// 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\_CALL\_H\_

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

#include <memory>

#include <string>

namespace flutter {

class EncodableValue;

// An object encapsulating a method call from Flutter whose arguments are of

// type T.

template <typename T = EncodableValue>

class MethodCall {

public:

// Creates a MethodCall with the given name and arguments.

MethodCall(const std::string& method\_name, std::unique\_ptr<T> arguments)

: method\_name\_(method\_name), arguments\_(std::move(arguments)) {}

virtual ~MethodCall() = default;

// Prevent copying.

MethodCall(MethodCall<T> const&) = delete;

MethodCall& operator=(MethodCall<T> const&) = delete;

// The name of the method being called.

const std::string& method\_name() const { return method\_name\_; }

// The arguments to the method call, or NULL if there are none.

const T\* arguments() const { return arguments\_.get(); }

private:

std::string method\_name\_;

std::unique\_ptr<T> arguments\_;

};

} // namespace flutter

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