// Licensed to the Apache Software Foundation (ASF) under one

// or more contributor license agreements. See the NOTICE file

// distributed with this work for additional information

// regarding copyright ownership. The ASF licenses this file

// to you under the Apache License, Version 2.0 (the

// "License"); you may not use this file except in compliance

// with the License. You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing,

// software distributed under the License is distributed on an

// "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

// KIND, either express or implied. See the License for the

// specific language governing permissions and limitations

// under the License.

#include "arrow/python/pyarrow.h"

#include <memory>

#include <utility>

#include "arrow/array.h"

#include "arrow/table.h"

#include "arrow/tensor.h"

#include "arrow/type.h"

#include "arrow/util/logging.h"

#include "arrow/python/common.h"

#include "arrow/python/datetime.h"

namespace {

#include "arrow/python/pyarrow\_api.h"

}

namespace arrow {

namespace py {

static Status UnwrapError(PyObject\* obj, const char\* expected\_type) {

return Status::TypeError("Could not unwrap ", expected\_type,

" from Python object of type '", Py\_TYPE(obj)->tp\_name, "'");

}

int import\_pyarrow() {

#ifdef PYPY\_VERSION

PyDateTime\_IMPORT;

#else

internal::InitDatetime();

#endif

return ::import\_pyarrow\_\_lib();

}

#define DEFINE\_WRAP\_FUNCTIONS(FUNC\_SUFFIX, TYPE\_NAME) \

bool is\_##FUNC\_SUFFIX(PyObject\* obj) { return ::pyarrow\_is\_##FUNC\_SUFFIX(obj) != 0; } \

\

PyObject\* wrap\_##FUNC\_SUFFIX(const std::shared\_ptr<TYPE\_NAME>& src) { \

return ::pyarrow\_wrap\_##FUNC\_SUFFIX(src); \

} \

Result<std::shared\_ptr<TYPE\_NAME>> unwrap\_##FUNC\_SUFFIX(PyObject\* obj) { \

auto out = ::pyarrow\_unwrap\_##FUNC\_SUFFIX(obj); \

if (out) { \

return std::move(out); \

} else { \

return UnwrapError(obj, #TYPE\_NAME); \

} \

}

DEFINE\_WRAP\_FUNCTIONS(buffer, Buffer)

DEFINE\_WRAP\_FUNCTIONS(data\_type, DataType)

DEFINE\_WRAP\_FUNCTIONS(field, Field)

DEFINE\_WRAP\_FUNCTIONS(schema, Schema)

DEFINE\_WRAP\_FUNCTIONS(scalar, Scalar)

DEFINE\_WRAP\_FUNCTIONS(array, Array)

DEFINE\_WRAP\_FUNCTIONS(chunked\_array, ChunkedArray)

DEFINE\_WRAP\_FUNCTIONS(sparse\_coo\_tensor, SparseCOOTensor)

DEFINE\_WRAP\_FUNCTIONS(sparse\_csc\_matrix, SparseCSCMatrix)

DEFINE\_WRAP\_FUNCTIONS(sparse\_csf\_tensor, SparseCSFTensor)

DEFINE\_WRAP\_FUNCTIONS(sparse\_csr\_matrix, SparseCSRMatrix)

DEFINE\_WRAP\_FUNCTIONS(tensor, Tensor)

DEFINE\_WRAP\_FUNCTIONS(batch, RecordBatch)

DEFINE\_WRAP\_FUNCTIONS(table, Table)

#undef DEFINE\_WRAP\_FUNCTIONS

namespace internal {

int check\_status(const Status& status) { return ::pyarrow\_internal\_check\_status(status); }

PyObject\* convert\_status(const Status& status) {

DCHECK(!status.ok());

return ::pyarrow\_internal\_convert\_status(status);

}

} // namespace internal

} // namespace py

} // namespace arrow