# creating-cim-v2-experiment-documents-from-cmip6-spreadsheet

September 10, 2015

## 0.1 Setup

In [ ]:
```
import glob
import os

import pyesdoc
import pyesdoc.ontologies.cim as cim
import xlrd


# Path to test folder.
_HOME = os.path.join(os.path.expanduser("~"), "esdoc-test")

# Path to workbook being converted to CIM v2 documents.
_WORKBOOK = os.path.join(_HOME, "CMIP6Experiments.xlsx")

# Name of relevant worksheets to be found within workbook.
_WS_ENSEMBLE_REQUIREMENT = "EnsembleRequirement"
_WS_EXPERIMENT = "experiment"
_WS_FORCING_CONSTRAINT = "ForcingConstraint"
_WS_PARTY = "party"
_WS_REFERENCES = "references"
_WS_REQUIREMENT = "requirement"
_WS_TEMPORAL_CONSTRAINT = "TemporalConstraint"
_WS_URL = "url"

# Default document project code.
_DOC_PROJECT = 'CMIP6-TEST'

# Default document source.
_DOC_SOURCE = 'test-script'
```

## 0.2 Helper functions to extract data from workbook

In [ ]:
```
def _get_workbook():
    """Returns pointer to workbook object.

    """
    return xlrd.open_workbook(_WORKBOOK)


def _get_ws_names():
    """Returns set of worksheet names declared within workbook.
```

```
    """
    return _get_workbook().sheet_names()


def _get_ws(name):
    """Returns pointer to a named worksheet.

    """
    return _get_workbook().sheet_by_name(name)


def _get_ws_rows(name):
    """Returns collection of rows within a named worksheet.

    """
    return enumerate(_get_ws(name).get_rows())


def _get_ws_data(name):
    """Returns collection of rows within a named worksheet that correspond to actual data.

    """
    for idx, row in _get_ws_rows(name):
        if idx > 0:
            yield row


def _get_ws_col_map(name):
    """Returns map of column index to column names - supports situation when user reorders colu

    """
    for idx, row in _get_ws_rows(name):
        if idx == 0:
            return {col.value: col_idx for col_idx, col in enumerate(row)}


def _get_ws_cell_as_bool(value):
    """Converts a cell value to a boolean."""
    return value.lower() in [u'true', u't', u'yes', u'y']


def _get_ws_document(row, col_map, doc_type, doc_mappings):
    """Returns a cim document from a spreadsheet row.

    """
    # Create document.
    doc = pyesdoc.create(doc_type, project=_DOC_PROJECT, source=_DOC_SOURCE)

    # Apply attribute mappings.
    for mapping in doc_mappings:
        # Unpack mapping info.
        cell_value_convertor = None
        if isinstance(mapping, tuple):
            mapping, cell_value_convertor = mapping
```

```
            mapping = mapping.split(":")
            doc_attr = mapping[0]
            col_name = mapping[0] if len(mapping) == 1 else mapping[1]

            # Get cell value.
            cell_value = row[col_map[col_name]].value
            if cell_value_convertor:
                cell_value = cell_value_convertor(cell_value)

            # Set document attribute.
            setattr(doc, doc_attr, cell_value)

        return doc


    def _get_ws_documents(ws_name, doc_type, doc_mappings):
        """Returns set of cim documents within a spreadsheet."""
        result = list()
        col_map = _get_ws_col_map(ws_name)
        for row in _get_ws_data(ws_name):
            result.append(_get_ws_document(row, col_map, doc_type, doc_mappings))

        return result
```

### 0.3 Declare cell value convertors

```
In [ ]: def _convert_to_bool(value):
            """Converts a cell value to a boolean."""
            return value.lower() in [u'true', u't', u'yes', u'y']
```

### 0.4 Map worksheets to CIM v2 documents

```
In [ ]: def _get_temporal_constraints():
            """Returns set of temporal constraints definied within workbook.

            """
            mappings = [
                ("canonical_name"),
                ("conformance_is_requested:conformance_requested", _convert_to_bool),
                ("name"),
            ]

            return _get_ws_documents(_WS_TEMPORAL_CONSTRAINT, cim.v2.TemporalConstraint, mappings)

In [ ]: def _get_forcing_constraints():
            """Returns set of temporal constraints definied within workbook.

            """
            mappings = [
                ("canonical_name"),
                ("name"),
            ]

            return _get_ws_documents(_WS_FORCING_CONSTRAINT, cim.v2.ForcingConstraint, mappings)
```

## 0.5 Build document set

In [ ]:
```
docs = list()
docs += _get_temporal_constraints()
docs += _get_forcing_constraints()
```

## 0.6 Save CIM documents to file system

In [ ]:
```
# Set I/O directory.
pyesdoc.set_option("output_dir", _HOME)
```

In [ ]:
```
# Write document set to file system.
for doc in sorted(docs):
    print pyesdoc.write(doc)
```

In [ ]:
```
# Read from file system
docs = map(pyesdoc.read, glob.glob(os.path.join(_HOME, "*.json")))
for doc in sorted(docs):
    print doc
```

In [ ]:
```
# Clean up file system.
for fpath in glob.glob(os.path.join(_HOME, "*.json")):
    os.remove(fpath)
```