// 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\_METHOD\_RESULT\_H\_

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

#include <string>

namespace flutter {

class EncodableValue;

// Encapsulates a result returned from a MethodCall. Only one method should be

// called on any given instance.

template <typename T = EncodableValue>

class MethodResult {

public:

MethodResult() = default;

virtual ~MethodResult() = default;

// Prevent copying.

MethodResult(MethodResult const&) = delete;

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

// Sends a success response, indicating that the call completed successfully

// with the given result.

void Success(const T& result) { SuccessInternal(&result); }

// Sends a success response, indicating that the call completed successfully

// with no result.

void Success() { SuccessInternal(nullptr); }

// Sends an error response, indicating that the call was understood but

// handling failed in some way.

//

// error\_code: A string error code describing the error.

// error\_message: A user-readable error message.

// error\_details: Arbitrary extra details about the error.

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

const std::string& error\_message,

const T& error\_details) {

ErrorInternal(error\_code, error\_message, &error\_details);

}

// Sends an error response, indicating that the call was understood but

// handling failed in some way.

//

// error\_code: A string error code describing the error.

// error\_message: A user-readable error message (optional).

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

const std::string& error\_message = "") {

ErrorInternal(error\_code, error\_message, nullptr);

}

// Sends a not-implemented response, indicating that the method either was not

// recognized, or has not been implemented.

void NotImplemented() { NotImplementedInternal(); }

protected:

// Implementation of the public interface, to be provided by subclasses.

virtual void SuccessInternal(const T\* result) = 0;

// Implementation of the public interface, to be provided by subclasses.

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

const std::string& error\_message,

const T\* error\_details) = 0;

// Implementation of the public interface, to be provided by subclasses.

virtual void NotImplementedInternal() = 0;

};

} // namespace flutter

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