ApiDOM OpenAPI 3.1 命名空间

用于处理 OpenAPI 3.1 规范的高性能 Rust 库。该库是 ApiDOM 生态系统的一部分，专门为 OpenAPI 3.1 提供类型安全的抽象语法树(AST)操作。

功能特性

🔧 字段注册系统

- **动态字段处理**: 通过 `FieldHandlerMap` 注册自定义字段处理器
- **模式匹配**: 支持正则表达式模式字段（如 x-* 扩展）
- **宏支持**: 提供便捷的 `register_fixed_fields!` 和 `register_pattern_fields!` 宏

🛠️ 字段提取器

- **类型安全提取**: 从 Element 中安全提取各种类型的字段值
- **批量操作**: 支持批量字段提取和验证
- **扩展字段**: 专门处理 OpenAPI 扩展字段（x-* 前缀）
- **默认值支持**: 提供字段缺失时的默认值机制

📄 Schema 加载器

- **多格式支持**: 支持 JSON 文件加载和解析
- **CST 转换**: 高效的 CST（具体语法树）到 AST 转换
- **错误处理**: 详细的解析错误信息和恢复机制

🏗️ 构建器分发

- **类型安全**: 为 OpenAPI 元素提供类型安全的构建器
- **自动分发**: 智能字段分发到对应的元素处理器
- **扩展支持**: 支持规范扩展字段的处理

📋 OpenAPI 3.1 元素

- **InfoElement**: 处理 API 信息元数据
- **ServerElement**: 服务器配置和描述
- **PathItemElement**: 路径项的定义和操作

快速开始

基本使用

```
use apidom_ns_openapi_3_1::{
    elements::*,
    field_registry::*,
    schema_loader::*,
    builder_dispatch::*,
```

```
    field_extractor::*,
};
use apidom_ast::{StringElement, Element};

// 创建 Info 元素
let mut info = InfoElement::new();
info.set_title(StringElement::new("My API"));
info.set_version(StringElement::new("1.0.0"));

// 使用字段处理器
let mut handlers = FieldHandlerMap::<InfoElement>::new();
handlers.register_fixed("title", |value, target, _| {
    if let Element::String(s) = value {
        target.set_title(s.clone());
        Some(())
    } else {
        None
    }
});

// 使用字段提取器
let title = FieldExtractor::extract_string(&element, "title");
let version = FieldExtractor::extract_version(&element, "version");
let extensions = FieldExtractor::extract_extension_fields(&element);

// 验证必填字段
let validation = FieldExtractor::validate_required_fields(&element, &
["title", "version"]);

// 解析 JSON Schema
let loader = SchemaLoader::new();
let json = r#"{"type": "object", "properties": {"name": {"type":
"string"}}}"#;
let element = loader.parse_json_to_element(json)?;
```

运行示例

```
cargo run --example openapi_3_1_demo
```

架构设计

模块结构

```
apidom-ns-openapi-3-1/
├── src/
│   ├── lib.rs              # 库入口
│   ├── elements.rs         # OpenAPI 3.1 元素定义
│   ├── field_registry.rs   # 字段注册和处理系统
```

```
│   ├── schema_loader.rs    # Schema 文件加载器
│   ├── builder_dispatch.rs # 构建器分发器
│   └── field_extractor.rs  # 字段提取工具
├── examples/
│   └── openapi_3_1_demo.rs # 使用示例
└── tests/                  # 测试文件
```

设计理念

1. **类型安全**: 利用 Rust 的类型系统确保 OpenAPI 规范的正确性
2. **高性能**: 零成本抽象和高效的内存管理
3. **可扩展**: 支持自定义字段处理器和扩展
4. **标准兼容**: 严格遵循 OpenAPI 3.1 规范

API 参考

FieldHandlerMap

字段处理器映射表，用于注册和分发字段处理逻辑。

```
impl<T> FieldHandlerMap<T> {
    pub fn new() -> Self
    pub fn register_fixed(&mut self, field_name: impl Into<String>,
handler: FieldHandler<T>)
    pub fn register_pattern(&mut self, pattern: &str, handler:
FieldHandler<T>) -> Result<(), regex::Error>
    pub fn set_default(&mut self, handler: FieldHandler<T>)
    pub fn dispatch(&self, field_name: &str, value: &Element, target:
&mut T, folder: Option<&mut dyn Fold>) -> bool
}
```

FieldExtractor

字段提取器，用于从 Element 中安全地提取各种类型的字段值。

```
impl FieldExtractor {
    pub fn extract_string(element: &Element, field_name: &str) ->
Option<String>
    pub fn extract_number(element: &Element, field_name: &str) ->
Option<f64>
    pub fn extract_integer(element: &Element, field_name: &str) ->
Option<i64>
    pub fn extract_boolean(element: &Element, field_name: &str) ->
Option<bool>
    pub fn extract_string_array(element: &Element, field_name: &str) ->
Option<Vec<String>>
    pub fn extract_extension_fields(element: &Element) ->
```

```
HashMap<String, Element>
    pub fn validate_required_fields(element: &Element, required_fields:
&[&str]) -> Result<(), Vec<String>>
    pub fn extract_with_default<T, F>(element: &Element, field_name:
&str, extractor: F, default: T) -> T
}
```

SchemaLoader

Schema 文件加载和解析器。

```
impl SchemaLoader {
    pub fn new() -> Self
    pub fn load_from_file<P: AsRef<Path>>(&mut self, path: P) ->
Result<Element, SchemaLoadError>
    pub fn parse_json_to_element(&self, json_str: &str) ->
Result<Element, SchemaLoadError>
    pub fn get_definition(&self, schema: &Element, def_path: &str) ->
Option<Element>
}
```

BuilderDispatch

OpenAPI 元素构建器分发器。

```
impl BuilderDispatch {
    pub fn new() -> Self
    pub fn build_info(&self, source: &Element) -> Option<InfoElement>
    pub fn build_server(&self, source: &Element) ->
Option<ServerElement>
    pub fn build_path_item(&self, source: &Element) ->
Option<PathItemElement>
}
```

测试

项目包含全面的测试覆盖：

```
# 运行所有测试
cargo test

# 运行特定模块测试
cargo test field_registry
cargo test schema_loader
cargo test builder_dispatch
cargo test field_extractor
```

集成

该库设计为与其他 ApiDOM 组件无缝集成：

- **apidom-ast**: 提供基础 AST 类型
- **apidom-cst**: 提供 CST 解析功能
- **apidom-visit**: 提供访问者模式支持

版本兼容性

- **Rust**: 1.70+
- **Edition**: 2024
- **OpenAPI**: 3.1.x

许可证

本项目使用 MIT 许可证 - 详见 LICENSE 文件。

贡献

欢迎贡献！请参阅 CONTRIBUTING 了解详细信息。

相关项目

- ApiDOM - 原始 JavaScript/TypeScript 实现
- OpenAPI 规范 - OpenAPI 3.1 官方规范