

Tree Language Developer's Guide

Ceyhun Ciper

November 2009

Contents

1	Functionality	1
2	Usage	1
2.1	API	1
2.2	Examples	1
3	Reference	2
3.1	Data Structures	2
3.1.1	Tree	2

1 Functionality

1. Parse a tree and return it as a “Tree” class
2. Parse a tree and display it as a tree in xaml

2 Usage

2.1 API

In the following, a *tree definition* is a string and a *tree* is a Tree:

```
“tree definition”.ToInteractiveTree() → Control  
“tree definition”.ToCanvas() → Canvas  
“tree definition”.ToXaml() → Xaml  
“tree definition”.ToTree() → Tree  
tree.ToInteractiveTree() → Control  
tree.ToCanvas() → Canvas  
tree.ToXaml() → Xaml
```

2.2 Examples

Code examples:

```
“a ( a1 ( a11 a12 ) a2 )”.ToInteractiveTree() → Control  
“a ( a1 ( a11 a12 ) a2 )”.ToCanvas() → Canvas  
“a ( a1 ( a11 a12 ) a2 )”.ToXaml() → Xaml  
“a ( a1 ( a11 a12 ) a2 )”.ToTree() → Tree  
“a ( a1 ( a11 a12 ) a2 )”.ToTree().ToCanvas() → Canvas  
“a ( a1 ( a11 a12 ) a2 )”.ToTree().ToXaml() → Xaml
```

3 Reference

3.1 Data Structures

3.1.1 Tree

Tree:

Node: *anything*

SubTrees: *list of Trees*

Depth: *depth of the tree*