// 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\_WINDOWS\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_DART\_PROJECT\_H\_

#define FLUTTER\_SHELL\_PLATFORM\_WINDOWS\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_DART\_PROJECT\_H\_

#include <string>

#include <vector>

namespace flutter {

// A set of Flutter and Dart assets used to initialize a Flutter engine.

class DartProject {

public:

// Creates a DartProject from a series of absolute paths.

// The three paths are:

// - assets\_path: Path to the assets directory as built by the Flutter tool.

// - icu\_data\_path: Path to the icudtl.dat file.

// - aot\_library\_path: Path to the AOT snapshot file.

//

// The paths may either be absolute or relative to the directory containing

// the running executable.

explicit DartProject(const std::wstring& assets\_path,

const std::wstring& icu\_data\_path,

const std::wstring& aot\_library\_path) {

assets\_path\_ = assets\_path;

icu\_data\_path\_ = icu\_data\_path;

aot\_library\_path\_ = aot\_library\_path;

}

// Creates a DartProject from a directory path. The directory should contain

// the following top-level items:

// - icudtl.dat (provided as a resource by the Flutter tool)

// - flutter\_assets (as built by the Flutter tool)

// - app.so, for an AOT build (as built by the Flutter tool)

//

// The path can either be absolute, or relative to the directory containing

// the running executable.

explicit DartProject(const std::wstring& path) {

assets\_path\_ = path + L"\\flutter\_assets";

icu\_data\_path\_ = path + L"\\icudtl.dat";

aot\_library\_path\_ = path + L"\\app.so";

}

~DartProject() = default;

// Sets the Dart entrypoint to the specified value.

//

// If not set, the default entrypoint (main) is used. Custom Dart entrypoints

// must be decorated with `@pragma('vm:entry-point')`.

void set\_dart\_entrypoint(const std::string& entrypoint) {

if (entrypoint.empty()) {

return;

}

dart\_entrypoint\_ = entrypoint;

}

// Returns the Dart entrypoint.

const std::string& dart\_entrypoint() const { return dart\_entrypoint\_; }

// Sets the command line arguments that should be passed to the Dart

// entrypoint.

void set\_dart\_entrypoint\_arguments(std::vector<std::string> arguments) {

dart\_entrypoint\_arguments\_ = std::move(arguments);

}

// Returns any command line arguments that should be passed to the Dart

// entrypoint.

const std::vector<std::string>& dart\_entrypoint\_arguments() const {

return dart\_entrypoint\_arguments\_;

}

private:

// Accessors for internals are private, so that they can be changed if more

// flexible options for project structures are needed later without it

// being a breaking change. Provide access to internal classes that need

// them.

friend class FlutterEngine;

friend class FlutterViewController;

friend class DartProjectTest;

const std::wstring& assets\_path() const { return assets\_path\_; }

const std::wstring& icu\_data\_path() const { return icu\_data\_path\_; }

const std::wstring& aot\_library\_path() const { return aot\_library\_path\_; }

// The path to the assets directory.

std::wstring assets\_path\_;

// The path to the ICU data.

std::wstring icu\_data\_path\_;

// The path to the AOT library. This will always return a path, but non-AOT

// builds will not be expected to actually have a library at that path.

std::wstring aot\_library\_path\_;

// The Dart entrypoint to launch.

std::string dart\_entrypoint\_;

// The list of arguments to pass through to the Dart entrypoint.

std::vector<std::string> dart\_entrypoint\_arguments\_;

};

} // namespace flutter

#endif // FLUTTER\_SHELL\_PLATFORM\_WINDOWS\_CLIENT\_WRAPPER\_INCLUDE\_FLUTTER\_DART\_PROJECT\_H\_