Word Document Library for Go

Independent project · 2025 · Solo · Public repository · Open source (MIT)

A Go library that builds Word (.docx) files in code through a chainable API: text formatting, headings, lists, tables, images and hyperlinks.

Links: Repository (mbadocx) · Go package docs · Examples

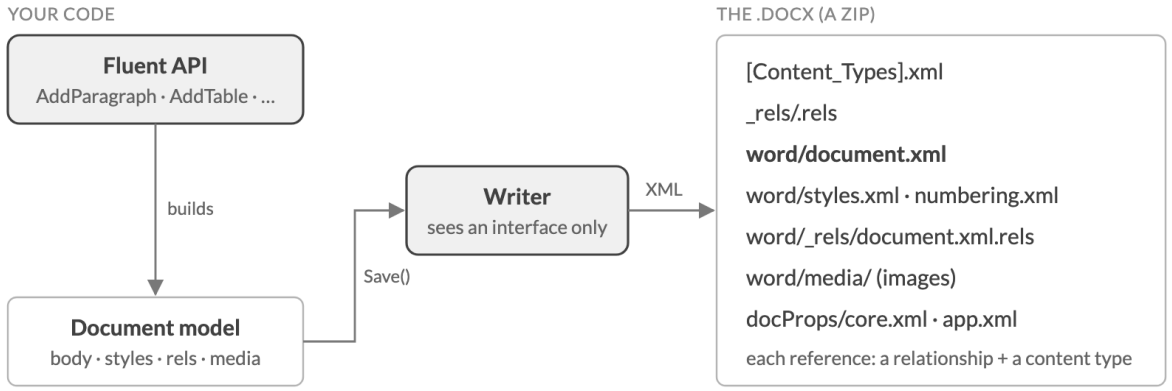
Why

Generating Word reports from a Go service usually means filling a template or calling a converter in another language. I wanted to build the document directly from Go code, and to understand the Office Open XML format underneath instead of treating it as a black box.

What it does

- Creates a new .docx from scratch and saves it as a valid Word package.
- Formats text: bold, italic, underline, strike, colour and font size.
- Adds headings, bulleted, numbered, legal and Roman lists, and line and page breaks.
- Builds tables with per-cell formatting and alignment.
- Embeds images with borders and alignment, and adds external hyperlinks.

Key flow



1. New() creates a document with its body, styles, relationships, content types and metadata.
2. Calls such as AddParagraph().AddText("…").SetBold(true) build the body as a tree of elements.
3. Images and hyperlinks register a relationship as they are added, so every reference has a target.
4. Save() marshals each part to XML and writes the package as a zip: document, styles, numbering, relationships, content types and properties.

Key decisions

Decision	Why / trade-off
Writer depends on an interface	The zip writer sees the document only through a small interface (body, styles, relationships, media). The model and the writer never import each other, at the cost of a few extra types.
Fluent setters	Every setter returns its receiver, so formatting reads as one chain. The trade-off is that a setter cannot also return an error.
Parts generated from Go structs	Every XML part is marshalled from typed structs with encoding/xml, not copied from a template file, so the output is fully controlled in code.
Relationships reused by target	Linking the same URL twice reuses one relationship instead of adding a duplicate.
Cell formatting through a callback	SetCellFormattedText takes a func(*Run), so a cell gets the full run API without a second set of table-only setters.

Non-goals

- Opening or editing existing .docx files: the library writes new documents.
- A template engine: documents are built in code, not from placeholders.

Outcome

- Published as a Go module and listed on pkg.go.dev, under the MIT licence.
- Five runnable examples (basic, text formatting, lists, tables, images), each producing a sample .docx kept in the repository.
- Lesson: a .docx is a zip of XML parts tied together by relationships and content types; every new element type has to keep all three in step, or Word refuses the file.

Code pointers

- writer/writer.go: how the package is assembled into a zip.
- types/types.go: the interface the writer depends on.
- elements/run.go: the chainable text formatting API.

Stack

Go · Office Open XML · encoding/xml · archive/zip