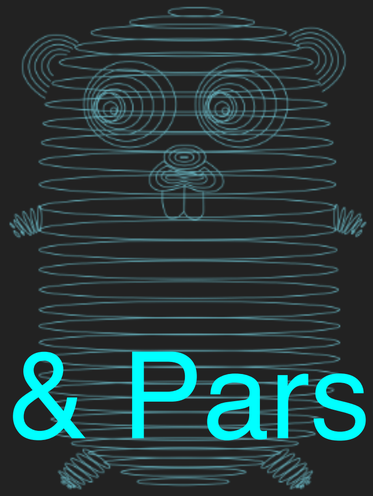
英语流利说



从零开始用 Go 实现 Lexer & Parser

何源 yuan@liulishuo.com

流利说


流利说
Help everyone become a global citizen!
github/lingochamp

流利说

Empower everyone to achieve their full potential

何源

yuan@liulishuo.com

Platform Tech Lead

Agenda

- Background
- Design
- Implementation
- Testing

Background

英语流利说

懂你英语

| 高效 | 系统 | 贴心 |

你的专属人工智能老师

G20 峰会官方志愿者语言培训提供方

Item Bank Information

ID 5731c25ed9ded376d400000d

Test level

Part 1

Bank 1

Type general

Group

Activity #1

Activity Type MCQ1

Question Audio(s) Search audios..

Display format DF1

#1 Picture Search Pictures..

Options:

TR Audio:

Search audios..

+ ADD ACTIVITY

SAVE

英语流利说

挑战

- 产品 MVP 阶段，变化快，尝试多
- 课程内容 debug 复杂，流程长
- 表单录入大量内容效率低下
- 内容经常变化，需要 version control & rollback
- 教研开发课程需要成熟稳定的工具支持

Elon Musk ✓
@elonmusk

Following

Engineering is true magic

11:36 PM - 23 Dec 2018

43,994 Retweets **239,394** Likes

4.6K 44K 239K

Tweet your reply

fab13n @fab13n · 24 Dec 2018

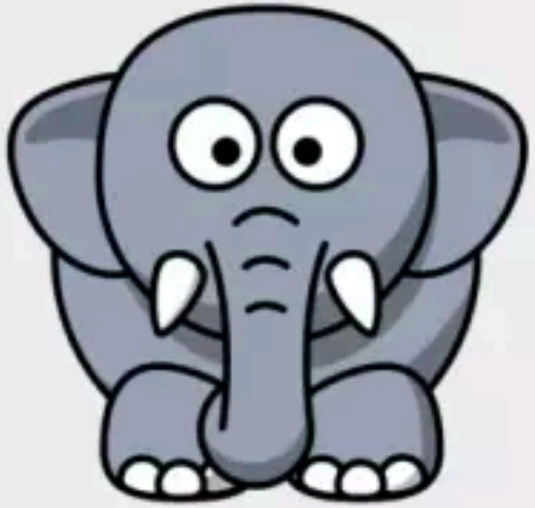
Replying to @elonmusk

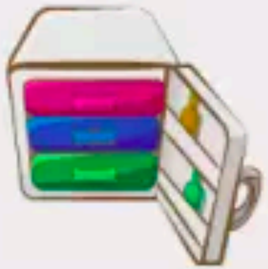
As Pratchett wrote, "it's still magic even if you know how it's done" 😋

63

没有什么问题是工程师不能解决的，如果有，那就 2 个工程师

Put the elephant





Into the fridge

- 结构需要灵活，但也要有一定的约束
- 内容发布之前 CI 检查, 每次改动都走 CI
- 使用标记语言代替表单录入，保住前端发际线！
- 教教研用 git 管理内容
- 自动化录入流程，git push -> CI -(tag) -> deploy

L03M0100.course

```
578  """
579
580  ===== Group 1526034270671845
581
582  [TYPE Pre]
583  ID:5380272044350205954
584  DFx:
585  Pic(id=5ae018865de7de77e07e9e83):a_vet.png
586  Text:He is a **vet**.
587  Audio(id=5ac20df05de7de7d6fb03e7b):He is a vet.
588  Animation:
589  Pic(effect=fadeIn, id=5ae018865de7de77e07e9e83,
     time=0):a_vet.png
590  """
591  vocab:
592  - word: vet
593    desc: 兽医
594    pos: n
595  """
596
597
598  ==== Episode 1528891694097360 COMP
599
600  ===== Group 1528876595414165
601
602  [TYPE MCP1]
603
604
605
606
607  Pic(id=5b7f9b0f5de7de09d76c899c):doctor_quit_drinking_D.png
608  Pic(id=5b750cca5de7de3b620dba77):dentist_D.png
609  TR(id=5ac20df05de7de7d6fb03e7b):He is a vet.
610  """
611  vocab:
```

Language CourseScript Preview

Level: 3 | Milestone: 30100 | Session: 13986205588382529571 | Copy Code

- Episode 1
- Episode 2
- Episode 3
- Episode 4
- Episode 5
- Episode 6
- Episode 7
- Episode 8
- Episode 9
- Episode 10
- Episode 11
- Episode 12
- Episode 13
- Episode 14
- Episode 15
- Episode 16
- Episode 17
- Episode 18

pre

< > | 1 | 2

0:01 / 0:01

He is a vet.

fadeIn

He is a **vet**.

Item Bank Information

ID 5731c25ed9ded376d400000d

Test level

Part 1

Bank 1

Type general

Group

Activity #1

Activity Type MCQ1

Question Audio(s) Search audios..

Display format DF1

#1 Picture Search Pictures..

Options:

TR Audio:

Search audios..

+ ADD ACTIVITY

SAVE


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

撸起袖子！

英语流利说

==

One data structure to fit another structure

Parsing & Lexing

撸起袖子加油干！

英语流利说

三步走

- 写一个 Parser: course file -> pb file
- 写一个 Lexer
- 串起来

1. 定义格式

```
18
19    ===== Group 1526034270606411
20
21    [TYPE Pre]
22    ID: 17662298699972008O8
23    DF1:
24    Pic(id=5849427d5de7de2948c7a825): 5thfloor.png
25    Audio(id=5849427d5de7de2948c7a825): The Jackson's apartment is fairly large.
26    Animation:
27    Pic(id=5849427d5de7de2948c7a825, effect=fadeIn, time=0): 5thfloor.png
28
```

```
==== Episode 4620445585239506279 COMP

===== Group 1526034270599757

[TYPE MCQ1]
ID:13628360098857879074
Audio(id=5b18ccd85de7de5e6af44c88):Where does he work?
Pic(id=5b15efb35de7de5a1dc9fb06):Steve_buyer_BE.png
Opts:
Text(checked=true):He works for a clothing store.
Text:He works for a shoe factory.
Text:He is a buyer.
TR(id=5b2a180c5de7de5e6af4755d):He works for a large clothing store in New York.
```

```
[TYPE C&D]
ID:17662298699972008 21
Pic(id=5849427d5de7de2948c7a825): Bill_Helen.png
Text: They {{don't/doesn't/different}} teach at the {{same}} school.
TR(id=5849427d5de7de2948c7a825): They don't teach at the same school.
```

2. 定义语法

英语流利说

Extended Backus-Naur Form(aka. EBNF)

is a code that expresses the grammar of a formal language

```
Activity = Blocks | eof;
Blocks = Block ( Emptys Blocks | eof );
Emptys = empty { empty };
Block = Type | ID | DisplayFormat | Element | Animation | Options | TR;
Type = LeftSquare, 'TYPE',  Space,  String, RightSquare;
ID = "ID", Colon | Paragraph;
DisplayFormat = "DF" | VisibleCharacter { VisibleCharacter } | NewLine;
Element = { Picture } | { Audio } | { Text };
Animation = 'Animation', Colon | Picture { Picture };
Options = "Opts", Colon | NewLine | Element { Element };
TR = "TR", Colon | paragraph;
Picture = "Pic", Colon | Attrs | paragraph;
Text = "Text", Colon | Attrs | paragraph;
Audio = "Audio", Colon | Attrs | paragraph;
Attrs = LeftParen, ( Attr { Attr }),RightParen | NewLine;
Attr = ( String, "=", String, Space );

Paragraph = String { String };
String = VisibleCharacter { VisibleCharacter } Newline;
VisibleCharacter = Unicode | Others | Alphanum | Escaped |
InlineWhitespace;
Escaped = Backslash Special;
Unicode = "&" Alphanum Alphanum Alphanum Alphanum ";";
Alphanum = ( Alphabet | Digit );
Alphabet = "A"| "B" | "C" | "D" | "E" | "F" | "G" | | "I" | "J" |
            "K" | "L" | "M" | "N" | "O" | "P" | "Q" "R" | "S" | "T" |
            "U" | "V" | "W" | "X" | "Y" | "Z" | "a" "b" | "c" | "d" |
            "e" | "f" | "g" | "h" | "i" | "j" | "k" "l" | "m" | "n" |
            "o" | "p" | "q" | "r" | "s" | "t" | "u" "v" | "w" | "x" |
            "y" | "z";
Number = NonZeroDigit { Digit };
NonZeroDigit = "1" | "2" | "3" | "4" | "5" | "6" | "8" | "9";
Digit = "0" | NonZeroDigit;
Newline = "\n";
InlineWhitespace = Tab | Space;
Tab = "\t";
Space = "\s";
Whitespace = "\s" | "\t" | "\n" | "\r";
RightSquare = "]";
LeftSquare = "[";
Colon = ":";
LeftParen = "(";
RightParen = ")";
Slash = "/";
LeftDoubleCurly = "{{";
RightDoubleCurly = "}}";
```

3. 实现

实现

- 使用现有工具，比如 goyacc
- 正则表达式？:P
- Use states, actions, and a switch statement

现有工具

- 难 debug (can be very important)
- 有可能要多学一种语言，比如 EBNF
- 有可能性能不及预期
- 不适合 MVP 期间的产品迭代

正则

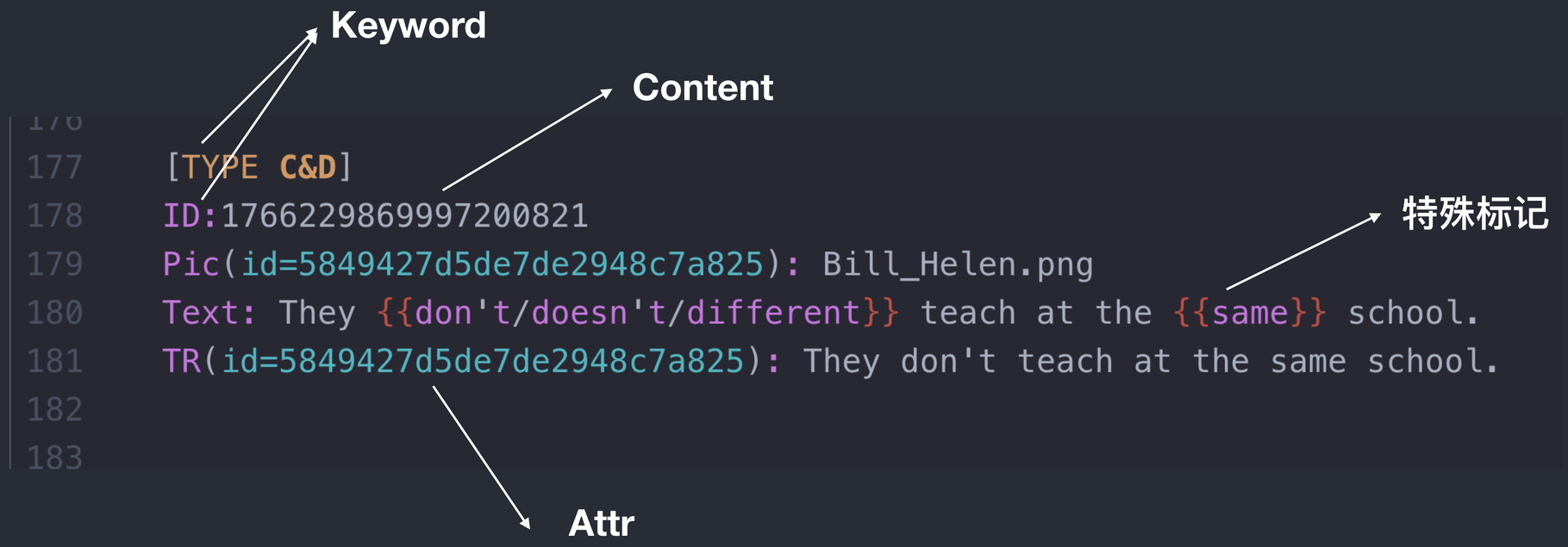
Some people, when confronted with a problem, think “I know, I'll use regular expressions.” Now they have two problems.

–Jamie Zawinski

英语流利说

Thinking in Go

英语流利说


Keyword
Content
特殊标记
[TYPE C&D]
ID:17662298699972008 21
Pic(id=5849427d5de7de2948c7a825): Bill_Helen.png
Text: They {{don't/doesn't/different}} teach at the {{same}} school.
TR(id=5849427d5de7de2948c7a825): They don't teach at the same school.
Attr

start

cursor

```
[TYPE C&D]\nID:17662298699972008 21\nPic(id=5849427d5de7de2948c7a825):
Bill_Helen.png\nText: They {{don't/doesn't/different}} teach at the
{{same}} school.\nTR(id=5849427d5de7de2948c7a825): They don't teach at
the same school.\n\n\n\n[TYPE MCQ3]\nID:17662298699972009 01\nText: We
touch things ____ our hands.\nOpts:\nText: to  \nText(checked=true):
with \nText: for  \nText: around\nTR: The quick brown fox jumps over the
lazy dog haha
```

end

State Machine

StateFunc

Executes an action, returns the next state

```
// stateFn represents the state of the scanner
// as a function that returns the next state.
type stateFn func(*lexer) stateFn
```

The run loop

```
// run lexes the input by executing state functions
// until the state is nil.
func run() {
  for state := startState; state != nil; {
    state = state(lexer)
  }
}
```

1. token item

```
// item represents a token or text string returned from the scanner.
type item struct {
  typ  itemType // The type of this item
  pos  Pos      // The starting position, in bytes, of this item in the input
string
  val  string   // The value of this item.
  line int      // The line number at the start of this item.
}
```

```
type itemType int

const (
  itemNil   itemType = iota // used in the parser to indicate no type
  itemError                 // error occurred; value is text of error
  itemEOF
  itemBool                  // Bool
  itemLeftParen             // '('
  itemRightParen            // ')'
  itemLeftDoubleCurly       // "{{"
  itemSlash                 // '/' in side options
  itemRightDoubleCurly      // "}}"
  itemLeftSquare            // '[' in type defination [TYPE a]
  itemRightSquare           // '[' in type defination [TYPE b]
  itemComma                 // ,

}
```

```
func (i item) String() string {
  switch {
  case i.typ == itemEOF:
    return "EOF"
  case i.typ == itemError:
    return i.val
  case i.typ > itemKeyword:
    return fmt.Sprintf("<%s>", i.val)
  case len(i.val) > 10:
    return fmt.Sprintf("%.10q...", i.val)
  }
  return fmt.Sprintf("%q", i.val)
}
```

2. lexer

```
// lexer holds the state of the scanner.
type lexer struct {
  name    string    // the name of the input; used only for error reports
  input   string    // the string being scanned
  state   stateFn   // the next lexing function to enter
  pos     Pos       // current position in the input
  start   Pos       // start position of this item
  width   Pos       // width of last rune read from input
  lastPos Pos       // position of most recent item returned by nextItem
  items   chan item // channel of scanned items
  line    int       // 1+number of newlines seen
}
```

```go
// lex creates a new scanner for the input string.
func lex(name, input string) *lexer {
  l := &lexer{
    name:  name,
    input: input,
    items: make(chan item),
    line:  1,
  }
  go l.run()
  return l
}
```

```
// run lexes the input by executing state functions until
// the state is nil.
func (l *lexer) run() {
  for l.state = lexTop; l.state != nil; {
    l.state = l.state(l)
  }
  close(l.items) // No more tokens will be delivered.
}
```

英语流利说

start

cursor

```
[TYPE C&D]\nID:17662298699972008 21\nPic(id=5849427d5de7de2948c7a825):
Bill_Helen.png\nText: They {{don't/doesn't/different}} teach at the
{{same}} school.\nTR(id=5849427d5de7de2948c7a825): They don't teach at
the same school.\n\n\n\n[TYPE MCQ3]\nID:17662298699972009 01\nText: We
touch things ____ our hands.\nOpts:\nText: to  \nText(checked=true):
with \nText: for  \nText: around\nTR: The quick brown fox jumps over the
lazy dog haha
```

end

```go
// next returns the next rune in the input.
func (l *lexer) next() rune {
  if int(l.pos) >= len(l.input) {
    l.width = 0
    return eof
  }
  r, w := utf8.DecodeRuneInString(l.input[l.pos:])
  l.width = Pos(w)
  l.pos += l.width
  if r == '\n' {
    l.line++
  }
  return r
}
```

```go
// peek returns but does not consume the next rune in the input.
func (l *lexer) peek() rune {
  r := l.next()
  l.backup()
  return r
}

// backup steps back one rune. Can only be called once per call of next.
func (l *lexer) backup() {
  l.pos -= l.width
  // Correct newline count.
  if l.width == 1 && l.input[l.pos] == '\n' {
    l.line--
  }
}

// current retruns the string which between l.start and l.pos
func (l *lexer) current() string {
  return l.input[l.start:l.pos]
}
```

```
// ignore skips over the pending input before this point.
func (l *lexer) ignore() {
  l.start = l.pos
}

// skip ignores all input that matches the given predicate.
func (l *lexer) skip(pred func(rune) bool) {
  for {
    r := l.next()
    if pred(r) {
      continue
    }
    l.backup()
    l.ignore()
    return
  }
}
```

```
// emit passes an item back to the client.
func (l *lexer) emit(t itemType) {
  leadingSpace := ""
  // Some items contain text internally. If so, count their newlines.
  switch t {
  case itemText, itemString:
    l.line += strings.Count(l.input[l.start:l.pos], "\n")
    leadingSpace = strings.Repeat(" ", l.leadingSpaceCnt)
  }
  l.items <- item{t, l.start, leadingSpace + l.current(), l.line}
  l.start = l.pos
  l.resetLeadingSpaceCnt()
}
```

串起来

1. Run the state machine as a goroutine
2. Emit values on a channel

```
// run lexes the input by executing state functions until
// the state is nil.
func (l *lexer) run() {
  for l.state = lexTop; l.state != nil; {
    l.state = l.state(l)
  }
  close(l.items) // No more tokens will be delivered.
}
```

```
// lexTop consumes elements at the top level of
CouseScript text
func lexTop(l *lexer) stateFn {
   r := l.next()
   switch {
   case isSpace(r) || isEndOfLine(r):
      if isSpace(r) {
         l.leadingSpaceCnt++
      }
      l.ignore()
      return lexTop
   case r == eof:
      if l.pos > l.start {
         fmt.Printf("%q\n", l.input[l.start:l.pos])
         return l.errorf("unexpected EOF")
      }
      l.emit(itemEOF)
      return nil
   }

   // backup if next char is valid
   l.backup()
   switch s := l.input[l.pos:]; {
   case hasPrefix(s, commentStart):
      return lexCommentStart
   default:
      return lexActivity
   }
}
```

```
// lexComment scans a comment
func lexComment(l *lexer) stateFn {
  r := l.peek()
  if isEndOfLine(r) || r == eof {
    l.emit(itemText)
    return lexTop
  }
  l.next()
  return lexComment
}
```

```
func lexActivity(l *lexer) stateFn
{
  r := l.next()
  switch {
  case isKeywordChar(r):
    if unicode.IsLetter(r) {
      l.backup()
    }
    return lexContent
  case r == leftSquare:
    l.ignore()
    return lexActivityType
  }
  return lexTop
}
```

start

cursor

```
[TYPE C&D]\nID:17662298699972008 21\nPic(id=5849427d5de7de2948c7a825):
Bill_Helen.png\nText: They {{don't/doesn't/different}} teach at the
{{same}} school.\nTR(id=5849427d5de7de2948c7a825): They don't teach at
the same school.\n\n\n\n[TYPE MCQ3]\nID:17662298699972009 01\nText: We
touch things ____ our hands.\nOpts:\nText: to  \nText(checked=true):
with \nText: for  \nText: around\nTR: The quick brown fox jumps over the
lazy dog haha
```

end

```
func isKeywordChar(r rune) bool {
  c := map[rune]bool{
    '_':          true,
    ' ':          true,
    leftParen:    true,
    doubleQuote:  true,
    rightParen:   true,
    colon:        true,
    comma:        true,
    keySep:       true,
    leftCurly:    true,
    rightCurly:   true,
  }
  return c[r] || unicode.IsLetter(r)
}
```

```
func isValidChar(r rune) bool {
  valid := isSpace(r) ||
    unicode.IsPunct(r) ||
    unicode.IsLetter(r) ||
    unicode.IsDigit(r) ||
    unicode.IsNumber(r) ||
    unicode.IsSymbol(r)
  return valid && r != leftCurly && r !=
rightCurly
}
```

```
func (l *lexer) debug(format string, v ...interface{}) {
  log.SetFlags(log.LstdFlags | log.Lshortfile)
  format = fmt.Sprintf("%s:%d ==> %q\n", l.name, l.line, format)
  log.Printf("[DEBUG] "+format, v...)
}
```

```
// errorf returns an error token and terminates the scan by passing
// back a nil pointer that will be the next state, terminating
l.nextItem.
func (l *lexer) errorf(format string, args ...interface{}) stateFn {
  l.items <- item{
    itemError,
    l.start,
    fmt.Sprintf(format, args...),
    l.line,
  }
  return nil
}
```

3. parser

parsing

1. Define AST
2. Lex item -> AST node

```
// A Node is an element in the parse tree. The interface is trivial.
// The interface contains an unexported method so that only
// types local to this package can satisfy it.
type Node interface {
  Type() NodeType
  Position() Pos // byte position of start of node in full original
input string
  String() string
  // tree returns the containing *Tree.
  // It is unexported so all implementations of Node are in this
package.
  tree() *Tree
}
```

```
// Tree is the representation of a single parsed file.
type Tree struct {
  Name      string
  Root      *ListNode
  lex       *lexer
  token     [3]item // three-token lookahead for parser.
  peekCount int
}
```

```
// Parse retruns a Tree of coursescript
func Parse(name, text string) (*Tree, error) {
  t := New(name)
  _, err := t.Parse(text)
  return t, err
}

// Parse parses the text to a representation of Course
func (t *Tree) Parse(text string) (tree *Tree, err error) {
  defer t.recover(&err)
  t.lex = lex(t.Name, text)
  if err = t.parse(); err != nil {
    return nil, err
  }
  return t, nil
}
```

```
// Parse parses the text to a representation of Course
// parse is the top-level parser for a script
// it runs to EOF.
func (t *Tree) parse() error {
  t.Root = t.newList(t.peek().pos)
  for t.peek().typ != itemEOF {
    n, err := t.hrchyOrActivity(t.lastHrchyNode)
    if err != nil {
      return err
    }
    if n == nil {
      break
    }
    t.Root.append(n)
  }
  return nil
}
```

```
// peek returns but does not consume the nex
func (t *Tree) peek() item {
  if t.peekCount > 0 {
    return t.token[t.peekCount-1]
  }
  t.peekCount = 1
  t.token[0] = t.lex.nextItem()
  return t.token[0]
}
```

```
// activity parse the activity body in loop
// and break if a hrchyNode found
func (t *Tree) activity(parent *HrchyNode) (Node, error) {
  var act *ActivityNode
  for {
    item := t.next()
    switch {
    case item.typ < itemKeyword || item.typ == itemEOF:
      // return nil so t.Parse can handle it
      return nil, nil
    case item.typ == itemkType:
      n, err := t.expect(itemText)
      if err != nil {
        return nil, err
      }
      act = t.newActivity(item.pos, parent, t.newResource(NodeActivityType,
n.pos, n.val))
      // then call next to get resource nodes
      return t.parseResource(act)
    }
  }
}
```

```
type ActivityNode struct {
  NodeType
  Pos
  Line     int
  Parent   *HrchyNode
  tr       *Tree
  Typ      *ResourceNode
  Resource ListNode
}
```

```
func (t *Tree) expect(typ itemType) (item, error) {
  return it, t.assertEqual(typ, it.typ)
}

// newActivity create a new activity Node
func (t *Tree) newActivity(pos Pos, parent *HrchyNode, typ *ResourceNode) *ActivityNode {
  return &ActivityNode{
    NodeType: NodeActivity,
    Pos:      pos,
    Line:     t.token[0].line,
    Typ:      typ,
    Parent:   parent,
    Text:     []byte(text),
  }
}
```

```
func (a *ActivityNode) String() string {
  b := bytes.NewBufferString(a.Typ.String())
  for _, n := range a.Resource.Nodes {
    b.WriteString(n.String())
  }
  return b.String()
}
```

Testing

1. Lexer

```
type lexTest struct {
 name  string
 input string
 items []item
}

func mkItem(typ itemType, text string) item {
 return item{
    typ: typ,
    val: text,
 }
}

var (
 tEOF    = mkItem(itemEOF, "")
 tType   = mkItem(itemkType, "TYPE")
 tPic    = mkItem(itemkPic, "Pic")
 tAudio  = mkItem(itemkAudio, "Audio")
 tText   = mkItem(itemkText, "Text")
 tID     = mkItem(itemkID, "ID")
 tOpts   = mkItem(itemkOpts, "Opts")
)

var lexTests = []lexTest{
   {"empty", "", []item{tEOF}},

   //--------------------  Activities
   {"activityType", "[TYPE MCQ1]", []item{
      tType,
      mkItem(itemText, " MCQ1"),
      tEOF,
   }},
   {"invalid activityType end", "[TYPE MCQ1", []item{
      tType,
      mkItem(itemText, " MCQ1"),
      mkItem(itemError, `expected ']' to end the activity type
instead`),
   }},
   {"invalid activityType", "TYPE MCQ1", []item{
      mkItem(itemError, "TYPE keyword should be started with
   }},
   {"pic", "Pic: abc.jpg", []item{
      tPic,
      tColon,
      mkItem(itemText, " abc.jpg"),
      tEOF,
   }},
}
```

```
50
51 // collect gathers the emitted items into a slice.
52 func collect(t *lexTest) (items []item) {
53  l := lex(t.name, t.input)
54  for {
55    item := l.nextItem()
56    items = append(items, item)
57    if item.typ == itemEOF || item.typ == itemError {
58      break
59    }
60  }
61  return
62 }
63
64 func TestLex(t *testing.T) {
65  // n := len(lexTests) - 2
66  for _, test := range lexTests {
67    items := collect(&test)
68    if !equal(items, test.items, false) {
69      t.Errorf("%s: got\n\t%+v\nexpected\n\t%v", test.name, items, test.items)
70    }
71  }
72 }
```

2. Parser

```
 6 func TestParse(t *testing.T) {
 7  for _, test := range parseTests {
 8    result, err := New(test.name).Parse(test.input)
 9    switch {
10    case err == nil && !test.ok:
11      t.Errorf("%q: expected error; got none", test.name)
12      continue
13    case err != nil && test.ok:
14      t.Errorf("%q: unexpected error: %v", test.name, err)
15    case err != nil && !test.ok:
16      if *debug {
17        fmt.Printf("%s: %s\n\t%s\n", test.name, test.input, err)
18      }
19      continue
20    }
21
22    r := result.Root.String()
23    if r != test.result {
24      t.Errorf("%s: got\n%#v\nexpected\n\t%#v", test.name, r, test.result)
25    }
26  }
27 }
```

Lexers are fun!

Parsers are fun!

Go is awesome!!

如非万不得已，不要自己写 Parser :P

英语流利说

Yes! Go 浪!

流利说专属周边现已上线!

Thank you!