NAME

`Content_Node` − Node in an HTML tree for HTML elements that possibly have attributes and child nodes, i.e., content.

SYNOPSIS

```
class Content_Node : public Element_Node, public Parent_Node {
public:
      Content_Node( char const *name, element const&, Parent_Node* = 0 );
      Content_Node(
            char const *name, element const&,
            char const *att_begin, char const *att_end,
            Parent_Node* = 0
      );
      virtual ~Content_Node();

      // overridden
      virtual void      visit( visitor const&, int depth = 0 );

      // inherited
      attribute_map     attributes;
      void              add_child( HTML_Node *child );
      child_list& children();
      child_list const& children() const;
      bool              empty() const;
      bool              remove_child( HTML_Node *child );
      char   const*     name() const;
      Content_Node*     parent() const;
      void              parent( Content_Node *new_parent );
};
```

DESCRIPTION

`Content_Node` is-an `Element_Node` and an `Parent_Node` that contains attributes and child nodes, i.e., content. For example, the `SELECT` element below is a parent of the newline `Text_Node` after the `SELECT` and all of the `OPTION` elements:

```
<SELECT NAME="menu">          parent child
      <OPTION>Blueberry       child grandchild
      <OPTION>Chocolate       child grandchild
      <OPTION>Raspberry       child grandchild
</SELECT>
```

whereas an element such as `IMG` can have no child nodes.

Public Interface

Constructors

These are the same as those for `Element_Node`.

Destructor

There is nothing noteworthy about it.

`virtual void visit( visitor const &v, int depth = 0 )`

This member function overrides `visit()`. First, it visits this node passing `false` for `is_end_tag`. If the visitor's `operator()` returns `false`, return immediately. Otherwise, call `Parent_Node::visit()` that visits each child node in order passing `depth + 1`, then visit this node again passing `true` for `is_end_tag`. If the visitor's `operator()` returns `false`, return immediately. Otherwise repeat the entire visit cycle. In pseudo-code:

```
do {
      if ( !v( this, depth, false ) )
            break;
      Parent_Node::visit( v, depth + 1 );
} while ( v( this, depth, true ) );
```

SEE ALSO

Element_Node(3), **HTML_Node**(3), **Parent_Node**(3).

AUTHOR

Paul J. Lucas <*pjl@best.com*>