//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#pragma once

namespace Memory

{

#ifdef PROFILE\_RECYCLER\_ALLOC

#ifdef RECYCLER\_DUMP\_OBJECT\_GRAPH

class RecyclerObjectDumper

{

public:

typedef bool (\*DumpFunction)(type\_info const \* typeinfo, bool isArray, void \* objectAddress);

static void RegisterDumper(type\_info const \* typeinfo, DumpFunction dumperFunction);

static void DumpObject(type\_info const \* typeinfo, bool isArray, void \* objectAddress);

private:

RecyclerObjectDumper() {}

~RecyclerObjectDumper();

static RecyclerObjectDumper Instance;

static BOOL EnsureDumpFunctionMap();

typedef JsUtil::BaseDictionary<type\_info const \*, RecyclerObjectDumper::DumpFunction, NoCheckHeapAllocator> DumpFunctionMap;

static DumpFunctionMap \* dumpFunctionMap;

};

template <typename T, RecyclerObjectDumper::DumpFunction dumpFunction>

class AutoRegisterRecyclerObjectDumper

{

public:

static AutoRegisterRecyclerObjectDumper Instance;

private:

AutoRegisterRecyclerObjectDumper()

{

RecyclerObjectDumper::RegisterDumper(&typeid(T), dumpFunction);

}

};

template <typename T, RecyclerObjectDumper::DumpFunction dumpFunction>

AutoRegisterRecyclerObjectDumper<T, dumpFunction> AutoRegisterRecyclerObjectDumper<T, dumpFunction>::Instance;

#define AUTO\_REGISTER\_RECYCLER\_OBJECT\_DUMPER(T, func) template AutoRegisterRecyclerObjectDumper<T, func>;

#else

#define AUTO\_REGISTER\_RECYCLER\_OBJECT\_DUMPER(T, func)

#endif

void DumpRecyclerObjectGraph();

#else

#define AUTO\_REGISTER\_RECYCLER\_OBJECT\_DUMPER(T, func)

#endif

}

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#include "CommonMemoryPch.h"

#ifdef PROFILE\_RECYCLER\_ALLOC

// Initialization order

// AB AutoSystemInfo

// AD PerfCounter

// AE PerfCounterSet

// AM Output/Configuration

// AN MemProtectHeap

// AP DbgHelpSymbolManager

// AQ CFGLogger

// AR LeakReport

// AS JavascriptDispatch/RecyclerObjectDumper

// AT HeapAllocator/RecyclerHeuristic

// AU RecyclerWriteBarrierManager

#pragma warning(disable:4075) // initializers put in unrecognized initialization area on purpose

#pragma init\_seg(".CRT$XCAS")

RecyclerObjectDumper::DumpFunctionMap \* RecyclerObjectDumper::dumpFunctionMap = nullptr;

RecyclerObjectDumper RecyclerObjectDumper::Instance;

RecyclerObjectDumper::~RecyclerObjectDumper()

{

if (dumpFunctionMap)

{

NoCheckHeapDelete(dumpFunctionMap);

}

}

BOOL

RecyclerObjectDumper::EnsureDumpFunctionMap()

{

if (dumpFunctionMap == nullptr)

{

dumpFunctionMap = NoCheckHeapNew(DumpFunctionMap, &NoCheckHeapAllocator::Instance);

}

return (dumpFunctionMap != nullptr);

}

void

RecyclerObjectDumper::RegisterDumper(type\_info const \* typeinfo, DumpFunction dumperFunction)

{

if (EnsureDumpFunctionMap())

{

Assert(!dumpFunctionMap->ContainsKey(typeinfo));

dumpFunctionMap->Add(typeinfo, dumperFunction);

}

}

void

RecyclerObjectDumper::DumpObject(type\_info const \* typeinfo, bool isArray, void \* objectAddress)

{

if (typeinfo == nullptr)

{

Output::Print(L"Address %p", objectAddress);

}

else

{

DumpFunction dumpFunction;

if (dumpFunctionMap == nullptr || !dumpFunctionMap->TryGetValue(typeinfo, &dumpFunction) || !dumpFunction(typeinfo, isArray, objectAddress))

{

Output::Print(isArray? L"%S[] %p" : L"%S %p", typeinfo->name(), objectAddress);

}

}

}

#endif

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#pragma once

#ifdef RECYCLER\_DUMP\_OBJECT\_GRAPH

class RecyclerObjectGraphDumper

{

public:

struct Param

{

bool (\*dumpReferenceFunc)(wchar\_t const \*, void \*objectAddress, void \*referenceAddress);

bool dumpRootOnly;

bool skipStack;

#ifdef RECYCLER\_STATS

RecyclerCollectionStats stats;

#endif

};

RecyclerObjectGraphDumper(Recycler \* recycler, Param \* param);

~RecyclerObjectGraphDumper();

void BeginDumpObject(void \* objectAddres);

void BeginDumpObject(wchar\_t const \* name);

void BeginDumpObject(wchar\_t const \* name, void\* objectAddress);

void EndDumpObject();

void DumpObjectReference(void \* objectAddress, bool remark);

Recycler \* recycler;

Param \* param;

wchar\_t const \* dumpObjectName;

wchar\_t tempObjectName[256];

void \* dumpObject;

#ifdef PROFILE\_RECYCLER\_ALLOC

type\_info const \* dumpObjectTypeInfo;

bool dumpObjectIsArray;

#endif

bool isOutOfMemory;

};

#endif

#ifdef RECYCLER\_DUMP\_OBJECT\_GRAPH

#define BEGIN\_DUMP\_OBJECT(recycler, address) { if (recycler->objectGraphDumper != nullptr) { recycler->objectGraphDumper->BeginDumpObject(address); }

#define BEGIN\_DUMP\_OBJECT\_ADDRESS(name, address) { if (this->objectGraphDumper != nullptr) { this->objectGraphDumper->BeginDumpObject(name, address); }

#define DUMP\_OBJECT\_REFERENCE(recycler, address) if (recycler->objectGraphDumper != nullptr) { recycler->objectGraphDumper->DumpObjectReference(address, false); }

#define DUMP\_OBJECT\_REFERENCE\_REMARK(recycler, address) if (recycler->objectGraphDumper != nullptr && recycler->IsValidObject(address)) { recycler->objectGraphDumper->DumpObjectReference(address, true); }

#define END\_DUMP\_OBJECT(recycler) if (recycler->objectGraphDumper != nullptr) { recycler->objectGraphDumper->EndDumpObject(); } }

#define DUMP\_IMPLICIT\_ROOT(recycler, address) BEGIN\_DUMP\_OBJECT(recycler, L"Implicit Root"); DUMP\_OBJECT\_REFERENCE(recycler, address); END\_DUMP\_OBJECT(recycler);

#else

#define BEGIN\_DUMP\_OBJECT(recycler, address)

#define BEGIN\_DUMP\_OBJECT\_ADDRESS(name, address)

#define DUMP\_OBJECT\_REFERENCE(recycler, address)

#define DUMP\_OBJECT\_REFERENCE\_REMARK(recycler, address)

#define END\_DUMP\_OBJECT(recycler)

#define DUMP\_IMPLICIT\_ROOT(recycler, address)

#endif

//-------------------------------------------------------------------------------------------------------

// Copyright (C) Microsoft. All rights reserved.

// Licensed under the MIT license. See LICENSE.txt file in the project root for full license information.

//-------------------------------------------------------------------------------------------------------

#include "CommonMemoryPch.h"

#ifdef RECYCLER\_DUMP\_OBJECT\_GRAPH

RecyclerObjectGraphDumper::RecyclerObjectGraphDumper(Recycler \* recycler, RecyclerObjectGraphDumper::Param \* param) :

recycler(recycler),

param(param),

dumpObjectName(nullptr),

dumpObject(nullptr),

isOutOfMemory(false)

#ifdef PROFILE\_RECYCLER\_ALLOC

, dumpObjectTypeInfo(nullptr)

#endif

{

recycler->objectGraphDumper = this;

}

RecyclerObjectGraphDumper::~RecyclerObjectGraphDumper()

{

recycler->objectGraphDumper = nullptr;

}

void RecyclerObjectGraphDumper::BeginDumpObject(wchar\_t const \* name)

{

Assert(dumpObjectName == nullptr);

Assert(dumpObject == nullptr);

dumpObjectName = name;

}

void RecyclerObjectGraphDumper::BeginDumpObject(wchar\_t const \* name, void \* address)

{

Assert(dumpObjectName == nullptr);

Assert(dumpObject == nullptr);

swprintf\_s(tempObjectName, \_countof(tempObjectName), L"%s %p", name, address);

dumpObjectName = tempObjectName;

}

void RecyclerObjectGraphDumper::BeginDumpObject(void \* objectAddress)

{

Assert(dumpObjectName == nullptr);

Assert(dumpObject == nullptr);

this->dumpObject = objectAddress;

#ifdef PROFILE\_RECYCLER\_ALLOC

if (recycler->trackerDictionary)

{

Recycler::TrackerData \* trackerData = recycler->GetTrackerData(objectAddress);

if (trackerData != nullptr)

{

this->dumpObjectTypeInfo = trackerData->typeinfo;

this->dumpObjectIsArray = trackerData->isArray;

}

else

{

Assert(false);

this->dumpObjectTypeInfo = nullptr;

this->dumpObjectIsArray = nullptr;

}

}

#endif

}

void RecyclerObjectGraphDumper::EndDumpObject()

{

Assert(this->dumpObjectName != nullptr || this->dumpObject != nullptr);

this->dumpObjectName = nullptr;

this->dumpObject = nullptr;

}

void RecyclerObjectGraphDumper::DumpObjectReference(void \* objectAddress, bool remark)

{

if (this->param == nullptr || !this->param->dumpRootOnly || recycler->collectionState == CollectionStateFindRoots)

{

if (this->param != nullptr && this->param->dumpReferenceFunc)

{

if (!this->param->dumpReferenceFunc(this->dumpObjectName, this->dumpObject, objectAddress))

return;

}

Output::Print(L"\"");

if (this->dumpObjectName)

{

Output::Print(L"%s", this->dumpObjectName);

}

else

{

Assert(this->dumpObject != nullptr);

#ifdef PROFILE\_RECYCLER\_ALLOC

RecyclerObjectDumper::DumpObject(this->dumpObjectTypeInfo, this->dumpObjectIsArray, this->dumpObject);

#else

Output::Print(L"Address %p", objectAddress);

#endif

}

Output::Print(remark? L"\" => \"" : L"\" -> \"");

recycler->DumpObjectDescription(objectAddress);

Output::Print(L"\"\n");

}

}

#endif