bk-firecrawl 仕様書

WebサイトをAI対応データに変換する
APIプラットフォーム

github.com/BKStock/bk-firecrawl

社内技術資料 | 2026年2月

Firecrawlとは何か

Webデータ収集の課題を解決するAPIサービス

従来の課題

動的サイトの壁

JavaScriptレンダリングが必要なSPAや動的コンテンツの取得が困難。

複雑な対策

プロキシ管理、IPローテーション、アンチボット対策の実装コストが高い。

非構造化データ

HTMLから必要なデータを抽出し、構造化する処理が煩雑。

Firecrawlの解決策

LLM対応フォーマット

Markdownや構造化JSONに直接変換し、AIエージェントが即座に利用可能。

メディア解析

PDF、DOCX、画像からもテキストを自動抽出し、情報の取りこぼしを防ぐ。

>80%

カバレッジ (信頼性)

業界トップクラスのベンチマーク評価

5つのコア機能

Firecrawlの機能全体像

Scrape機能の詳細

柔軟なフォーマットとアクション対応

対応出力フォーマット

`markdown`	クリーンなMarkdownテキスト
`json`	スキーマベースの構造化データ
`html`	整形済みHTML
`screenshot`	Base64エンコード画像
`links`	ページ内リンク一覧
`branding`	色・フォント・タイポグラフィ

Actions機能 (スクレイプ前の操作)

ログインが必要なページや、スクロールしないと表示されないコンテンツなど、 **ユーザー操作が必要な場面**で一連のアクションを実行できます。

write	click	wait	press	scroll	screenshot
テキスト入力	要素クリック	待機	キー押下	スクロール	撮影

Actions Example

```
"actions": [
  { "type": "write", "text": "user@example.com" },
  { "type": "click", "selector": "#login-btn" },
  { "type": "wait", "milliseconds": 2000 }
]
```

Agent機能

AIによる自律的なWebデータ収集

コンセプト

URLを事前に知らなくても、プロンプトで欲しいデータを記述するだけでAIが自動的に収集します。

- /extract エンドポイントの進化版
- Pydanticスキーマによる構造化出力
- 特定URLへのフォーカスも可能

spark-1-mini

Default

コスト効率を重視した軽量モデル。一般的なデータ収集タスクに最適で、Proモデルより60%安価に利用可能です。

推奨ユースケース

- 単純な情報収集
- コスト重視の大量処理
- 一般的なWebスクレイピング

spark-1-pro

High Precision

複雑な推論と探索能力を持つ高性能モデル。複数のパスを探索し、高精度な抽出を行います。

推奨ユースケース

- 複数サイト間のデータ比較
- 認証が必要なサイトからの抽出
- 複雑なナビゲーションが必要なリサーチ

大規模データ収集機能

Crawl・Map・Batch Scraping

Crawl

`POST /v2/crawl`

Webサイト全体を一括クロールし、全ページのコンテンツを取得

- 非同期ジョブ実行 ジョブIDで進捗管理が可能
- SDK自動ポーリング 完了まで自動待機するオプション
- ページ数制御 `limit` 取得する最大ページ数を指定可能

Map

`POST /v2/map`

Webサイト上の全URLを即座に発見し、一覧化

- 高速なサイト構造把握 クロール前の全体像確認に最適
- メタデータ取得 URLごとのタイトル・説明文も取得
- 検索フィルタリング `search` 特定キーワードを含むURLのみ抽出

Batch

`Batch Scraping`

複数のURLを一度に非同期でスクレイピング処理

- 大規模並列処理 数千URLの規模にも対応可能
- 非同期実行 サーバー負荷を分散しつつ効率的に収集
- エラーハンドリング 個別の失敗が全体に影響しない設計

SDK・インテグレーション

多言語対応と豊富な連携先

公式SDK

Python SDK

```
$ pip install firecrawl-py
```

Node.js SDK

```
$ npm install @mendable/firecrawl-js
```

Java SDK NEW

```
$ apps/java-sdk (Source)
```

コミュニティSDK

Go SDK

Rust SDK

インテグレーション

AIエージェント連携

Firecrawl Skill, MCP対応

Claude Code

Codex

OpenCode

プラットフォーム連携

ノーコード/ローコードツール統合

Zapier

n8n

Lovable

システムアーキテクチャ

Docker Composeによるマイクロサービス構成

セルフホスティングと環境設定

Docker Composeによる迅速なデプロイ

bash — 80x24

```
# 1. リポジトリのクローン
$ git clone https://github.com/BKStock/bk-firecrawl
$ cd bk-firecrawl

# 2. 環境変数の設定
$ cp .env.example .env
$ vim .env

# 3. サービスの起動（バックグラウンド）
$ docker compose up -d

# 起動確認
$ docker compose ps
```

主要な環境変数 (.env)

Variable	Description
`OPENAI_API_KEY`	Agent機能 (/v2/agent) 利用時に必須
`USE_DB_AUTHENTICATION`	APIキー認証の有効化 (true/false)
`NUM_WORKERS_PER_QUEUE`	同時実行ワーカー数 (Default: 4)
`PORT`	APIサーバーのポート (Default: 3002)

アクセスポイント

API ENDPOINT
`http://localhost:3002`

HEALTH CHECK
`/health`

セキュリティと運用上の注意事項

安全なデータ収集と安定運用のためのベストプラクティス

セキュリティ対策

コンプライアンス準拠 SOC2 Type II

エンタープライズグレードのセキュリティ基準に準拠し、データの機密性と可用性を担保。

認証・アクセス制御

APIリクエストはBearer Tokenで保護。データベースへの外部アクセスはDockerネットワークレベルで遮断。

管理UIの保護

BullMQ Dashboardなどの管理画面は、Basic認証またはリバースプロキシによるアクセス制限が必須。

運用・監視

リソース制御

同時実行数（Concurrency）の制限とレートリミットを設定し、サーバー負荷と対象サイトへの影響を最小化。

可観測性 (Observability)

コンテナログの集約と、ジョブキューの状態監視（Waiting, Active, Failed）による早期異常検知。

推奨ネットワーク構成

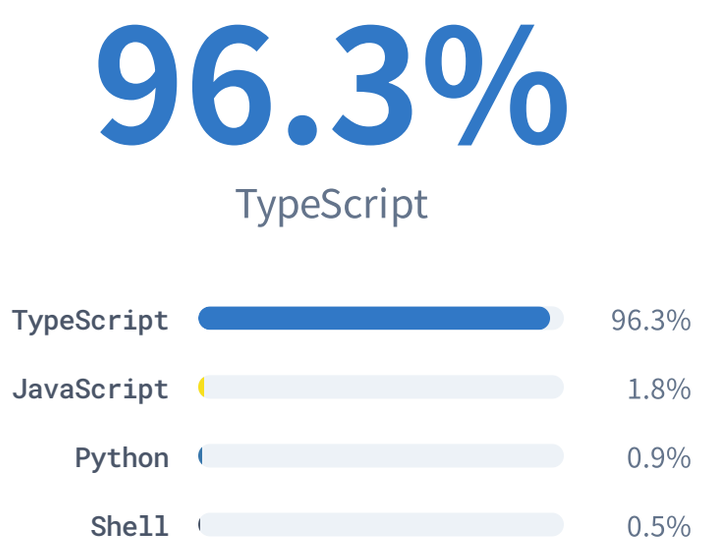
本番環境ではNginxやCaddy等のリバースプロキシを配置し、SSL終端とヘッダーセキュリティを適用。

セルフホスティング時は、.envファイルの `USE_DB_AUTHENTICATION=true` 設定を推奨

技術スタックと言語構成

TypeScriptを中心としたモダンな構成

言語比率



appsディレクトリ構成

api
メインAPIサーバー・バックエンドロジック
Node.js | Express | Redis

playwright-service
ヘッドレスブラウザ管理・スクレイピング実行
Playwright | Chromium

python-sdk
Pythonクライアントライブラリ
Poetry | PyPI

js-sdk
Node.js/TypeScriptクライアントライブラリ
npm | TypeScript

まとめ

次世代のWebデータ収集基盤として