NAME

Locale::Language - ISO two letter codes for language identification (ISO 639)

SYNOPSIS

```
    use Locale::Language;

    $lang = code2language('en');        # $lang gets 'English'
    $code = language2code('French');    # $code gets 'fr'

    @codes   = all_language_codes();
    @names   = all_language_names();
```

DESCRIPTION

The `Locale::Language` module provides access to the ISO two-letter codes for identifying languages, as defined in ISO 639. You can either access the codes via the *conversion routines* (described below), or via the two functions which return lists of all language codes or all language names.

CONVERSION ROUTINES

There are two conversion routines: `code2language()` and `language2code()`.

code2language()

This function takes a two letter language code and returns a string which contains the name of the language identified. If the code is not a valid language code, as defined by ISO 639, then `undef` will be returned.

```
    $lang = code2language($code);
```

language2code()

This function takes a language name and returns the corresponding two letter language code, if such exists. If the argument could not be identified as a language name, then `undef` will be returned.

```
    $code = language2code('French');
```

The case of the language name is not important. See the section *KNOWN BUGS AND LIMITATIONS* below.

QUERY ROUTINES

There are two function which can be used to obtain a list of all language codes, or all language names:

`all_language_codes()`

Returns a list of all two-letter language codes. The codes are guaranteed to be all lower-case, and not in any particular order.

`all_language_names()`

Returns a list of all language names for which there is a corresponding two-letter language code. The names are capitalised, and not returned in any particular order.

EXAMPLES

The following example illustrates use of the `code2language()` function. The user is prompted for a language code, and then told the corresponding language name:

```
    $| = 1;    # turn off buffering

    print "Enter language code: ";
    chop($code = <STDIN>);
    $lang = code2language($code);
    if (defined $lang)
    {
        print "$code = $lang\n";
    }
    else
    {
        print "’$code’ is not a valid language code!\n";
    }
```

KNOWN BUGS AND LIMITATIONS

- In the current implementation, all data is read in when the module is loaded, and then held in memory. A lazy implementation would be more memory friendly.
- Currently just supports the two letter language codes - there are also three-letter codes, and numbers. Would these be of any use to anyone?

SEE ALSO

Locale::Country

ISO codes for identification of country (ISO 3166). Supports 2-letter, 3-letter, and numeric country codes.

Locale::Script

ISO codes for identification of written scripts (ISO 15924).

Locale::Currency

ISO three letter codes for identification of currencies and funds (ISO 4217).

ISO 639:1988 (E/F)

Code for the representation of names of languages.

http://lcweb.loc.gov/standards/iso639-2/langhome.html

Home page for ISO 639-2.

AUTHOR

Neil Bowers <neil@bowers.com>

COPYRIGHT