

# Introducing the StringTokenizer

**John Thickstun**  
**Sirius Software Inc.**



Sirius Software, Inc.

# Parsing linear data

- Xml and other non-linear structures are becoming more and more common
- But, much data still exists in a less structured form
- For example
  - CSV files
  - Configuration files
  - Code
- So Sirius has created the StringTokenizer to help process these diverse formats



# The StringTokenizer

- Parse a file into discrete pieces of data: tokens
- A token is a piece of data surrounded by whitespace
  - The string 'alice bob charlie' tokenizes (by default) to 'alice', 'bob', 'charlie'
- Step through the file token by token
  - Less messy logic for handling delimiters, whitespace, etc.
- Define for yourself what a token is
  - By default, the StringTokenizer treats spaces as whitespace, but you can set whatever characters you want to be treated as whitespace



# Tokenizing a String

b

```
%toke          is object StringTokenizer
```

```
%toke = new
```

```
%toke:string = 'one two five!'
```

```
repeat while %toke:notAtEnd
```

```
    print %toke:nextToken
```

```
end repeat
```

```
end
```

```
one  
two  
five!
```



# Getting tokenized data

- Internal string cursor keeps track of where you are in the tokenizing process
- You can iterate through tokens
  - NextToken - gets the next token
  - CurrentToken - gets the current token
  - PeekToken - look at the next token, but don't advance the cursor yet
- Loop until you run out of tokens
  - atEnd, notAtEnd tell you whether you've got all the tokens
- Use skipTokens to skip uninteresting tokens



# Token Characters

- Most data isn't that simple
- What about something like '2+2=5'?
- No problem!
- TokenChars are the answer
  - TokenChars are treated as distinct tokens, even when they aren't surrounded by whitespace
  - To deal with '2+2=5' just make '+' and '=' tokenchars
  - Then '2+2=5' tokenizes to '2', '+', '2', '=', '5'
- Set tokenchars using the 'tokenchars=' parameter when creating a new tokenizer, or use the tokenChars method



# Tokenizing Arithmetic

```
b
%toke          is object StringTokenizer
%output       is longstring

%toke = new(tokenchars='+-*/=,')
%toke:string = '2+2=5, x*x + 1 = 0, y=seven'

%output = ''
repeat while %toke:notAtEnd
  if %toke:nextToken = ',' then
    print %output
    %output = ''
  else
    %output = %output with '"' with %toke:currentToken with '", '
  end if
end repeat
print %output
end
```

```
"2", "+", "2", "=", "5",
"x", "*", "x", "+", "1", "=", "0",
"y", "=", "seven",
```



# Whitespace

- You can define for yourself what's whitespace
- This may be useful for CSV parsing
  - Just make ',' a whitespace character
  - Then 'alice,bob,charlie' will tokenize to 'alice', 'bob', 'charlie'
- To define whitespace characters, use the 'spaces=' named parameter when creating your StringTokenizer
- Or use the (StringTokenizer):spaces method to change your current tokenizer's whitespaces



# Quote Characters

- Use quotes to “turn off” the StringTokenizer
- The default quote character is '"', but of course you can redefine it to whatever you please
- Any part of the string enclosed by quotes will be treated as an individual token
- A couple utilities:
  - RemoveQuotes - set to true/false to determine whether a quoted token's value should include the enclosing quotes
  - CurrentQuoted - returns true if the current token is quoted



# Comma Separated Values

b

```
%toke          is object StringTokenizer  
%list          is object StringList
```

```
%toke = new(spaces=', ' with '25':X,quotes='\"')  
%toke:removeQuotes = True
```

```
%list = new  
text to %list  
  John,"New York","Sirius Software"  
  Gary,Massachusetts,"Sirius Software"  
  Goff,Australia,Centrelink  
end text
```

```
%toke:string = %list:createLines
```

```
repeat while %toke:notAtEnd  
  Print %toke:nextToken with 'lives in' with %toke:nextToken(skip=1)  
end repeat
```

```
end
```

```
John lives in New York  
Gary lives in Massachusetts  
Goff lives in Australia
```



## Some more useful methods

- NextChar – advance one character
- PeekChar – look at the next character
- FindToken – Find a token in the string; advance the cursor to its position
- CurrentTokenPosition – the starting position in the string of the current token
- NextPosition – the position in the string just after the current token
- TokensToUpper/TokensToLower



# More CSV parsing

```
...
%list = new
text to %list
  John,"New York","Sirius Software"
  Gary,Massachusetts,"Sirius Software"
  Goff,Australia,Centrelink
end text
...

%toke:tokensToUpper = True
%toke:nextPosition = 1
if %toke:findToken('GARY') then
  print 'Gary:'
  repeat 2 times
    print %toke:nextToken
  end repeat
end if

end
```

```
Gary:
MASSACHUSETTS
SIRIUS SOFTWARE
```



# Extracting a substring

```
%start is float
```

```
...  
%list = new  
text to %list  
    John,"New York","Sirius Software"  
    Gary,Massachusetts,"Sirius Software"  
    Goff,Australia,Centrelink  
end text
```

```
...
```

```
%toke:tokenchars='25':X  
%toke:spaces=', '
```

```
%toke:nextPosition = 1  
if %toke:findToken('Goff') then  
    %start = %toke:currentTokenPosition  
    %toke:findToken('25':X)  
    print %toke:substring(%start,%toke:currentTokenPosition  
                          - %start)  
end if
```

```
Goff,Australia,Centrelink
```



# Conclusions

- The StringTokenizer can be a useful way to pick apart data for further processing
- It matches or beats the performance of comparable existing facilities
  - Comparable performance to \$word; much better performance than \$lstr\_word
  - Comparable performance with \$parse/\$parseex, but the StringTokenizer is much more flexible

