_(

std::make\_unique<internal::ReplyManager>(std::move(reply\_handler))),

codec\_(codec) {}

~EngineMethodResult() = default;

protected:

// |flutter::MethodResult|

void SuccessInternal(const T\* result) override {

std::unique\_ptr<std::vector<uint8\_t>> data =

codec\_->EncodeSuccessEnvelope(result);

reply\_manager\_->SendResponseData(data.get());

}

// |flutter::MethodResult|

void ErrorInternal(const std::string& error\_code,

const std::string& error\_message,

const T\* error\_details) override {

std::unique\_ptr<std::vector<uint8\_t>> data =

codec\_->EncodeErrorEnvelope(error\_code, error\_message, error\_details);

reply\_manager\_->SendResponseData(data.get());

}

// |flutter::MethodResult|

void NotImplementedInternal() override {

reply\_manager\_->SendResponseData(nullptr);

}

private:

std::unique\_ptr<internal::ReplyManager> reply\_manager\_;

const MethodCodec<T>\* codec\_;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_COMMON\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_ENGINE\_METHOD\_RESULT\_H\_