TextGen

TextGen language aspect

Introduction

The *TextGen* language aspect defines a model to text transformation. It comes in handy each time you need to convert your models into the text form directly. The language contains constructs to print out text, transform nodes into text values and give the output some reasonable layout.

Operations

The *append* command performs the transformation and adds resulting text to the output. You can use *found error* command to report problems in the model. The *with indent* command demarcates blocks with increased indentation. Alternatively, the *increase depth* and *decrease depth* commands manipulate the current indentation depth without being limited to a block structure. The *indent buffer* command applies the current indentation (as specified by *with ident* or *increase/decrease depth*) for the current line.

Operation	Arguments
append	any number of: • {string value} • \n • $list{node.list} - list without separator • $list{node.list with ,} - with separator • $ref{node.reference} • ${node.child}
found error	error text
decrease depth	decrease indentation level from now onwards
increase depth	increase indentation level from now on
indent buffer	apply indentation to the current line
with indent { <code> }	increase indentation level for the <code>

Examples

1. Here is an example of the **text gen** component for the *ForeachStatement* (jetbrain.mps.baseLanguage).

```
text gen component for concept ForeachStatement {
  (node, context, buffer)->void {
    if (node.loopLabel != null) {
      append \n ${node.loopLabel.name} {:} ;
    } else if (node.label != null) {
      append \n ${node.label} {:} ;
    }
    append \n ;
    indent buffer ;
    append {for (} ${node.variable} { : } ${node.iterable} {) {} ;
    with indent {
      append ${node.body} ;
    }
    append \n {}} ;
  }
}
```

1. This is an artificial example of the **text gen**:

```
text gen component for concept CodeBlockConcept {
  (node, context, buffer)->void {
      indent buffer ;
      append {codeBlock {} \n ;
      with indent {
        indent buffer ;
        append {// Begin of codeBlock} \n ;

        indent buffer ;
        append {int i = 0} \n ;

        indent buffer ;
        append {// End of codeBlock} \n ;
      }
      append {}} ;
  }
}
```

producing following code block containing a number of lines with indentation:

```
text gen component for concept CodeBlockConcept {
codeBlock {
  // Begin of codeBlock
  int i = 0
  // End of codeBlock
}
```