**# 法令知識ベース連携モジュール**

法令知識ベース連携モジュールは、e-Gov 法令API を利用して日本法の条文を取得・パース・インデックス化し、対話システム（Gemini）や上流モジュールが利用できるように「条文検索 API」「条文要約 API」「論点抽出 API」を提供します。

**## 機能概要**

- **\*\*法令取得\*\***: e-Gov API から法令データを取得・パース

- **\*\*条文検索\*\***: 法令や条文を検索・取得

- **\*\*条文要約\*\***: AI（Gemini）を使用した条文要約

- **\*\*論点抽出\*\***: 複数条文から法的論点を抽出

- **\*\*キャッシュ\*\***: Redis を使用したパフォーマンス最適化

- **\*\*同期バッチ\*\***: 定期的な法令データ更新

**## システム要件**

- Python 3.10+

- PostgreSQL 14+

- Redis 6+

- Google Gemini API Key

**## インストール**

**### 1. 依存パッケージのインストール**

```bash

pip install -r requirements.txt

```

**### 2. 環境変数の設定**

`.env` ファイルを作成し、以下の環境変数を設定してください：

```bash

cp env.example .env

```

`.env` ファイルを編集：

```env

# Google Gemini API

GEMINI\_API\_KEY=your\_gemini\_api\_key\_here

GEMINI\_MODEL=gemini-1.5-flash

# e-Gov API

E\_GOV\_API\_KEY=your\_egov\_api\_key\_here

E\_GOV\_BASE\_URL=https://elaws.e-gov.go.jp/api/1

# PostgreSQL Database

DATABASE\_URL=postgresql://user:password@localhost:5432/law\_chat\_db

# Redis Cache

REDIS\_URL=redis://localhost:6379/0

CACHE\_TTL=86400

# ログ設定

LOG\_LEVEL=INFO

```

**### 3. データベースの初期化**

```bash

# Alembic でマイグレーション（将来実装）

alembic upgrade head

```

**## 使用方法**

**### 開発サーバーの起動**

```bash

python -m app.main

```

または

```bash

uvicorn app.main:app --reload --port 8083

```

**### API エンドポイント**

**#### 1. 法令リスト取得**

```bash

GET /laws/list?page=1&per\_page=20&law\_type=法律

```

レスポンス:

```json

{

"laws": [

{

"law\_id": "CIVIL\_LAW\_001",

"title": "民法",

"law\_no": "明治29年法律第89号",

"law\_type": "法律",

"enact\_date": "1896-04-27T00:00:00"

}

],

"total": 1,

"page": 1,

"per\_page": 20

}

```

**#### 2. 法令詳細取得**

```bash

GET /laws/{law\_id}

```

レスポンス:

```json

{

"law\_id": "CIVIL\_LAW\_001",

"title": "民法",

"law\_no": "明治29年法律第89号",

"enact\_date": "1896-04-27T00:00:00",

"articles": [

{

"article\_no": "第1条",

"heading": "私権の内容",

"text": "私権は、公共の福祉に適合しなければならない...",

"structure": {...}

}

],

"metadata": {...}

}

```

**#### 3. 条文取得**

```bash

GET /laws/{law\_id}/articles/{article\_no}

```

レスポンス:

```json

{

"article\_no": "第1条",

"heading": "私権の内容",

"text": "私権は、公共の福祉に適合しなければならない。",

"structure": {

"items": [

{"level": 1, "text": "私権は、公共の福祉に適合しなければならない。"}

]

}

}

```

**#### 4. 条文要約**

```bash

POST /laws/{law\_id}/articles/{article\_no}/summarize

Content-Type: application/json

{

"max\_length": 200,

"style": "plain"

}

```

レスポンス:

```json

{

"summary\_text": "私権は公共の福祉に適合する必要があり...",

"original\_reference": {

"law\_id": "CIVIL\_LAW\_001",

"article\_no": "第1条"

},

"citations": [],

"style": "plain",

"word\_count": 150

}

```

**#### 5. 論点抽出**

```bash

POST /laws/extract\_topics

Content-Type: application/json

{

"texts": ["条文1", "条文2", "条文3"],

"mode": "topic\_extraction",

"max\_topics": 5

}

```

レスポンス:

```json

{

"topics": [

{

"id": "1",

"title": "私権と公共の福祉",

"description": "私権の行使は公共の福祉に適合する必要がある...",

"source\_refs": ["民法第1条", "民法第2条"]

}

],

"relations": [

{

"topic\_id\_1": "1",

"topic\_id\_2": "2",

"relation\_type": "補足"

}

]

}

```

**#### 6. 法令検索**

```bash

POST /laws/search?keyword=民法&page=1&per\_page=20

```

**## ETL バッチ（同期処理）**

**### フル同期（初回インポート）**

```bash

python -m app.scripts.sync\_egov --mode full

```

**### 差分更新**

```bash

python -m app.scripts.sync\_egov --mode update

```

**### Cron 設定（毎日午前2時に実行）**

```bash

# crontab -e

0 2 \* \* \* cd /path/to/WP2-3 && python -m app.scripts.sync\_egov --mode update

```

**## テスト**

**### 単体テストの実行**

```bash

# 全てのテストを実行

pytest

# 特定のテストファイルを実行

pytest tests/test\_parser.py

# カバレッジレポート付きで実行

pytest --cov=app tests/

```

**### テスト結果例**

```

tests/test\_parser.py::test\_parse\_xml PASSED

tests/test\_parser.py::test\_extract\_articles PASSED

tests/test\_api.py::test\_root\_endpoint PASSED

```

**## プロンプトテンプレート**

**### 要約プロンプト（summarize\_ja.txt）**

法令条文の要約プロンプトテンプレート

**### 論点抽出プロンプト（extract\_topics\_ja.txt）**

法令条文からの論点抽出プロンプトテンプレート

**## アーキテクチャ**

```

app/

├── main.py # FastAPI アプリケーション

├── config.py # 設定管理

├── logger.py # ログ設定

├── schemas.py # Pydantic スキーマ

├── models/ # SQLAlchemy モデル

│ └── models.py

├── services/ # ビジネスロジック

│ ├── egov\_client.py # e-Gov API クライアント

│ ├── xml\_parser.py # XML パーサー

│ ├── summarizer.py # 要約サービス

│ ├── topic\_extractor.py # 論点抽出サービス

│ └── cache\_service.py # キャッシュサービス

├── clients/ # 外部APIクライアント

│ └── gemini\_client.py # Gemini API クライアント

├── api/ # API ルーター

│ └── laws.py # 法令API

├── scripts/ # バッチスクリプト

│ └── sync\_egov.py # e-Gov 同期バッチ

└── utils/ # ユーティリティ

└── error\_mapping.py # エラーマッピング

prompt\_templates/ # プロンプトテンプレート

├── summarize\_ja.txt

└── extract\_topics\_ja.txt

tests/ # テスト

├── test\_parser.py

└── test\_api.py

```

**## データベーススキーマ**

**### legal\_refs（法令マスタ）**

- law\_id (PK): 法令ID

- title: 法令名

- law\_no: 法令番号

- law\_type: 法令種別

- enact\_date: 施行年月日

- source\_url: e-Gov API のURL

- raw\_xml: 元のXMLデータ

- created\_at, updated\_at: タイムスタンプ

**### articles（条文）**

- article\_id (PK): 条文ID

- law\_id (FK): 法令ID

- article\_no: 条番号

- heading: 見出し

- text: 条文本文

- parsed\_json: 正規化された構造データ

- created\_at, updated\_at: タイムスタンプ

**### article\_embeddings（条文埋め込み）**

- embedding\_id (PK): 埋め込みID

- article\_id (FK): 条文ID

- embedding: ベクトルデータ

- model\_name: モデル名

- created\_at, updated\_at: タイムスタンプ

**### sync\_log（同期ログ）**

- sync\_id (PK): 同期ID

- sync\_type: 同期種別

- started\_at, finished\_at: 開始・終了時刻

- result\_count: 取得件数

- status: 状態

- error\_message: エラーメッセージ

**## 開発ガイド**

**### API キーの取得**

1. **\*\*Google Gemini API Key\*\***

- [Google AI Studio](https://makersuite.google.com/app/apikey) で取得

2. **\*\*e-Gov API Key\*\***

- [e-Gov API](https://www.e-gov.go.jp/) で登録（利用規約を確認）

**### デバッグ**

ログレベルを `DEBUG` に設定：

```env

LOG\_LEVEL=DEBUG

```

**### モックモード**

Gemini API キーが設定されていない場合、自動的にモックモードで動作します：

- 要約: 条文の先頭部分を返す

- 論点抽出: 簡易的なキーワード抽出を返す

**## トラブルシューティング**

**### データベース接続エラー**

```bash

# PostgreSQL が起動しているか確認

pg\_isready

# 接続テスト

psql $DATABASE\_URL

```

**### Redis 接続エラー**

```bash

# Redis が起動しているか確認

redis-cli ping

# 接続テスト

redis-cli -u $REDIS\_URL ping

```

**### API レート制限エラー**

e-Gov API のレート制限に注意：

- 1分間に60リクエスト

- 日次制限あり（API仕様を確認）

**## ライセンス**

このプロジェクトは内部使用目的です。

**## 連絡先**

問題や質問がある場合は、プロジェクトメンテナーに連絡してください。