JAVA　アプリ　SpringBoot　API　コード解説

API 機能追加

src/main/java/com/example/demo/mapper/NoteMpper.java

```
 1 package com.example.springboot_app.mapper;
 2
 3 import java.util.List;
 4
 5 import org.apache.ibatis.annotations.Mapper;
 6 import org.apache.ibatis.annotations.Param;
 7
 8 import com.example.springboot_app.model.Note;
 9
10 @Mapper
11 public interface NoteMapper {
12
13     int count();
14
15     List<Note> findAll();
16
17     List<Note> findAllApi(@Param("q") String q);
18
19     int insert(String text);
20
21     Note findById(Long id);
22
23     int update(Note note);
24
25     int softDelete(Long id);
26
27     int toggleVisibility(@Param("id") Long id, @Param("isVisible") Integer isVisible);
28 }
```

src/main/resources/mapper/NoteMapper.xml

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <!DOCTYPE mapper
 3   PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
 4   "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
 5
 6 <mapper namespace="com.example.springboot_app.mapper.NoteMapper">
```

```
7
8  <!-- 疎通確認用：件数を取るだけ -->
9  <select id="count" resultType="int">
10     SELECT COUNT(*) FROM notes
11  </select>
12
13  <select id="findAll" resultType="com.example.springboot_app.model.Note">
14     SELECT
15        id,
16        text,
17        created_at,
18        is_deleted,
19        is_visible
20     FROM notes
21     WHERE is_deleted = 0
22       AND is_visible = 1
23     ORDER BY id DESC
24  </select>
25
26  <select id="findAllApi" resultType="com.example.springboot_app.model.Note">
27     SELECT
28        id,
29        text,
30        created_at,
31        is_deleted,
32        is_visible
33     FROM notes
34     WHERE is_deleted = 0
35       AND is_visible = 1
36     <if test="q != null and q != ''">
37       AND text LIKE CONCAT('%', #{q}, '%')
38     </if>
39     ORDER BY id DESC
40  </select>
41
42  <insert id="insert" parameterType="string">
43    INSERT INTO notes (text)
44    VALUES (#{text})
45  </insert>
46
47  <select id="findById" parameterType="long" resultType="com.example.springboot_app.model.Note">
48     SELECT
49        id,
50        text,
51        created_at,
52        is_deleted,
53        is_visible
```

```
    FROM notes
    WHERE id = #{id}
  </select>

  <update id="update" parameterType="com.example.springboot_app.model.Note">
    UPDATE notes
    SET text = #{text}
    WHERE id = #{id}
  </update>

  <update id="softDelete" parameterType="long">
    UPDATE notes
    SET is_deleted = 1
    WHERE id = #{id}
  </update>

  <update id="toggleVisibility">
    UPDATE notes
    SET is_visible = #{isVisible}
    WHERE id = #{id}
  </update>
</mapper>
```

src/main/java/com/example/demo/api/NoteApiController.java NEW

```
package com.example.springboot_app.api;

import java.util.List;

import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.example.springboot_app.mapper.NoteMapper;
import com.example.springboot_app.model.Note;

@RestController
@RequestMapping(value = "/api/notes", produces = MediaType.APPLICATION_JSON_VALUE)
public class NoteApiController {

  private final NoteMapper noteMapper;
```

```
    public NoteApiController(NoteMapper noteMapper) {
        this.noteMapper = noteMapper;
    }

    @GetMapping
    public ItemsResponse list(@RequestParam(name = "q", required = false) String q) {
        List<Note> items = noteMapper.findAllApi(q);
        return new ItemsResponse(items);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Note> findById(@PathVariable("id") Long id) {
        Note note = noteMapper.findById(id);
        if (note == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(note);
    }

    public static class ItemsResponse {

        private final List<Note> items;

        public ItemsResponse(List<Note> items) {
            this.items = items;
        }

        public List<Note> getItems() {
            return items;
        }
    }
}
```

src/main/java/com/example/demo/api/NoteListResponse.java　NEW

```
package com.example.springboot_app.api;

import java.util.List;

import com.example.springboot_app.model.Note;

public class NoteListResponse {

    public List<Note> items;

    public NoteListResponse(List<Note> items) {
        this.items = items;
```

```
  }
}
```

src/main/java/c om/example/demo/config/　NEW

```
package com.example.springboot_app.config;

import org.springframework.lang.NonNull;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class WebConfig {

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(@NonNull CorsRegistry registry) {
                registry.addMapping("/api/**")
                        .allowedOrigins("*")
                        .allowedMethods("GET")
                        .allowedHeaders("*");
            }
        };
    }
}
```

frontend/index.html

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>API 連携 TOP</title>
  <style>
    body {
      font-family: "Hiragino Kaku Gothic ProN", "Yu Gothic", "Meiryo", sans-serif;
      margin: 24px;
    }
    button {
      padding: 8px 16px;
      font-size: 16px;
    }
```

```
      #error {
        color: #b00020;
        margin-top: 12px;
      }
      ul {
        margin-top: 12px;
      }
    </style>
  </head>
  <body>
    <h1>API 連携 TOP</h1>
    <button id="loadButton"> データ表示 </button>
    <div id="error" role="alert"></div>
    <ul id="items"></ul>

    <script>
      const loadButton = document.getElementById('loadButton');
      const itemsList = document.getElementById('items');
      const errorBox = document.getElementById('error');

      function clearDisplay() {
        itemsList.innerHTML = '';
        errorBox.textContent = '';
      }

      loadButton.addEventListener('click', async () => {
        clearDisplay();

        try {
          const response = await fetch('http://localhost:8080/api/notes');
          if (!response.ok) {
            throw new Error(`HTTP ${response.status}`);
          }

          const data = await response.json();
          const items = Array.isArray(data.items) ? data.items : [];

          if (items.length === 0) {
            const emptyItem = document.createElement('li');
            emptyItem.textContent = ' データがありません。';
            itemsList.appendChild(emptyItem);
            return;
          }

          items.forEach((item) => {
            const li = document.createElement('li');
            li.textContent = item.text;
```

```
            itemsList.appendChild(li);
          });
        } catch (error) {
          errorBox.textContent = ` エラーが発生しました : ${error.message}`;
        }
      });
    </script>
  </body>
  </html>
```

プロンプト生成までの流れ

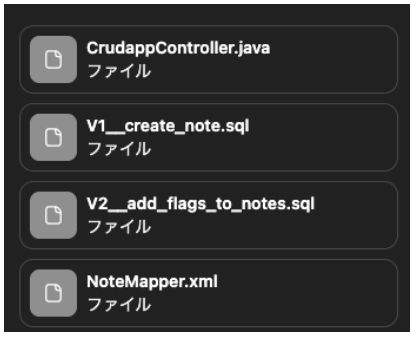
Pronpt 0

既存システムに WebAPI を追加 API は WebAPI を基本として再構成
json データは springboot で作成したアプリに機能を追加する形で進める

[
Chat GPT.....Answer
]

Pronpt1

作成済みファイルをアップロード



[
Chat GPT.....Answer
]

Pronpt2

ソースコードを解析
[
Chat GPT.....Answer
]

Pronpt3

全体を整理・再確認

[
Chat GPT.....Answer
]

Pronpt4

開発の手順を整理

[
Chat GPT.....Answer
]

Pronpt5

完成されたデータはどのように確認できるか
アウトプット形式を再確認

Pronpt6

ブラウザで確認
http://localhost:8080/api/notes http://localhost:8080/api/notes?q=API
ここで JSON が返れば API は連携成功。 この形式を採用

Pronpt7

全体を整理・再確認

[
Chat GPT.....Answer
]

Pronpt2

サーバーサイドとフロントサイドの設計書を作成

[

Chat GPT.....Answer
]

Pronpt3

生成ソースコードを確認

[
Chat GPT.....Answer
]

Pronpt4

MyBatis + DB 版して再考案

[
Chat GPT.....Answer
]

Pronpt5

GET だけにして再考案

[
Chat GPT.....Answer
]

Pronpt5

vscode の codex に流すためのプロンプトを生成

プロンプト完成例

Prompt 0：前提確認（何を変更して良いか）

あなたは既存の Spring Boot + MyBatis プロジェクトを改修します。
以下の前提で作業してください。

- 既存モデル：com.example.springboot_app.model.Note（id/text/createdAt/isDeleted/isVisible）
- 既 存 Mapper：com.example.springboot_app.mapper.NoteMapper（count/findAll/insert/findById/update/softDelete/toggleVisibility）

- 追加するのは GET 専用 API（/api/notes と /api/notes/{id}）のみ
- Swagger/OpenAPI は導入しない
- 画面（Thymeleaf 等）は触らない
- 静的 HTML から叩くために CORS は /api/** の GET のみ許可で追加する
これから個別に指示を出すので、以後はこの前提を守って実装してください。
返答は「理解した」ではなく、次の作業のための準備（影響範囲メモ）を短く出してください。

Prompt 1：NoteMapper.java に API 用メソッドを追加

NoteMapper.java に API 用の GET 一覧メソッドを 1 つ追加してください。

追加するシグネチャ：
List<Note> findAllApi(@Param("q") String q);

- import が必要なら追加する
- 既存メソッドは変更しない
- 出力は NoteMapper.java の完成形（全文）を提示してください

Prompt 2：NoteMapper.xml に findAllApi の SQL を追加

NoteMapper.xml に、NoteMapper.findAllApi(String q) に対応する <select> を追加してください。

要件：
- id="findAllApi"
- resultType="com.example.springboot_app.model.Note"
- SELECT: id, text, created_at, is_deleted, is_visible
- 条件 : is_deleted = 0 AND is_visible = 1
- q が指定されたら text LIKE '%q%'（CONCAT 使用）
- ORDER BY id DESC

既存の SQL や namespace は壊さない。
出力は NoteMapper.xml の完成形（全文）を提示してください。

Prompt 3：GET 専用 API コントローラを追加

GET 専用の API コントローラを新規作成してください。

ファイル：
src/main/java/com/example/springboot_app/api/NoteApiController.java

仕様：
- @RestController
- @RequestMapping("/api/notes")
- GET /api/notes
  - @RequestParam(name="q", required=false) String q

- noteMapper.findAllApi(q) を呼ぶ
- レスポンスは { "items": [...] } の形式（後述のラッパークラスを使用）
- GET /api/notes/{id}
- noteMapper.findById(id)
- null なら 404
- あれば 200 で Note を返す
- produces は application/json（XML は不要。Swagger 無し前提）

出力は NoteApiController.java の全文。

Prompt 4：一覧レスポンス用ラッパークラスを追加

一覧レスポンスを { "items": [...] } に固定するため、ラッパークラスを新規作成してください。

ファイル：
src/main/java/com/example/springboot_app/api/NoteListResponse.java

仕様：
- public class NoteListResponse
- public List<Note> items;
- コンストラクタで items を受け取る
- パッケージは com.example.springboot_app.api
- import を正しく

出力は NoteListResponse.java の全文。

Prompt 5：CORS 設定（/api/** の GET のみ許可）

静的 HTML（別オリジン）から API を呼べるように CORS 設定を追加してください。

ファイル：
src/main/java/com/example/springboot_app/config/WebConfig.java

仕様：
- @Configuration
- WebMvcConfigurer を @Bean で返す
- addCorsMappings で "/api/**" に設定
- allowedOrigins("*")（開発用）
- allowedMethods("GET")
- allowedHeaders("*")

出力は WebConfig.java の全文。

Prompt 6：静的フロント（トップ→ボタン→一覧表示）の HTML を生成

静的 HTML（サーバ生成ではない）で、API 連携の確認用画面を 1 枚作成してください。

ファイル：frontend/index.html（場所は任意、静的であること）

画面要件：
- タイトル「API 連携 TOP」
- ボタン「データ表示」
- ボタンを押すと fetch で GET http://localhost:8080/api/notes を呼ぶ
- 返ってきた JSON の items を <ul> に一覧表示する
- エラー時は画面にエラーメッセージを表示する
- 余計なフレームワークは使わない（素の HTML+JS）

出力は index.html の全文。

Prompt 7：動作確認手順（Swagger なし・ブラウザ直打ち）

Swagger なし前提で、動作確認手順を短くまとめてください。

必須確認 URL：
- http://localhost:8080/api/notes
- http://localhost:8080/api/notes?q=API
- http://localhost:8080/api/notes/1

成功判定：
- 200 で JSON が表示されること

フロント確認：
- 静的 index.html で「データ表示」ボタン押下→一覧表示

手順は箇条書きで。