## NAME

XML::SAX::ParserFactory – Obtain a SAX parser

## SYNOPSIS

```
use XML::SAX::ParserFactory;
use XML::SAX::XYZHandler;
my $handler = XML::SAX::XYZHandler->new();
my $p = XML::SAX::ParserFactory->parser(Handler => $handler);
$p->parse_uri("foo.xml");
# or $p->parse_string("<foo/>") or $p->parse_file($fh);
```

## DESCRIPTION

XML::SAX::ParserFactory is a factory class for providing an application with a Perl SAX2 XML parser. It is akin to DBI – a front end for other parser classes. Each new SAX2 parser installed will register itself with XML::SAX, and then it will become available to all applications that use XML::SAX::ParserFactory to obtain a SAX parser.

Unlike DBI however, XML/SAX parsers almost all work alike (especially if they subclass XML::SAX::Base, as they should), so rather than specifying the parser you want in the call to `parser()`, XML::SAX has several ways to automatically choose which parser to use:

- `$XML::SAX::ParserPackage`

  If this package variable is set, then this package is `require()`d and an instance of this package is returned by calling the `new()` class method in that package. If it cannot be loaded or there is an error, an exception will be thrown. The variable can also contain a version number:

  ```
  $XML::SAX::ParserPackage = "XML::SAX::Expat (0.72)";
  ```

  And the number will be treated as a minimum version number.

- Required features

  It is possible to require features from the parsers. For example, you may wish for a parser that supports validation via a DTD. To do that, use the following code:

  ```
  use XML::SAX::ParserFactory;
  my $factory = XML::SAX::ParserFactory->new();
  $factory->require_feature('http://xml.org/sax/features/validation');
  my $parser = $factory->parser(...);
  ```

  Alternatively, specify the required features in the call to the ParserFactory constructor:

  ```
  my $factory = XML::SAX::ParserFactory->new(
          RequiredFeatures => {
               'http://xml.org/sax/features/validation' => 1,
               }
          );
  ```

  If the features you have asked for are unavailable (for example the user might not have a validating parser installed), then an exception will be thrown.

  The list of known parsers is searched in reverse order, so it will always return the last installed parser that supports all of your requested features (Note: this is subject to change if someone comes up with a better way of making this work).

- SAX.ini

  ParserFactory will search `@INC` for a file called SAX.ini, which is in a simple format:

  ```
  # a comment looks like this,
  ; or like this, and are stripped anywhere in the file
  key = value # SAX.in contains key/value pairs.
  ```

All whitespace is non-significant.

This file can contain either a line:

```
ParserPackage = MyParserModule (1.02)
```

Where MyParserModule is the module to load and use for the parser, and the number in brackets is a minimum version to load.

Or you can list required features:

```
http://xml.org/sax/features/validation = 1
```

And each feature with a true value will be required.

- Fallback

  If none of the above works, the last parser installed on the user's system will be used. The XML::SAX package ships with a pure perl XML parser, XML::SAX::PurePerl, so that there will always be a fallback parser.

## AUTHOR

Matt Sergeant, matt@sergeant.org

## LICENSE