Branko Zečević | 2024-12

Go Internationalization (i18n)

Introduction to text translation and data formatting

Internationalization and localization

Internationalization (i18n):

- design and development of software to enable adaption (ie, localization) to language, regional and cultural conventions (target locale)

Localization (l10n):

- adaptation of internationalized software for specific region or language
- commonly includes: text translation, implementation of locale-specific components

Locale:

- set of parameters that defines language, region and related preferences

Locale identifier:

- language code, country/region code

Locale settings

Common settings:

- Number format
- Character classification, case conversion
- Date-time format
- String collation
- Currency format
- Paper size
- Color
- Location (country or region)

  …

Standard locale data:

- Common Locale Data Repository (CLDR), https://cldr.unicode.org/

Go i18n

Package text:

- https://pkg.go.dev/golang.org/x/text
- repository of text-related packages related to internationalization (i18n) and localization (l10n)
- character encodings, text transformations, locale-specific text handling, ...

Selected text/* packages:

- encoding: interface for character encodings
- language: BCP 47 language tags
- message: formatted I/O for localized strings
- number: formats numbers according to locale
- runes: transforms for UTF-8 encoded text
- unicode: implementations of Unicode standards

...

Go i18n (2)

Package language:

- https://pkg.go.dev/golang.org/x/text/language
- implements BCP 47 language tags and related functionality
- implements functions to parse language tag, match tag to list of supported languages, ...

Package message:

- https://pkg.go.dev/golang.org/x/text/message
- formatted I/O for localized strings
- implements type Printer for language-specific formatted I/O
- implements replacement for fmt *Printf functions
  (Printer).Printf, (Printer).Sprintf(), ...

Translation

Concept:

- print messages with packages text/language, text/message
- use gotext tool to:
  - extract messages for translation from code
  - parse translated files (JSON)
  - create catalog with translated messages

Message:

- simple message (text only)
- message with variable reference
- (optional) pluralized version of translated message

Translation (3)

gotext

- https://pkg.go.dev/golang.org/x/text/cmd/gotext
- tool to manage text in Go source code

Features:

- merge translations and generate catalog
- extract strings to be translated from code
- rewrites fmt functions to use message printer
- generate code to insert translated messages

Setup:

- go install golang.org/x/text/cmd/gotext@latest

Call:

- executed from go generate
- refers to *Printf functions: Printf(), Sprintf(), Fprintf()

Translation (4)

Workflow:

- create application package for translation
  res/translation/translation.go
- include go generate command to execute gotext ‘update’
- build with go generate (gotext):
  - gotext examines code, searches for call to message.Printer
  - gotext extracts message string
  - gotext outputs catalog file (w/ init function)
  - gotext outputs locale JSON file for translation (ie, out.gotext.json)
- translate message file:
  - store translation in ‘copy’ (ie, as messages.gotext.json)
- import application package for translation (w/ init function)
- rebuild with go generate (gotext)

Number formatting

Package fmt:

- https://pkg.go.dev/fmt
- no language-specific formatting
- implements *Printf functions
  fmt.Printf, fmt.Sprintf, fmt.Fprintf, ...

Package language/message:

- language-specific formatting
- implements *Printf functions
  (Printer).Printf, (Printer).Sprintf(), ...

Number formatting (2)

Package language/number:

- custom-specific formatting
- implements functions for Decimal, Percent, ...
- implements formatting options for width, padding, ...

Sample programs

i18n1

- test 1: parse language tag

i18n2

- with package res/translation, go generate
- test 2: init message printer

i18n3

- with package res/translation, go generate, no init :-(
- test 3: init message printer, use message

i18n4

- with package res/translation, go generate, init :-)
- test 4: init message printer, use message with variable
- test 5: format number for language
- test 6: format number for custom

Alternative packages

go-i18n:

- https://github.com/nicksnyder/go-i18n
- Features:
  - supports pluralized strings for 200+ languages in Unicode Common Locale Data Repository (CLDR)
  - Supports strings with named variables using text/template syntax
  - Supports message files of any format (e.g. JSON, TOML, YAML)

gotext:

- https://github.com/leonelquinteros/gotext
- Features:
  - GNU gettext utilities for Go
  - Implements GNU gettext support in native Go

References

i18n/l10n:

- https://en.wikipedia.org/wiki/Internationalization_and_localization

Locale:

- https://en.wikipedia.org/wiki/Locale_(computer_software)
- https://en.wikipedia.org/wiki/Common_Locale_Data_Repository

Language:

- https://en.wikipedia.org/wiki/IETF_language_tag (BCP 47)

Go software:

- https://pkg.go.dev/
- https://golangweekly.com/

Articles:

- https://golangweekly.com/issues/378

Thank you!

Contacts:

Branko Zečević, branko.zecevic@pointer.hr