---+ Package `TWiki::Infix::Parser`

A simple stack-based parser that parses infix expressions with nonary, unary and binary operators specified using an operator table.

Escapes are supported in strings, using backslash.

new($client_class, \%options) -> parser object

Creates a new infix parser. Operators must be added for it to be useful.

The tokeniser matches tokens in the following order: operators, quotes (" and '), numbers, words, brackets. If you have any overlaps (e.g. an operator '<' and a bracket operator '<<') then the first choice will match.

`$client_class` needs to be the *name* of a *package* that supports the following two functions:

- `newLeaf($val, $type)` - create a terminal. $type will be:
    1. if the terminal matched the `words` specification (see below).
    2. if it is a number matched the `numbers` specification (see below)
    3. if it is a quoted string
- =newNode($op, @params) - create a new operator node. @params is a variable-length list of parameters, left to right. $op is a reference to the operator hash in the \@opers list.

These functions should throw Error::Simple in the event of errors. TWiki::Infix::Node is such a class, ripe for subclassing.

The remaining parameters are named, and specify options that affect the behaviour of the parser:

1. `words=>qr//` - should be an RE specifying legal words (unquoted terminals that are not operators i.e. names and numbers). By default this is `\w+`. It's ok if operator names match this RE; operators always have precedence over atoms.
2. `numbers=>qr//` - should be an RE specifying legal numbers (unquoted terminals that are not operators or words). By default this is `qr/[+-]?(?:\d+\.\d+|\d+\.|\.\d+|\d+)(?:[eE][+-]?\d+)?/`, which matches integers and floating-point numbers. Number matching always takes precedence over word matching (i.e. "1xy" will be parsed as a number followed by a word. A typical usage of this option is when you only want to recognise integers, in which case you would set this to `numbers => qr/\d+/`.

ObjectMethod addOperator `(\%oper)`

Add an operator to the parser.

`\%oper` is a hash (or an object), containing the following fields:

- `name` - operator string
- `prec` - operator precedence, positive non-zero integer. Larger number => higher precedence.
- `arity` - set to 1 if this operator is unary, 2 for binary. Arity 0 is legal, should you ever need it.
- `close` - used with bracket operators. `name` should be the open bracket string, and `close` the close bracket. The existance of `close` marks this as a bracket operator.
- `casematters=` - indicates that the parser should check case in the operator name (i.e. treat 'AND' and 'and' as different). By default operators are case insensitive. **Note** that operator names must be caselessly unique i.e. you can't define 'AND' and 'and' as different operators in the same parser. Does not affect the interpretation of non-operator terminals (names).

Other fields in the hash can be used for other purposes; the parse tree generated by this parser will point to the hashes passed to this function.

Field names in the hash starting with `InfixParser_` are reserved for use by the parser.

ObjectMethod parse `($string) -> $parseTree`

Parses `$string`, calling `newLeaf` and `newNode` in the client class as necessary to create a parse tree. Returns the result of calling `newNode` on the root of the parse.

Throws TWiki::Infix::Error in the event of parse errors.

This topic: TWiki > TWikiInfixParserDotPm
Topic revision: r2 - 2008-12-06 - TWikiContributor