LynPDF RS API Guide v0.1.2

คู่มือฉบับขยายนี้ครอบคลุม API ที่ใช้งานได้จริงของ LynPDF RS 0.1.2 ตั้งแต่ quick start ไปจนถึงงานเอกสารภาษาไทย, syntax highlight, fit-to-page, diagnostics, และ certificate template แบบ production

ไฟล์นี้คือ API Guide หลักของโปรเจกต์ (docs/api-guide-v0.1.2.md)

Companion Example Repository

สำหรับผู้ใช้ที่ต้องการ clone โปรเจกต์ตัวอย่างเพื่อทดลอง flow หน้าเว็บไทย -> ใบเสร็จ PDF โดยตรง:

https://github.com/mrchoke/lynpdf-rs-example

```
git clone https://github.com/mrchoke/lynpdf-rs-example
cd lynpdf-rs-example
cargo run
```

1) Installation

LynPDF RS เวอร์ชัน production setup รองรับการใช้งานแบบไม่ต้อง bundle ฟอนต์ในแพ็กเกจ ให้ bootstrap ฟอนต์ก่อนด้วยสคริปต์:

```
./scripts/download-fonts.sh --dest ./fonts
```

หรือ PowerShell:

```
./scripts/download-fonts.ps1 -Dest ./fonts
```

Cargo.toml

```
[dependencies]
lynpdf-rs = "0.1.2"
```

ตรวจสอบเวอร์ชันจาก runtime

```
use lynpdf_rs::LYNPDF_RS_VERSION;

fn main() {
    println!("LynPDF RS version: {}", LYNPDF_RS_VERSION);
}
```

2) Public API Surface (ครบตาม v0.1.2)

Core functions

render_file_to_pdf

render_html_to_pdf

apply_syntax_highlighting_to_html

Certificate functions

build_fullpage_certificate_html

render_fullpage_certificate_pdf

render_fullpage_certificate_pdf_to_file

Core types

RenderOptions

RenderRequest

PdfDocument

PageSize

UserFontMapping

RenderFontStyle

CertificateTemplateOptions

CertificateOrientation

Diagnostic

DiagnosticLevel

3) Quick Start: HTML file -> PDF

```rust
use lynpdf_rs::{render_file_to_pdf, RenderOptions, Result};

fn main() -> Result<()> {
    let options = RenderOptions::default();
    let pdf = render_file_to_pdf(
        "examples/demo-single-page.html",
        Some("examples/styles.css"),
        "examples/output/quick-start.pdf",
        options,
    )?;

    println!("pages={} bytes={}", pdf.pages, pdf.bytes.len());
    Ok(())
}
```

4) String API: HTML + CSS -> PDF

```rust
use lynpdf_rs::{render_html_to_pdf, RenderOptions, RenderRequest, Result};
use std::path::PathBuf;

fn main() -> Result<()> {
    let html = r#"
<!doctype html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="title" content="API Render Example" />
  <meta name="author" content="LynPDF Team" />
</head>
<body>
  <h1>สวัสดี LynPDF</h1>
  <p>RenderRequest lets you pass HTML/CSS directly.</p>
</body>
</html>
"#;

    let css = r#"
@page { size: A4; margin: 24mm; }
body { font-family: Sarabun, sans-serif; color: #13222f; }
h1 { color: #0e7490; }
"#;

    let request = RenderRequest {
        html: html.to_string(),
        css: css.to_string(),
        base_dir: PathBuf::from("."),
        css_base_dir: None,
        options: RenderOptions::default(),
    };

    let pdf = render_html_to_pdf(request)?;
    std::fs::write("examples/output/api-string-render.pdf", pdf.bytes)?;
    Ok(())
}
```

5) Function Catalog และหน้าที่

Function	ใช้เมื่อ	ผลลัพธ์
render_file_to_pdf(input, css, output, options)	มีไฟล์ HTML/Markdown อยู่แล้ว	PdfDocument + เขียนไฟล์ PDF
render_html_to_pdf(request)	มี HTML/CSS เป็น string	PdfDocument ในหน่วยความจำ

Function	ใช้เมื่อ	ผลลัพธ์
apply_syntax_highlighting_to_html(html, options)	ต้อง preprocess code block ก่อน render	HTML ที่ฝัง token coloring
build_fullpage_certificate_html(options)	ต้องการ template HTML certificate	HTML หน้าเดียวเต็มหน้า
render_fullpage_certificate_pdf(options)	ต้องการ PDF certificate เป็น bytes	PdfDocument
render_fullpage_certificate_pdf_to_file(options, path)	ต้องการเขียน certificate ลงไฟล์ทันที	PdfDocument + ไฟล์ PDF

6) RenderOptions API (Builder Methods)

RenderOptions::default() มี methods ต่อไปนี้:

with_user_font_dir(path)
with_user_font_mapping(mapping)
with_kerning(enabled)
with_ligatures(enabled)
with_font_variation("AXIS=VALUE")
with_page_scale_percent(percent)
with_fit_to_page(enabled)
with_syntax_highlighting(enabled)
with_syntax_highlight_theme(theme)
page_scale_factor()

ตัวอย่างรวม:

```
use lynpdf_rs::{RenderFontStyle, RenderOptions, UserFontMapping};

let options = RenderOptions::default()
    .with_user_font_dir("./fonts/custom")
    .with_user_font_mapping(
        UserFontMapping::new("MyThai", "./fonts/custom/MyThai-Regular.ttf")
            .with_weight(400)
            .with_style(RenderFontStyle::Normal),
    )
    .with_user_font_mapping(
        UserFontMapping::new("MyThai", "./fonts/custom/MyThai-Bold.ttf")
            .with_weight(700)
            .with_style(RenderFontStyle::Normal),
    )
    .with_kerning(true)
    .with_ligatures(true)
    .with_font_variation("wght=500")
    .with_page_scale_percent(100.0)
    .with_fit_to_page(false)
    .with_syntax_highlighting(true)
    .with_syntax_highlight_theme("lynpdf-light");
```

สำหรับ service deployment สามารถกำหนดหลาย path ผ่าน env:

```
export LYNPDF_FONT_DIRS="/opt/lynpdf/fonts:/usr/local/share/fonts/lynpdf"
```

7) UserFontMapping API

Functions ที่ใช้บ่อย:

UserFontMapping::new(family, path)
.with_weight(weight)
.with_style(RenderFontStyle::{Normal, Italic})

โครงสร้างนี้สำคัญมากกับเอกสารไทย เพราะช่วย map family เดียวกันตามน้ำหนัก/สไตล์ให้ fallback และ shaping เลือกฟอนต์ได้ถูกตัว

8) Thai Document Recipe (Production)

เคล็ดลับสำหรับเอกสารไทยที่ยาวและซับซ้อน:

ใช้ฟอนต์ไทยหลักผ่าน with_user_font_mapping ให้ครบ Regular/Bold/Italic
เปิด with_kerning(true) และ with_ligatures(true)
สำหรับ variable font ใช้ with_font_variation("wght=...")
ถ้าเนื้อหาเสี่ยงล้นหน้า ใช้ with_fit_to_page(true) หรือ with_page_scale_percent(...)
ใน CSS code block ให้ใช้ white-space: pre-wrap เพื่อรักษา \n และตัดบรรทัดได้

9) Markdown -> PDF + Syntax Highlight

```rust
use lynpdf_rs::{render_file_to_pdf, RenderOptions, Result};

fn main() -> Result<()> {
    let options = RenderOptions::default()
        .with_syntax_highlighting(true)
        .with_syntax_highlight_theme("lynpdf-light");

    let pdf = render_file_to_pdf(
        "tests/fixtures/test-18-markdown.md",
        None::<&str>,
        "tests/output/test-18-markdown-highlighted.pdf",
        options,
    )?;

    println!("markdown pages={}", pdf.pages);
    Ok(())
}
```

Notes:

code fence เช่น rust, typescript, sql, html จะถูกลงสีตาม token
ทั้ง HTML pipeline และ Markdown pipeline ใช้กลไกเดียวกัน (render_html_to_pdf จะเรียก apply_syntax_highlighting_to_html ก่อน parse เสมอ)

ถ้าต้องการปิด ใช้ with_syntax_highlighting(false) หรือ --no-syntax-highlight

HTML -> PDF: ถ้า code block ยังไม่สวย

เพื่อให้ผลลัพธ์นิ่งและอ่านง่ายสำหรับ HTML ตรง แนะนำให้ใช้โครง <pre><code class="language-...">...</code></pre> และใส่ CSS นี้:

```
pre {
  margin: 0 0 12px;
  padding: 10px 12px;
  border: 1px solid #d0d7de;
  background: #f6f8fa;
  white-space: pre-wrap;
  overflow-wrap: anywhere;
  word-break: break-word;
}

pre code {
  display: block;
  font-family: monospace;
  font-size: 12px;
  line-height: 1.5;
}
```

เพิ่มเติมใน v0.1.2: ถ้า <pre>/<code> ไม่มี style ระบบจะเติม default style ที่ปลอดภัยให้อัตโนมัติระหว่าง preprocess syntax highlight แต่ถ้ามี style อยู่แล้ว ระบบจะไม่ทับของเดิม

10) Certificate API: Full-page Single-page

```rust
use lynpdf_rs::{
    render_fullpage_certificate_pdf_to_file,
    CertificateOrientation,
    CertificateTemplateOptions,
    Result,
};

fn main() -> Result<()> {
    let mut opts = CertificateTemplateOptions::default();
    opts.orientation = CertificateOrientation::Landscape;
    opts.recipient_name = "Mr. Choke".to_string();
    opts.certificate_id = "LPR-API-L-0001".to_string();

    let landscape = render_fullpage_certificate_pdf_to_file(
        &opts,
        "examples/output/certificate-landscape-api.pdf",
    )?;

    opts.orientation = CertificateOrientation::Portrait;
    opts.title = "Certificate of Completion".to_string();
    opts.certificate_id = "LPR-API-P-0001".to_string();

    let portrait = render_fullpage_certificate_pdf_to_file(
        &opts,
        "examples/output/certificate-portrait-api.pdf",
    )?;

    println!("landscape={} portrait={}", landscape.pages, portrait.pages);
    Ok(())
}
```

11) Diagnostics และ Metadata

ทุกการ render ส่งกลับ PdfDocument:

- bytes
- pages
- diagnostics

Diagnostic codes ที่เจอบ่อย:

- LYNPDF_VERSION
- LYNPDF_RENDERED
- LYNPDF_FIT_TO_PAGE
- LYNPDF_FIT_TO_PAGE_UNSATISFIED

Metadata default เมื่อเอกสารไม่กำหนดเอง:

Creator: LynPDF 0.1.2

Producer: LynPDF RS 0.1.2

12) CLI Mapping (เทียบกับ API)

```
lynpdf-rs input.html -o output.pdf --verbose \
  --font-dir ./fonts/custom \
  --font-map "MyBrand=./fonts/custom/MyBrand-Regular.ttf" \
  --font-map-bold "MyBrand=./fonts/custom/MyBrand-Bold.ttf" \
  --font-variation "wght=450" \
  --syntax-theme lynpdf-light \
  --fit-to-page
```

Flags ที่รองรับ:

--version

--font-dir

--font-map

--font-map-bold

--font-map-italic

--font-map-bold-italic

--no-kerning

--no-ligatures

--font-variation

--no-syntax-highlight

--syntax-theme

--page-scale

--fit-to-page

13) Example Index

examples/api-certificate-single-page.rs

examples/generate-api-guide-pdf.rs

examples/demo-certificate-single-page-landscape.html

examples/demo-certificate-single-page-portrait.html

tests/fixtures/test-18-markdown.md

tests/fixtures/test-23-thai-mark-positioning.html

tests/fixtures/test-25-thai-language-stress.html

tests/fixtures/test-28-syntax-highlight.html

เอกสารนี้อ้างอิง LynPDF RS 0.1.2 โดยตรง