

Grammatica NET Component

Verion 6.8 - March, 9th 2007

Caution: this component only works with version 2 or higher of the .NET Framework

The component defines the namespace GrammaticaNET which contains two classes:

- **Grammatica** which is the main class and
- **Error** which is the class representing one error returned by the grammar or the spelling checker functions.

History

6.6 first version

6.7 Modifications since 6.6





- removal of loadRessources
- new property: lastExceptionLog
- new functions: GRAddIgnoredWord, GRClearIgnoredWords, GRRemoveIgnoredWord, SFGAnalyseWordLex, SFGInsertLex, SFGChangeLex, SFGGetListWordsLex, SFGGetValuesWordLex, and SFGRemoveLex
- new constants category : "Part of Speech", "Gender of a word", "number of a word" and "SFGGetListWordsLex"
- new class: LexWord
- new class: GrammaticaSharedLexicon
- new constant: GRLexSharedLexiconDifferentName

6.8 Modifications since 6.7

- new functions: GRIsFunctionRegistered
- new constants category: GRIsFunctionRegistered

Class Grammatica

Properties:

-  CheckingLanguage
-  InterfLanguage
-  lastExceptionLog
-  OptionNonUSUK

CheckingLanguage

The CheckingLanguage property can be set to:

Grammatica.GREnglish

Or

Grammatica.GRFrench

The CheckingLangage property sets the checking language.

InterfLanguage

The InterfLanguage property can be set to:

Grammatica.GREnglish

Or

Grammatica.GRFrench

The InterfLanguage property sets the language of the message.

lastExceptionLog

This property returns the log of any exception that may have occurred during the execution of the last invoked Grammatica function. If no exception occurs during the execution of the last invoked function, it returns an empty string. Note that you can get the value of an exception only once since (further calls return an empty string)

OptionNonUSUK

The OptionNonUSUK can be set to:

GRNonUKoption

GRNonUSoption

GRUSUKoption

GRUSUKoption -> accepts American and British spellings

GRNonUKoption -> accepts American spellings and rejects British spellings

GRNonUSoption -> accepts British spellings and rejects American spellings

GR Functions:

- ⇒ GRAddIgnoredWord(System::String)
- ⇒ Grammatica()
- ⇒ GRClearIgnoredWords()
- ⇒ GRCloseLex(System::String)
- ⇒ GRConjugate(System::String, System::Int16, System::String[]@, System::Int16, System::Int16)
- ⇒ GRInfinitive(System::String, System::String[]@, System::Int16, System::Int16)
- ⇒ GRInsertLex(System::String)
- ⇒ GROpenLex(System::String)
- ⇒ GRPlural(System::String, System::String[]@, System::Int16, System::Int16)
- ⇒ GRRemoveIgnoredWord(System::String)
- ⇒ GRRemoveLex(System::String)
- ⇒ GRSetRegistrationKeys(System::String[])
- ⇒ GRTestGrammarParagraph(System::String, System::Int32, System::Int32, GrammaticaNET::Error[]@, System::Int32, System::Int16, System::Int16)
- ⇒ GRTestSpellingParagraph(System::String, System::Int32, System::Int32, GrammaticaNET::Error[]@, System::Int16, System::Int16, System::Int16)

Grammatica()

This method is the constructor of the Grammatica class. It is called when you create a new instance of the class.

All the other functions are instance methods.

IMPORTANT NOTICE : If you want to call Grammatica functions in different threads, you should create one specific instance per thread

System.Int16 GRSetRegistrationKeys(System.String[] keys)

This function allows one to enter activation keys for the Grammatica Component. The keys parameter contains an array of Unicode strings holding the keys. This function returns the number of valid keys accepted by the component.

System.Int16 GRIsFunctionRegistered(System.String function)

This function allows one to know if a function is registered in the Grammatica library. The function is a Unicode string defined in the GRIsFunctionRegistered category. The return of the function is 1 if the function is registered 0 otherwise.

System.Void GRTestSpellingParagraph(System.String theParagraph, System.Int32 indexSelStart, System.Int32 indexSelEnd, GrammaticaNET.Error[] paragraphErrors, System.Int16 deepSearch, System.Int16 stopUnknowCapWord, System.Int16 accentsOnCapital)

Spell checking of a text.

theParagraph contains the text to correct in the form of a Unicode string
indexSelStart contains the index of the first character of the selection (starting from zero)
indexSelEnd contains the index of the last character of the selection
ParagraphTab is an array containing the errors returned. Each element of the array is an instance of the Error class which is described below
deepSearch equals one if one wishes to launch the extended search for suggestions corresponding to the spelling errors ; otherwise zero.
stopUnknowCapWord equals one if one wishes to flag unknown words beginning with a capital or a digit ; otherwise zero.
accentsOnCapital equals one if one wishes to require that capitals show accents (when the letter is accented) ; otherwise zero.

System.Void GRTestGrammarParagraph(System.String theParagraph, System.Int32 indexSelStart, System.Int32 indexSelEnd, GrammaticaNET.Error[] paragraphErrors, System.Int32 mistakesMask, System.Int16 deepSearch, System.Int16 stopUnknowCapWord, System.Int16 accentsOnCapital)

Grammar checking of a text.

theParagraph contains the text to correct in the form of a Unicode string
indexSelStart contains the index of the first character of the selection (starting from zero)
indexSelEnd contains the index of the last character of the selection
ParagraphTab is an array containing the errors returned. Each element of the array is an instance of the Error class which is described below
MistakesMask allows one to filter the errors that the user does not wish to receive (see the description of constants to be used in the Mistake Mask category below).
deepSearch equals one if one wishes to launch the extended search for suggestions corresponding to the spelling errors ; otherwise zero.
stopUnknowCapWord equals one if one wishes to flag unknown words beginning with a capital or a digit ; otherwise zero.
accentsOnCapital equals one if one wishes to require that capitals show accents (when the letter is accented) ; otherwise zero.

System.Int16 GRConjugate(System.String theWord, System.Int16 mode, System.String[] solutions, System.Int16 accentsOnCapital, System.Int16 language)

This function sends back the conjugations of a verb infinitive.

theWord contains the infinitive in the form of a Unicode string
mode is among the constants defined in the Conjugation mode constants category
solutions contains in the return the solutions. It is an array of Unicode strings.
accentsOnCapital equals one if one wishes to require that accented capitals have their accents ; otherwise zero
language allows to specify the language : either GREnglish or GRFrench

This function returns one of the constants defined in the Conjugate, Infinitive and Plural constants category (see its description below) with the exception of: `GErrorInvariable`, `GErrorNoPluralForm`, `GErrorVerbFormNoPluralForm`, `GErrorProperNounNoPluralForm` or `GRVariableNounAdjective`

System.Int16 GRInfinitive(System.String theWord, System.String[] solutions, System.Int16 accentsOnCapital, System.Int16 language)

This function sends back the infinitive(s) of a conjugated verb form.

`theWord` contains the conjugated verb form in the form of a Unicode string.
`solutions` contains in the return the solutions. It is an array of Unicode strings.

`accentsOnCapital` equals one if one wishes to require that accented capitals have their accents ; otherwise zero

`language` allows to specify the language : either `GREnglish` or `GRFrench`

This function returns one of the constants defined in the Conjugate, Infinitive and Plural constants category (see its description below) with the exception of: `GErrorInvariable`, `GErrorNoPluralForm`, `GErrorVerbFormNoPluralForm`, `GErrorProperNounNoPluralForm` or `GRVariableNounAdjective`

System.Int16 GRPlural(System.String theWord, System.String[] solutions, System.Int16 accentsOnCapital, System.Int16 language)

This function sends back the plural(s) of a noun or adjective.

`theWord` contains the noun or adjective in the form of a Unicode string
`solutions` contains in the return the solutions. It is an array of Unicode strings.

`accentsOnCapital` equals one if one wishes to require that accented capitals have their accents ; otherwise zero

`language` allows to specify the language : either `GREnglish` or `GRFrench`

This function returns amongst the constants defined in the Conjugate, Infinitive and Plural constants category (see its description below): `GErrorInvariable`, `GErrorNoPluralForm`, `GErrorVerbFormNoPluralForm`, `GErrorProperNounNoPluralForm` or `GRVariableNounAdjective`

System.Int16 GROpenLex(System.String fileName)

Opening of the lexicon. It will be created if it does not already exist.

The parameter `filename` contains the filename of the Lexicon file in the form of a Unicode string

This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): `GRLexNoErr`, `GRLexIOError`, `GRLexMemoryError` or `GRLexIncorrectVersion`

System.Int16 GRCloseLex(System.String fileName)

Closing the lexicon.

The parameter `filename` contains the filename of the Lexicon file in the form of a Unicode string

This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): GRLexNoErr or GRLexIOError

System.Int16 GRInsertLex(System.String word)

This function allows one to add a word to the lexicon of Grammatica in the form of a Unicode string.

The parameter word contains the word to insert.

The return of the function can return one of the constants defined in the Lexicon errors constants category (see its description below) with the exception of GRLexIncorrectVersion, GRLexSharedLexiconDifferentName and GRLexWordNotFound

System.Int16 GRRemoveLex(System.String word)

This function can delete a word from the lexicon of Grammatica in the form of a Unicode string.

The parameter word contains the word to delete.

This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): GRLexNoErr, GRLexErrorInLexicon, GRLexNotOpen, GRLexWordNotFound or GRLexIOError

System.Void GRAddIgnoredWord(System.String word)

This function adds the word (in the form of a Unicode string) to the list of ignored words. When a spelling mistake occurs for one of the words of this list, this spelling mistake is not returned by GRTestSpellingParagraph or GRTestGrammarParagraph.

If the list already contains the word, this function does nothing.

System.Void GRClearIgnoredWords()

This function clears all the words contained in the list of ignored words.

System.Void GRRemoveIgnoredWord(System.String word)

This function removes the word (in the form of a Unicode string) from the list of ignored words.

If the list does not contain the word, this function does nothing.

SFG Functions:

- 🍷 SFGAnalyseWordLex(System::String, System::Int16@, System::Int16@, System::Int16@, System::Int16@, System::Int16@, System::Str
- 🍷 SFGChangeLex(System::String, System::Int16, System::Int16, System::Int16, System::Int16, System::Int16, System::String, System::Str
- 🍷 SFGConjugateLex(System::String, System::Int16, System::String[]@, System::Int16, System::String, System::Int16)
- 🍷 SFGGetListWordsLex(GrammaticaNET::LexWord[]@, System::Int32)
- 🍷 SFGGetValuesWordLex(System::String, System::Int16, System::Int16@, System::Int16@, System::Int16@, System::Int16@, System::Str
- 🍷 SFGInsertLex(System::String, System::Int16, System::Int16, System::Int16, System::Int16, System::Int16, System::String, System::Str
- 🍷 SFGRemoveLex(System::String, System::Int16, System::Int16)

These functions allow one to use all the possibilities of Grammatica lexicon.

System.Int16 SFGAnalyseWordLex(System.String theWord, System.Int16 logo, System.Int16 type, System.Int16 gender, System.Int16 pluralForm, System.Int16 intransitiveVerb, System.String modelConj, System.Int16 language)

Automatic analysis of a word in order to automatically propose choices to the end user

- The parameter `theWord` contains the word to analyse (in the form of a Unicode string).
 - `logo` contains 1 upon return if the word is proposed as an acronym ; otherwise zero. This parameter is only valid if `type` equals `SFGProperNoun`.
 - `type` contains upon return the proposed type (part of speech) for the word. It is one of the constants of the "Part of Speech" category (see below).
 - `gender` contains upon return the proposed gender of the word. It is one of the constants of the "gender of a word" category (see below).
 - `pluralForm` contains upon return the proposed number (singular or plural) of the word. It is one of the constants of the "number of a word" category (see below).
 - `intransitiveVerb` equals 1 if the proposed verb is transitive ; otherwise 0. The type must be `SFGVerb` and the language must be `GRFrench` for this parameter to be useful.
 - `modelConj` proposes a conjugation model (in the form of a Unicode string). Type must equal `SFGVerb` for this parameter to be useful
 - `language` allows to specify the language : either `GREnglish` or `GRFrench`
- This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): `GRLexNoErr`, `GRLexIOError` or `GRLexAlreadyInDict`

System.Int16 SFGConjugateLex(System.String theWord, System.Int16 mode, System.String[] solutions, System.Int16 intransitiveVerb, System.String modelConj, System.Int16 language)

- `SFGConjugateLex` allows one to conjugate a verb not yet added to the lexicon.
 - `theWord` contains the infinitive (in the form of a Unicode string)
 - `mode` is among the constants defined in the Conjugation mode constants category
 - `solutions` contains in the return the solutions. It is an array of Unicode strings
 - `intransitiveVerb` equals 1 if the proposed verb is transitive ; otherwise 0.
 - `modelConj` is the proposed conjugation model (in the form of a Unicode string)
 - `language` allows one to specify the language : either `GREnglish` or `GRFrench`
- This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): `GRLexNoErr`, `GRLexIOError`, `GRLexUnknownConjModel`, `GRLexConjModelNotVerb`, `GRLexConjModelNotVerbSameConj`, `GRLexConjModelNotInfinitive`, `GRLexPastParticipleInvariable`, `GRLexConjModelNotValid` or `GRLexAlreadyInDict`

System.Int16 SFGInsertLex(System.String theWord, System.Int16 logo, System.Int16 type, System.Int16 gender, System.Int16 pluralForm, System.Int16 intransitiveVerb, System.String modelConj, System.String abbrev, System.Int16 language)

- `SFGInsertLex` allows one to add a word to the lexicon of Grammatica
- `theWord` contains the word to insert (in the form of a Unicode string)
- `logo` contains one if the proper noun is to be added as an acronym ; otherwise 0.
- `type` contains the word type (part of speech) to be added, using the constants defined in the "Part of Speech" category
- `gender` contains the gender of the word to be added, using the constants defined in the "gender of a word" category
- `pluralForm` contains the number (singular or plural) of the word to be added, using the constants defined "number of a word" category

- `intransitiveVerb` equals 1 if the proposed verb is transitive ; otherwise 0. The type must be `SFGVerb` and the language must be `GRFrench` for this parameter to be useful
- `modelConj` contains the conjugation model to use. Type must equal `SFGVerb` for this parameter to be useful
- `abbrev` contains the abbreviation of the proper noun. This parameter is only valid if type equals `SFGProperNoun`.
- `language` allows to specify the language : either `GREnglish` or `GRFrench`

The return of the function can return one of the constants defined in the Lexicon errors constants category (see its description below) with the exception of `GRLexIncorrectVersion`, `GRLexSharedLexiconDifferentName` and `GRLexWordNotFound`

The case of the error `GRLexAlreadyDefined` is special. In this case one must ask the user if he wishes to modify the lexicon by erasing the pre-existing form. If he responds OK, one must call the function `SFGChangeLex`

Important comments:

In English, adjectives necessarily have a number equal to `SFGIrregular`, which is to say that they are always invariable.
 In English, nouns and adjectives have no gender. One must use the gender `SFGMasculineFeminine` (both masculine and feminine).

Only proper nouns (type = `SFGProperNoun`) can be acronyms (logos). An acronym is a proper noun that may contain spaces, punctuation marks, and the typography of which must be respected in a text. If a proper noun is an acronym, its number can only be `SFGIrregular`, `SFGPlural` or `SFGSingularPlural`.
 One can only associate abbreviations to proper nouns (type = `SFGProperNoun`).

Proper nouns are defined in French. The engine recognizes French proper nouns during French or English grammar and spell-checking.

System.Int16 SFGChangeLex(System.String theWord, System.Int16 logo, System.Int16 type, System.Int16 gender, System.Int16 pluralForm, System.Int16 intransitiveVerb, System.String modelConj, System.String abbrev, System.Int16 language)

`SFGChangeLex` applies only when `SFGInsertLex` has sent back `GRLexAlreadyDefined` and the user has decided to modify a pre-existing word in the lexicon. The function then takes exactly the same parameters as those used in the calling of `SFGInsertLex` which has returned `GRLexAlreadyDefined`

This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): `GRLexNoErr`, `GRLexErrorInLexicon`, `GRLexMemoryError`, `GRLexNotOpen` or `GRLexIOError`

System.Int16 SFGGetListWordsLex(GrammaticaNET.LexWord[] lexicon, System.Int32 languageSelection)

`SFGGetListWordsLex` sends back the list of the lexicon's words. This list is sorted in lexicographical order.

- `lexicon` contains upon return the list of words as an array of instances of the `LexWord` class (see the definition of the `LexWord` class below)
- `languageSelection` indicates the language of the words which one desires to get in return. It can equal:
 - `SFGEnglishSelection` to return only English words;

- SFGFrenchSelection to return only French words;
- SFGEnglishSelection+SFGFrenchSelection to return both French and English words;
- SFGallLanguagesSel to return all words in all languages

This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): GRLexNoErr, GRLexMemoryError or GRLexIOError

Careful : proper nouns are always returned, regardless of the language selected. The attribute Language in LexWord thus equals GRFrench

System.Int16 SFGGetValuesWordLex(System.String theWord, System.Int16 type, System.Int16 logo, System.Int16 gender, System.Int16 pluralForm, System.Int16 intransitiveVerb, System.String modelConj, System.String abbrev, System.Int16 language)

- SFGGetValuesWordLex allows one to get the values of a word from the lexicon, giving it along with its type.
 - theWord contains the word to be searched into the lexicon in the form of a Unicode string
 - type contains the word type (part of speech), using the constants defined in the "Part of Speech" category
 - logo contains 1 upon return if the word is an acronym ; otherwise zero. This parameter is only valid if type equals SFGProperNoun.
 - gender contains upon return the gender of the word, using the constants defined in the "gender of a word" category
 - pluralForm contains upon return the number (singular or plural) of the word, using the constants defined in the "number of a word" category
 - intransitiveVerb equals 1 upon return if the verb is transitive ; otherwise 0. The type must be SFGVerb and the language must be GRFrench for this parameter to be useful
 - modelConj contains upon return the conjugation model to use in the form of a Unicode string. Type must equal SFGVerb for this parameter to be useful
 - abbrev contains upon return the abbreviation of the proper noun in the form of a Unicode string. This parameter is only valid if type equals SFGProperNoun
 - language allows to specify the language : either GREnglish or GRFrench
- This function returns amongst the constants defined in the Lexicon errors constants category (see its description below): GRLexNoErr, GRLexWordNotFound or GRLexIOError

System.Int16 SFGRemoveLex(System.String theWord, System.Int16 type, System.Int16 language)

SFGRemoveLex can delete a word from the lexicon, giving it along with its type

- > theWord contains the word to delete in the form of a Unicode string
- > type contains the word type, using the constants defined in the "Part of Speech" category
- > language allows to specify the language : either GREnglish or GRFrench

<- return: the return of the function can equal GRLexNoErr, GRLexErrorInLexicon, GRLexNotOpen, GRLexWordNotFound or GRLexIOError

Class Error

- ◆ errorIndexSelEnd
- ◆ errorIndexSelStart
- ◆ message
- ◆ suggestions

System.Int32 errorIndexSelEnd

errorIndexSelEnd is the index of the last character to select to flag the error

System.Int32 errorIndexSelStart

errorIndexSelStart is the index of the first character to select to flag the error

System.String message

message which is the message describing the error (which is a Unicode string)

System.String[] suggestions

suggestions which is the table of the suggestions. It is an array of Unicode strings.

Class LexWord

- ◆ abbrev
- ◆ language
- ◆ logo
- ◆ typeWord
- ◆ word

System.String abbrev

abbrev contains the abbreviation of a word (possible only if the word is a proper noun)

System.Int16 language

language contains GREnglish or GRFrench. However, for proper nouns, it is alwaysGRFrench.

System.Int16 logo

logo contains 1 if the word is a logo ; otherwise 0.

System.Int16 typeWord

typeWord contains the word type (part of speech) of the word (see the constants defined in the Part of Speech category below)

System.String word

word contains the word added to the lexicon

Class GrammaticaSharedLexicon

GrammaticaSharedLexicon is a direct subclass of Grammatica. It owns the same API as Grammatica. The difference resides in the fact that the lexicon file is shared amongst all the instances while each instance of Grammatica handles its own lexicon file.

The first time, GROpenLex is called, it opens the shared lexicon. Consequent calls to GROpenLex have the following effects:

- if the same file is specified, an internal counter containing the number of openings is incremented by one
- if another file is specified, GROpenLex returns GRLexSharedLexiconDifferentName

The shared lexicon file will be closed when the same number of GRCloseLex than of GROpenLex will be reached. The number of openings is decremented by one each time GRCloseLex is called.

GrammaticaSharedLexicon uses the "multiple readers/one writer" lock in order to share the lexicon.

CONSTANTS

Language category

GRFrench -> French
GREnglish -> English

These constants are used by the CheckingLanguage and the InterfLanguage properties of the Grammatica class

USUKOPTION category

GRUSUKoption -> accepts American and British spellings
GRNonUKoption -> accepts American spellings and rejects British spellings
GRNonUSoption -> accepts British spellings and rejects American spellings

Mistake Mask category

GRallErrorMask -> Returns all errors in the category of optional errors
GRcomplexPredicatePhrasesErrorMask -> Overly heavy subordination
GRduplicatedWordErrorMask -> duplicate word
GRincompleteNegationFormErrorMask -> Missing « ne » (in a French negative construction)
GRintransitiveErrorMask -> intransitivity mistakes
GRligaturesErrorMask -> Ligatures
GRlongSentenceErrorMask -> Excessively long sentence
GRmissingInterrogativeFormErrorMask -> Missing interrogative form
GRmissingQuestionMarkErrorMask -> Missing question mark
GRmissingVerbErrorMask -> Missing verb
GRquestionableConstructionErrorMask -> Problematic construction
GRrepeatedWordErrorMask -> Repeated words
GRtooManyRelativePronounsErrorMask -> Three or more relative pronouns
GRunCapitalizedProperNounErrorMask -> Proper nouns missing capitals
GRwidowedApostropheErrorMask -> Balance of apostrophes (inverted commas)
GRwidowedBracketErrorMask -> Balance of square brackets
GRwidowedCurlyBraceErrorMask -> Balance of curly brackets

GRwidowedParenthesesErrorMask -> Balance of parentheses
GRwidowedQuotationMarkErrorMask -> Balance of quotation marks
GRcomplexPredicatePhrasesErrorMask -> Overly heavy subordination
GRnonStandardVocabularyErrorMask -> non standard vocabulary (slang, colloquial, unsophisticated, literary or archaic)

Conjugation mode category

GRConditional -> conditional
GREnglishVerb -> English mode
GRFuture -> future indicative
GRImperative -> imperative
GRImperfect -> imperfect indicative
GRImperfectSubjunctive -> imperfect subjunctive
GRPastParticiple -> past participle
GRPresent -> present indicative
GRPresentParticiple -> present participle
GRPresentSubjunctive -> present subjunctive
GRSimplePast -> simple past indicative

Conjugate, Infinitive and Plural category

GRAuxiliary -> Auxiliary verb
GRCommonInfinitive -> Verb commonly used only in the infinitive
GRCommonInfinitivePastParticiple -> Verb commonly used only in the infinitive and past participle
GRCommonPlural -> Verb commonly used only in the plural
GRCommonThirdPerson -> Verb commonly used only in the 3rd person
GRDefective -> Defective verb
GRErrorInvariable -> Invariable or already in the plural form
GRErrorNoPluralForm -> No plural form
GRErrorNotInfinitive -> This word is not recognized as a verb in the infinitive
GRErrorNotVerbForm -> The word is not a verb form
GRErrorProperNounNoPluralForm -> Proper noun : no plural
GRErrorUnknownWord -> Word not recognized by Grammatica
GRErrorVerbFormNoPluralForm -> Verb form : no plural
GRFirstConjugation -> Regular verb of the first conjugation (French)
GRImpersonal -> Impersonal verb (uses « il » or « it » only)
GRIrregularEnglish -> Irregular verb in English
GRRegularEnglish -> Regular verb in English
GRSecondConjugation -> Regular verb of the second conjugation (French)
GRThirdConjugation -> Verb of the third conjugation (French)
GRVariableNounAdjective -> Variable noun or adjective
GRVeryDefective -> Very defective verb

Lexicon errors category

GRLexAlreadyDefined -> This word has already been added into the lexicon !
GRLexAlreadyInDict -> This word has not been defined as a proper noun, and it is present in the corpus. Please choose another !
GRLexAtLeastTwoLetters -> A word must contain at least two letters
GRLexConflictWithAbbr -> This word has already been defined as an abbreviation in the lexicon. Please choose another !
GRLexErrorInLexicon -> The modification could not be recorded because of an internal problem in the handling of the lexicon
GRLexIncorrectVersion -> This version of the lexicon is not compatible with Grammatica
GRLexIOError -> Input/output error in the lexicon
GRLexMemoryError -> Insufficient memory

GRLexNoErr -> No error
GRLexNoInvalidChar -> A word may not contain invalid characters
GRLexNoLigatureInEnglish -> Ligatures may not be used in English
GRLexNoMoreTwoLigatures -> This word may not contain two ligatures
GRLexNotOpen -> No lexicon is open !
GRLexSpaceNotLogo -> A word containing one or more spaces must be an acronym
GRLexTooCloseToFourLogos -> This word is too similar to four others already defined as an acronym. Please choose another !
GRLexTooCommonWord -> This word is part of a list of frequently used words and may not be entered as a proper noun or an acronym. Please choose another !
GRLexWordNotFound -> This word cannot be deleted, for it does not exist in the lexicon !
GRLexWordTooLong -> A word may contain a maximum of 24 characters.

Specific error returned by lexicon handling functions of GrammaticaSharedLexicon

GRLexSharedLexiconDifferentName -> a shared lexicon is already open with a different file.

Part of speech category

SFGProperNoun -> Proper noun
SFGCommonNoun -> Common noun
SFGAdjective -> Adjective
SFGVerb -> Verb
SFGAdverb -> Adverb

gender of a word category

SFGMasculine -> Masculine
SFGFeminine -> Feminine
SFGMasculineFeminine -> Masculine and Feminine

number of a word category

SFGSingularPluralInS -> Singular with a plural ending in « s » (English and French)
SFGIrregular -> Singular without a plural form, or with an irregular plural (English and French)
SFGPlural -> Plural (English and French)
SFGSingularPlural -> Singular and plural (English and French)
SFGSingularPluralInES -> Singular with a plural ending in « es » (English)
SFGSingularPluralInX -> Singular with a plural ending in « x » (French only)
SFGSingularPluralInIES -> Singular with a plural ending in « ies » (English only)
SFGSingularPluralInMEN -> Singular with a plural ending in « men » (English only)
SFGSingularPluralInAEandS -> Singular with a plural ending in « ae » and « s » (English only)

SFGGetListWordsLex category

SFGEnglishSelection -> SFGGetListWordsLex returns only English words (including proper nouns)
SFGFrenchSelection -> SFGGetListWordsLex returns only French words (including proper nouns)

SFGallLanguagesSel -> SFGGetListWordsLex returns all words

GRIFunctionRegistered category

GRFrenchSpelling
GRFrenchGrammar
GREnglishSpelling
GREnglishGrammar
GRDictionary
GRStemming
GRSentenceAnalysis
GREnglishMedical
GRSpanishSpelling
GRSpanishGrammar
GRGermanSpelling
GRGermanGrammar

How to use the spelling checker in VB.NET :

You must add a reference to the Grammatica .NET component (to the “GrammaticaNET.dll” DLL)

```
Dim gramChecker As GrammaticaNET.Grammatica
Dim errors As GrammaticaNET.Error()
Dim mistake As GrammaticaNET.Error
Dim phrase, suggestion As String

gramChecker = New GrammaticaNET.Grammatica()
gramChecker.CheckingLanguage = GrammaticaNET.Grammatica.GREnglish
gramChecker.InterfLanguage = GrammaticaNET.Grammatica.GREnglish

phrase = "didd you eate chocalote"
gramChecker.GRTestSpellingParagraph(phrase, 0, phrase.Length, errors, 1, 0, 0)
For Each mistake In errors
    Console.WriteLine(mistake.errorIndexSelStart)
    Console.WriteLine(mistake.errorIndexSelEnd)
    Console.WriteLine(mistake.message)
    For Each suggestion In mistake.suggestions
        Console.WriteLine(suggestion)
    Next
Next mistake
```

How to use the grammar checker in VB.NET :

You must add a reference to the Grammatica .NET component (to the “GrammaticaNET.dll” DLL)

```
Dim gramChecker As GrammaticaNET.Grammatica
Dim errors As GrammaticaNET.Error()
Dim mistake As GrammaticaNET.Error
Dim phrase, suggestion As String

gramChecker = New GrammaticaNET.Grammatica()
gramChecker.CheckingLanguage = GrammaticaNET.Grammatica.GREnglish
gramChecker.InterfLanguage = GrammaticaNET.Grammatica.GREnglish

phrase = "Bob eat an apples"
gramChecker.GRTTestGrammarParagraph(phrase, 0, phrase.Length, errors,
GrammaticaNET.Grammatica.GRallErrorMask, 1, 0, 0)
For Each mistake In errors
    Console.WriteLine(mistake.errorIndexSelStart)
    Console.WriteLine(mistake.errorIndexSelEnd)
    Console.WriteLine(mistake.message)
    For Each suggestion In mistake.suggestions
        Console.WriteLine(suggestion)
    Next
Next mistake
```