2020年度データ表現と処理　レポート課題

- 201811528 春名航亨
  - (知識情報・図書館学類 3年次)

お断り

- Ruby 2.7.0以上でしか動かない
  - Numbered Parameterを使用しているため

```
$ ruby -v
ruby 2.7.0p0 (2019-12-25 revision 647ee6f091) [x86_64-linux]
```

ex6.rbとex7.rbの説明

工夫点など

- クラス`Traverse`を`main`で呼ぶ
- コマンドライン引数で読み込むxmlファイルのパスを指定
  - 例: `$ ruby ex6.rb dcndl.xml`
  - クラスオブジェクトの宣言時に渡す
- 表示は、
  - ex-6.rbでは`@order_and_node_and_type`で
  - ex-7.rbでは`@node_path`で
- 2タブ

```
require 'rexml/document'

# クラスの宣言
class Traverse
  def initialize(xml)
    @xml = xml
    @doc = REXML::Document.new(File.new(@xml))
    @order_and_node_and_type = ->(order, node, type) {
      '%s|%s|%s' % [order, node, type]
    }
    @check_elm_or_text = ->(node) {
      node.kind_of?(REXML::Text) ?
        "text" : node.kind_of?(REXML::Element) ?
          "element" : "unknown"
    }
    @node_path = ->(parent, current) {
      '%s|%s' % [parent, current]
    }
  end
```

```
    def traverse(node, order, parent)
      # 6:
      # puts @order_and_node_and_type.call(
      #     order, node.name, @check_elm_or_text.call(node)
      # )
      # 7:
      # puts @node_path.call(parent, order) if parent != 0
      tmp = order
      order += 1
      if node.has_text? && ! node.has_elements?
          node.texts.each { |text|
              # 6:
              # puts @order_and_node_and_type.call(
              #     order, text, @check_elm_or_text.call(text)
              # )
              # 7:
              # puts @node_path.call(order - 1, order)
              order += 1
          }
      end
      node.elements.each do |child|
        order = traverse(child, order, tmp)
      end
      return order
    end

    def execute
      traverse(@doc.root, 1, 0)
      return 0
    end

  end

  # $ ruby ex6.rb dcndl.xml
  # $ ruby ex7.rb dcndl.xml

  def main
    t = Traverse.new($*[0])
    t.execute
  end

  exit(main)
```

ex14.rbの説明

工夫点など

- クラスXPathToSQLをmainで呼ぶ
- コマンドライン引数で読み込むXPath, dbファイルのパスを指定
    - 例: `$ ruby ex14.rb "/child::OAI-PMH" dcndl.db`
    - クラスオブジェクトの宣言時に渡す

    - 一応引数の数が2かチェック
- SQLは任意入力になるところはプレイスホルダーで
    - 要素名とかは`?`, 配列の大きさ埋め込みは`#{}`記法で
- 長ったらしいSQLはヒアドキュメントで見やすく
- XML出力は、`REXML::Formatters::Pretty`で整形
- 4タブ
- 資料中のedgeテーブルの`start`, `end`は`parent`, `child`とした

```
require 'sqlite3'
require 'rexml/document'
require 'rexml/formatters/pretty'

class XPathToSQL

    # イニシャライザ
    def initialize(xpath, db_path)
        @xpath = xpath
        @db = SQLite3::Database.new(db_path)
    end

    # WHERE句生成
    def mk_where(label_size)
        # ルート直下のみの時をセット
        a = <<~WHERE
            n1.id = 1
            and n1.name = ?
        WHERE
        # 深さ1超過のときの処理を追記
        if label_size != 1
            a += (1...label_size).map{
                <<~ADD
                    and e#{_1}.parent = n#{_1}.id
                    and e#{_1}.child = n#{_1.next}.id
                    and n#{_1.next}.name = ?
                ADD
            } * ?\n
        end
        return a
    end

    # FROM句生成
    def mk_from(label_size)
        if label_size == 1
            'node n1'
        else
            (1..label_size).map{
                "node n#{_1}, edge e#{_1}"
            }.join(?,).sub(/, edge e\d+$/, "") # 末尾の/edge n\d/を削除
        end
    end

    # 今回受け付けるXPathは
```

```
    # <Path> ::= "/child::"
    # <XPath> ::= <Path>+
    def xpath_chk(xpath, steps)
        xpath != ?/ && steps.any?{_1 !~ /^child::/}
    end

    # XPathをSQLにパース
    def xpath_to_sql()
        labels = []
        steps = @xpath.split(?/).drop(1)
        raise SyntaxError, @xpath if xpath_chk(@xpath, steps)
        steps.map{_1[7..]}.each {
            labels << _1
        }
        q_num = labels.size
        sql = <<~SQL
            SELECT n#{q_num}.id
            FROM #{mk_from(q_num)}
            WHERE #{mk_where(q_num)};
        SQL
        return {:sql => sql, :labels => labels}
    end

    # XPathをパースしたSQLを実行
    def lookup(sql, labels)
        result = []
        @db.transaction{
            @db.execute(sql, *labels) {
              result << "%s" % _1[0]
            }
        }
        return result
    end

    # ヒットしたidを持つ要素以下の木に対応するXMLを出力
    def printnode(id, out)
        current, child = [], []
        current = @db.execute(<<~SQL, id)
            SELECT name, type FROM node WHERE id = ?;
        SQL
        if current[0][1] == 'element'
            out << '<%s>' % current[0][0]
            child = @db.execute(<<~SQL, id)
                SELECT child FROM edge WHERE parent = ?;
            SQL

            child.each {|row|
                printnode(row[0].to_i, out)
            }
            out << '</%s>' % current[0][0]
        else current[0][1] == 'text'
            out << current[0][0]
        end
    end
```

```
    # XMLをきれいにして表示
    def printexe(id)
        res = ""
        @db.transaction{
            printnode(id, res)
        }

        REXML::Formatters::Pretty.new.write(
            REXML::Document.new(res), $>
        )
        $> << ?\n
    end
end

# $ ruby ex14.rb "/child::books"
# $ ruby ex14.rb "/child::books/child::book"
# $ ruby ex14.rb "/child::books/child::book/child::title"

def main
    if (argnum = $*.size) != 2
        errmsg = 'wrong number of arguments'\
                 "(given #{argnum}, expected 2)."
        raise ArgumentError, errmsg
        return 1
    end
    x = XPathToSQL.new(*$*)
    conv_sql = x.xpath_to_sql
    res_ids = x.lookup(conv_sql[:sql], conv_sql[:labels])
    res_ids.each{
        puts "=== #{_1} ==="
        x.printexe(_1)
    }
    return 0
end

exit(main)
```

感想・要望等

- 「XPathの仕様をすべて踏まえたSQLのジェネレータを作ってみたい」と感じた.
- 個人的にはそういったモジュール/クラスを「作ってみよう」授業の方が学びあるものだろうと思う.
- レポート作成にあたって自作コードのRubyを正しくシンタックスハイライトしてくれるCSSが見つからなかった.
    - `"/"`を`?/`と書いたのが悪い