J-F-Liu / lopdf Public

Code | Issues 52 | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

master | Go to file | Code

wiktor-k Fix documentation issues and make README testable (#171) ... 17 hours ago 292

.chglog	Add Change Log	2 years ago
.github/workflows	Fix clippy warnings & add clippy build job (#128)	2 years ago
.vscode	Disable auto format markdown	4 years ago
assets	insert image on page	4 years ago
benches	Various improvements, updated libraries and image features (#118)	2 years ago
examples	Bookmarks (#135)	17 months ago
pdfutil	Fix pdfutil build error	29 days ago
src	Fix documentation issues and make README testable (#171)	17 hours ago
tests	Various improvements, updated libraries and image features (#118)	2 years ago
.editorconfig	Change indent_style to space	6 months ago
.gitignore	Fix documentation issues and make README testable (#171)	17 hours ago
.travis.yml	Also test with nom parsing feature enabled.	3 years ago
CHANGELOG.md	Add Change Log	2 years ago
Cargo.toml	remove unneccessary time 0.1 dependency (#163)	4 months ago
LICENSE	Initial commit	6 years ago
README.md	Fix documentation issues and make README testable (#171)	17 hours ago
rustfmt.toml	Various improvements, updated libraries and image features (#118)	2 years ago

README.md

crates.io v0.27.0 | build passing | docs passing

A Rust library for PDF document manipulation.

Example Code

- Create PDF document

```
use lopdf::dictionary;
use lopdf::{Document, Object, Stream};
use lopdf::content::{Content, Operation};

let mut doc = Document::with_version("1.5");
let pages_id = doc.new_object_id();
let font_id = doc.add_object(dictionary! {
    "Type" => "Font",
    "Subtype" => "Type1",
    "BaseFont" => "Courier",
});
let resources_id = doc.add_object(dictionary! {
    "Font" => dictionary! {
        "F1" => font_id,
    },
});
let content = Content {
    operations: vec![
        Operation::new("BT", vec![]),
        Operation::new("Tf", vec!["F1".into(), 48.into()]),
        Operation::new("Td", vec![100.into(), 600.into()]),
```

About

A Rust library for PDF document manipulation.

#rust #pdf-document #rust-library

Readme
MIT license
941 stars
16 watching
90 forks

Releases 20

0.20.0 Latest
on Mar 7, 2019

+ 19 releases

Packages

No packages published

Contributors 30

+ 19 contributors

Languages

Rust 100.0%

```
        Operation::new("Tj", vec![Object::string_literal("Hello World!")]),
        Operation::new("ET", vec![]),
    ],
};
let content_id = doc.add_object(Stream::new(dictionary! {}, content.encode().unwrap()));
let page_id = doc.add_object(dictionary! {
    "Type" => "Page",
    "Parent" => pages_id,
    "Contents" => content_id,
});
let pages = dictionary! {
    "Type" => "Pages",
    "Kids" => vec![page_id.into()],
    "Count" => 1,
    "Resources" => resources_id,
    "MediaBox" => vec![0.into(), 0.into(), 595.into(), 842.into()],
};
doc.objects.insert(pages_id, Object::Dictionary(pages));
let catalog_id = doc.add_object(dictionary! {
    "Type" => "Catalog",
    "Pages" => pages_id,
});
doc.trailer.set("Root", catalog_id);
doc.compress();
doc.save("example.pdf").unwrap();
```

- Merge PDF documents

```
use lopdf::dictionary;

use std::collections::BTreeMap;

use lopdf::content::{Content, Operation};
use lopdf::{Document, Object, ObjectId, Stream, Bookmark};

pub fn generate_fake_document() -> Document {
    let mut doc = Document::with_version("1.5");
    let pages_id = doc.new_object_id();
    let font_id = doc.add_object(dictionary! {
        "Type" => "Font",
        "Subtype" => "Type1",
        "BaseFont" => "Courier",
    });
    let resources_id = doc.add_object(dictionary! {
        "Font" => dictionary! {
            "F1" => font_id,
        },
    });
    let content = Content {
        operations: vec![
            Operation::new("BT", vec![]),
            Operation::new("Tf", vec!["F1".into(), 48.into()]),
            Operation::new("Td", vec![100.into(), 600.into()]),
            Operation::new("Tj", vec![Object::string_literal("Hello World!")]),
            Operation::new("ET", vec![]),
        ],
    };
    let content_id = doc.add_object(Stream::new(dictionary! {}, content.encode().unwrap()));
    let page_id = doc.add_object(dictionary! {
        "Type" => "Page",
        "Parent" => pages_id,
        "Contents" => content_id,
        "Resources" => resources_id,
        "MediaBox" => vec![0.into(), 0.into(), 595.into(), 842.into()],
    });
    let pages = dictionary! {
        "Type" => "Pages",
        "Kids" => vec![page_id.into()],
        "Count" => 1,
    };
    doc.objects.insert(pages_id, Object::Dictionary(pages));
    let catalog_id = doc.add_object(dictionary! {
        "Type" => "Catalog",
        "Pages" => pages_id,
    });
    doc.trailer.set("Root", catalog_id);

    doc
}

fn main() -> std::io::Result<()> {
```