! This class has been made inactive. No posts will be allowed until an instructor reactivates the class.

note @23 stop following 17 views

Actions

Transform Template Parser

Here's my simple take on a parser for transformer templates as they are written in the assignment doc.

A few assumptions are made or inferred:

- The keywords `TRANSFORM`, `INPUTS`, and `OUTPUTS` are required.
- The above keywords are case-sensitive.
- The transform target C has no value
  - This could easily change later, but since the target can be any country, it appears to make more sense to have that happen elsewhere than include it as part of parsing the template.
- `INPUTS` must occur before `OUTPUTS`
- An input or output entry consists of a resource name and a quantity
  - Resource names are lowercase and uppercase letters only [a-zA-Z]
  - Quantities are integers
- One template per file

Parser:

```
# Standard Libraries
from dataclasses import dataclass, field
import os
import re
from typing import List

@dataclass
class ResourceQuantity:
  name: str = field()
  quantity: int = field()

@dataclass
class TransformTemplate:
  name: str = field(default="")
  inputs: List[ResourceQuantity] = field(default_factory=list)
  outputs: List[ResourceQuantity] = field(default_factory=list)

def read_file(file_path: str) -> List[dict]:
  file_contents = None
  with open(file_path, mode='r') as file:
    file_contents = file.read()
  return file_contents

def validate_nonempty(template: str = "") -> bool:
```

```
  return template != ""

def validate_enclosed(template: str = "") -> bool:
  left_paren_count = template.count("(")
  right_paren_count = template.count(")")
  if left_paren_count != right_paren_count:
    return False
  return True

def validate_keywords(template: str = "") -> bool:
  transform_keywords = ["TRANSFORM", "INPUTS", "OUTPUTS"]
  for keyword in transform_keywords:
    if not keyword in template:
      return False
  return True

def validate(template: str = ""):
  if not validate_nonempty(template):
    raise Exception("Empty template")
  if not validate_enclosed(template):
    raise Exception("Incorrect parentheses counts, verify all expressions are properl
y enclosed")
  elif not validate_keywords(template):
    raise Exception("Missing required keywords, verify transform is syntactically cor
rect")

def build_resource_quantities(resource_quantities_block):
  quantities = []

  regex = r"\(([A-Za-z]+) (\d)\)"
  matches = re.finditer(regex, resource_quantities_block, re.MULTILINE)
  for match in matches:
      resource_name, resource_quantity = match.groups()
      quantities.append(ResourceQuantity(name=resource_name, quantity=int(resource_qu
antity)))

  return quantities

def build_transform_template(template_path: str, template: str) -> TransformTemplate:
  transform = TransformTemplate()

  basename = os.path.basename(template_path)
  transform_name = os.path.splitext(basename)[0]
  transform.name = transform_name

  inputs_start = template.index("INPUTS")
  outputs_start = template.index("OUTPUTS")

  inputs_string = template[inputs_start:outputs_start]
```

```
  outputs_string = template[outputs_start:]

  transform.inputs = build_resource_quantities(inputs_string)
  transform.outputs = build_resource_quantities(outputs_string)

  return transform

def parse(template_path: str) -> TransformTemplate:
  template = read_file(template_path)
  validate(template)
  transform_template = build_transform_template(template_path, template)
  return transform_template
```

run code snippet

alloys.tmpl:

```
(TRANSFORM C
  (INPUTS (Population 1)
          (MetallicElements 2))
  (OUTPUTS (Population 1)
           (MetallicAlloys 1)
           (MetallicAlloysWaste 1)))
```

Usage:

```
template_path="./alloys.tmpl"
transform_template = parse(template_path)
print(transform_template)
```

Output:

```
TransformTemplate(name='alloys', inputs=[ResourceQuantity(name='Population', quantity
=1), ResourceQuantity(name='MetallicElements', quantity=2)], outputs=[ResourceQuantit
y(name='Population', quantity=1), ResourceQuantity(name='MetallicAlloys', quantity=
1), ResourceQuantity(name='MetallicAlloysWaste', quantity=1)])
```

run code snippet

There's a little bit of validation that takes place that I thought made sense to catch common errors one might make trying to write out templates by hand.

The data classes could just as easily be replaced with plain dictionaries or a different model implementation.

programming_project programming_project / part_1_discussions

~ An instructor (Will Hedgecock) endorsed this note ~

Edit good note | 3 Updated 1 year ago by John Ford

followup discussions *for lingering questions and comments*

Start a new followup discussion

Compose a new followup discussion