챗봇 시스템 유지보수 가이드

> 작성일: 2025년 1월 19일(월)
> 버전: 1.0

목차

1. 프로젝트 개요
2. 시스템 아키텍처
3. 파일 구조
4. Java 백엔드 상세 설명
5. JavaScript 프론트엔드 상세 설명
6. API 통신 기초 개념
7. 설정 및 환경변수
8. 트러블슈팅 가이드
9. 자주 묻는 질문 (FAQ)

---

1. 프로젝트 개요

1.1 시스템 소개

이 시스템은 한국도로공사 포털에 통합된 AI 챗봇입니다. 사용자가 질문을 입력하면, 외부 AI 서버에서 답변을 받아와 화면에 표시합니다.

1.2 기술 스택

구분	기술	설명
백엔드	Spring Framework	전자정부 프레임워크 기반
프론트엔드	Vanilla JavaScript	별도 라이브러리 없음
JSON 처리	Gson, Jackson	JSON 파싱 라이브러리
통신 방식	HTTP POST, SSE	REST API 및 스트리밍

1.3 주요 기능

- 사용자 질문 입력 및 AI 답변 표시
- 대화 이력 유지 (멀티턴 대화)
- 마크다운 형식 답변 렌더링
- 추천 질문 표시
- 답변 복사 기능

---

2. 시스템 아키텍처

2.1 전체 데이터 흐름



2.2 요청/응답 흐름 상세

```
[1] 사용자가 질문 입력 후 전송 버튼 클릭
        ↓
[2] chat.js가 대화 히스토리와 함께 JSON 데이터 생성
        ↓
[3] /portal/chatbot/requestChatbot.do 로 POST 요청
        ↓
[4] ChatBotController가 요청 받아서 JSON 파싱
        ↓
[5] ChatPostService.callChatApi() 호출
        ↓
[6] 외부 AI 서버로 HTTP POST 요청 (API Key 포함)
        ↓
[7] AI 서버가 응답 반환 (JSON 또는 SSE 스트림)
        ↓
[8] ChatPostService가 응답 파싱하여 Map으로 반환
        ↓
[9] ChatBotController가 JSON으로 변환하여 응답
        ↓
[10] chat.js가 응답 받아서 화면에 표시
```

---

3. 파일 구조

```
src/
├── main/
│   ├── java/kr/co/ex/portal/chatbot/
│   │   ├── ChatBotController.java       ← API 엔드포인트 (컨트롤러)
│   │   ├── ChatBotPopupController.java  ← 팝업 페이지 컨트롤러
│   │   ├── ChatPostService.java         ← 외부 API 호출 (서비스)
│   │   ├── ChatMessage.java             ← 메시지 데이터 모델
│   │   ├── SearchFilter.java            ← 검색 필터 모델
│   │   └── enums/
│   │       └── Roles.java               ← 부서별 역할 정의
│   └── webapp/
│       ├── js/site/new_portal/
│       │   └── chat.js                  ← 프론트엔드 전체 로직
│       └── images/site/new_portal/chat/
│           └── [44개 이미지 파일]        ← UI 아이콘/이미지
```

---

4. Java 백엔드 상세 설명

4.1 ChatBotController.java

> 역할: 프론트엔드 요청을 받아서 서비스를 호출하고 응답을 반환하는 **진입점**

파일 위치: `main/java/kr/co/ex/portal/chatbot/ChatBotController.java`

클래스 구조

```
@Controller
public class ChatBotController {

    @Autowired
    private ChatPostService chatPostService;  // 서비스 주입

    private final Gson gson = new Gson();     // JSON 변환 도구
}
```

메서드 상세

`requestChatbot()` - 챗봇 API 엔드포인트

항목	내용
URL	`/portal/chatbot/requestChatbot.do`
HTTP 메서드	POST
파라미터	`requestData` (JSON 문자열)
반환	JSON 응답 (PrintWriter로 직접 출력)

```
@RequestMapping("/portal/chatbot/requestChatbot.do")
public void requestChatbot(
        @RequestParam(value = "requestData", required = false) String requestDataJson,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception {
```

처리 흐름:

```
1. requestDataJson 파라미터로 JSON 문자열 수신
   예: {"history":[...],"filter":{"must":["homepage"]},...}

2. Gson으로 JSON을 Map으로 변환
   Map<String, Object> requestData = gson.fromJson(requestDataJson, Map.class);

3. stream 옵션을 false로 설정 (현재 스트리밍 미사용)
   requestData.put("stream", false);

4. ChatPostService 호출하여 외부 API 요청
   Map<String, Object> apiResponse = chatPostService.callChatApi(requestData);

5. 응답 Map 구성
   result.put("success", true);
   result.put("response", apiResponse.get("response"));
   ...

6. JSON으로 변환하여 응답
   response.setContentType("application/json;charset=UTF-8");
   out.print(gson.toJson(result));
```

요청 데이터 예시:

```
{
  "history": [
    {"role": "user", "content": "통행료 문의"},
    {"role": "assistant", "content": "어떤 구간의 통행료를 알고 싶으신가요?"},
    {"role": "user", "content": "서울-부산"}
  ],
  "filter": {"must": ["homepage"]},
  "stream": false,
  "message_id": "550e8400-e29b-41d4-a716-446655440000",
  "session_id": "123e4567-e89b-12d3-a456-426614174000",
  "user_id": "guest",
  "current_time": "2025-01-19T10:30:00.000Z",
  "think_mode": true
}
```

응답 데이터 예시:

```
{
  "success": true,
  "response": "서울-부산 구간 통행료는 약 35,000원입니다.",
  "sources": ["통행료안내_2024.pdf"],
  "suggested_questions": ["결제 방법", "휴게소 정보"],
  "metadata": {...},
  "session_id": "123e4567-e89b-12d3-a456-426614174000"
}
```

4.2 ChatPostService.java

> 역할: 외부 AI 서버와 HTTP 통신을 담당하는 **핵심 서비스**

파일 위치: `main/java/kr/co/ex/portal/chatbot/ChatPostService.java`

설정값 (application.properties에서 주입)

```
@Value("${chatbot.api.url:http://172.16.160.69:38084/v1/client/chat}")
private String apiUrl;  // 외부 AI API 주소

@Value("${chatbot.api.key:sk-prod-xYz123AbCdEf456GhIjK789LmNoPqRsTuVwXyZ012345}")
private String apiKey;  // API 인증 키
```

> 중요: 실제 운영 시 application.properties에서 값을 설정해야 합니다!

메서드 상세

`callChatApi()` - 외부 API 호출 메서드

```
public Map<String, Object> callChatApi(Map<String, Object> requestData) throws Exception
```

항목	내용
매개변수	`requestData` - 요청 데이터 Map
반환값	API 응답 Map
예외	통신 오류 시 Exception 발생

처리 흐름 상세:

```
// [1단계] 요청 데이터를 JSON 문자열로 변환
String jsonBody = objectMapper.writeValueAsString(requestData);

// [2단계] HTTP 연결 설정
URL url = new URL(apiUrl);  // 예: http://172.16.160.69:38084/v1/client/chat
HttpURLConnection conn = (HttpURLConnection) url.openConnection();
conn.setRequestMethod("POST");  // POST 방식

// [3단계] HTTP 헤더 설정
conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");
conn.setRequestProperty("Accept", "application/json, text/event-stream, */*");
conn.setRequestProperty("X-API-Key", apiKey);  // ★ API 인증 키

// [4단계] 타임아웃 설정
conn.setConnectTimeout(30000);  // 연결 타임아웃 30초
conn.setReadTimeout(60000);     // 읽기 타임아웃 60초

// [5단계] 요청 본문 전송
conn.setDoOutput(true);
OutputStream os = conn.getOutputStream();
os.write(jsonBody.getBytes("UTF-8"));

// [6단계] 응답 확인 및 처리
int responseCode = conn.getResponseCode();
String contentType = conn.getContentType();

if (responseCode == 200) {  // HTTP_OK
    if (contentType.contains("text/event-stream")) {
        // SSE 스트림 응답 처리
        return handleEventStream(reader);
    } else {
        // 일반 JSON 응답 처리
        return objectMapper.readValue(response, Map.class);
    }
}
```

`handleEventStream()` - SSE 스트림 처리 메서드

> SSE(Server-Sent Events)란?
> 서버가 데이터를 한 번에 보내지 않고, 조금씩 나눠서 보내는 방식입니다.
> ChatGPT처럼 답변이 글자 단위로 나오는 효과를 구현할 때 사용합니다.

```
private Map<String, Object> handleEventStream(BufferedReader reader) throws Exception
```

SSE 데이터 형식:

```
data:{"response":"안녕","done":false}
data:{"response":"하세요","done":false}
data:{"response":"!","done":true,"sources":[...]}
```

처리 로직:

```
while ((line = reader.readLine()) != null) {
    // "data:" 접두어 제거
    if (line.startsWith("data:")) {
        jsonData = line.substring(5).trim();
    }

    // JSON 파싱
    Map<String, Object> data = objectMapper.readValue(jsonData, Map.class);

    // 응답 텍스트 누적
    if (data.containsKey("response")) {
        fullResponse.append(data.get("response"));
    }

    // 완료 여부 확인
    if (Boolean.TRUE.equals(data.get("done"))) {
        break;
    }
}
```

4.3 ChatBotPopupController.java

> 역할: 챗봇 팝업 페이지를 표시하는 컨트롤러

파일 위치: `main/java/kr/co/ex/portal/chatbot/ChatBotPopupController.java`

메서드 상세

`popup()` - 팝업 페이지 렌더링

항목	내용
URL	`/chatbot/popup.do`
HTTP 메서드	GET
반환	JSP 페이지 경로 (`site/portal/chatbot/chat`)

```
@RequestMapping("/chatbot/popup.do")
public String popup(Model model, HttpServletRequest request) {
    // 카테고리 목록 설정
    List<Map<String, Object>> categoryList = getCategoryList();
    model.addAttribute("categoryList", categoryList);

    return "site/portal/chatbot/chat";  // JSP 경로
}
```

`getCategoryList()` - 카테고리 목록 생성

```
private List<Map<String, Object>> getCategoryList() {
    List<Map<String, Object>> categoryList = new ArrayList<>();

    // TODO: 실제로는 DB에서 조회하도록 수정 필요
    categoryList.add(createCategory("FAQ", "자주 묻는 질문", "faq", 1));
    categoryList.add(createCategory("ROAD", "도로 정보", "road", 2));
    categoryList.add(createCategory("TOLL", "통행료 안내", "toll", 3));
    categoryList.add(createCategory("SERVICE", "휴게소 정보", "service", 4));

    return categoryList;
}
```

4.4 ChatMessage.java

> 역할: 채팅 메시지를 담는 데이터 모델 (POJO)

파일 위치: `main/java/kr/co/ex/portal/chatbot/ChatMessage.java`

```
public class ChatMessage {
    private String role;      // "user" 또는 "assistant"
    private String content;   // 메시지 내용

    // 기본 생성자
    public ChatMessage() {}

    // 매개변수 있는 생성자
    public ChatMessage(String role, String content) {
        this.role = role;
        this.content = content;
    }

    // Getter/Setter
    public String getRole() { return role; }
    public void setRole(String role) { this.role = role; }
    public String getContent() { return content; }
    public void setContent(String content) { this.content = content; }
}
```

사용 예시:

```
// 사용자 메시지
ChatMessage userMsg = new ChatMessage("user", "통행료 알려줘");

// AI 응답 메시지
ChatMessage botMsg = new ChatMessage("assistant", "어떤 구간의 통행료를 알고 싶으신가요?");
```

4.5 SearchFilter.java

> 역할: 검색 필터 조건을 담는 모델

파일 위치: `main/java/kr/co/ex/portal/chatbot/SearchFilter.java`

```
public class SearchFilter {

    // 필터 타입 열거형
    public enum FilterType {
        HOMEPAGE("homepage");  // 홈페이지

        private final String value;

        FilterType(String value) {
            this.value = value;
        }

        public String getValue() {
            return value;
        }
    }

    private List<String> must;  // 필터 조건 배열

    // 생성자, Getter, Setter...
}
```

사용 예시:

```
{
  "filter": {
    "must": ["homepage"]
  }
}
```

4.6 Roles.java (열거형)

> 역할: 부서별 필터 규칙 정의

파일 위치: `main/java/kr/co/ex/portal/chatbot/enums/Roles.java`

```
public enum Roles {
    // 열거형 상수 (필터값, 부서코드, 부서명)
    HIPORTAL("hiportal", "dept_hiportal", "홈페이지"),
    NEBID_INT("nebid_int", "dept_nebid_int", "전자조달 내부"),
    NEBID_EXT("nebid_ext", "dept_nebid_ext", "전자조달 외부"),
    DISASTER("disaster", "dept_disaster", "재난관리처"),
    GENERAL_AFFAIRS("general_affairs", "dept_general_affairs", "총무처"),
    TECHNICAL_REVIEW("technical_review", "dept_technical_review", "기술심사처"),
    EXHOME("exhome", "dept_exhome", "ex-home"),
    DEFAULT("default", "default", "기본");

    private final String filterValue;   // 필터 값
    private final String deptCode;      // 영문 부서 코드
    private final String deptName;      // 한글 부서명

    // 생성자
    Roles(String filterValue, String deptCode, String deptName) {
        this.filterValue = filterValue;
        this.deptCode = deptCode;
        this.deptName = deptName;
    }

    // Getter 메서드들...

    // 필터 값으로 Role 찾기
    public static Roles fromFilterValue(String filterValue) {
        for (Roles role : values()) {
            if (role.getFilterValue().equals(filterValue)) {
                return role;
            }
        }
        return DEFAULT;  // 못 찾으면 기본값 반환
    }

    // 부서 코드로 Role 찾기
    public static Roles fromDeptCode(String deptCode) {
        for (Roles role : values()) {
            if (role.getDeptCode().equals(deptCode)) {
                return role;
            }
        }
        return DEFAULT;
    }
}
```

사용 예시:

```
// 필터 값으로 부서 찾기
Roles role = Roles.fromFilterValue("disaster");
System.out.println(role.getDeptName());  // 출력: 재난관리처

// 부서 코드로 찾기
Roles role2 = Roles.fromDeptCode("dept_nebid_int");
System.out.println(role2.getDeptName());  // 출력: 전자조달 내부
```

---

5. JavaScript 프론트엔드 상세 설명

5.1 chat.js 전체 구조

파일 위치: `main/webapp/js/site/new_portal/chat.js`

```
const JS = (() => {
    // ===== 내부 변수 =====
    let sessionId = generateUUID();        // 세션 ID (새로고침 시 재생성)
    let conversationHistory = [];          // 대화 이력 배열
    let isLoading = false;                 // 로딩 상태 (중복 요청 방지)

    // ===== 유틸리티 함수 =====
    const $ = (el) => document.querySelector(el);      // 단일 요소 선택
    const $$ = (el) => document.querySelectorAll(el);  // 복수 요소 선택

    // ===== 공개 메서드 =====
    return {
        init  // 초기화 함수만 외부 공개
    };
})();
```

5.2 주요 함수 상세 설명

`generateUUID()` - UUID 생성 함수

> UUID란?
> 전 세계적으로 고유한 식별자입니다. 형식: `xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`

```
const generateUUID = () => {
    return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function(c) {
        const r = Math.random() * 16 | 0;
        const v = c === 'x' ? r : (r & 0x3 | 0x8);
        return v.toString(16);
    });
};
```

용도: 세션 ID, 메시지 ID 생성

`escapeHtml()` - HTML 이스케이프 함수

> 보안 중요! XSS 공격 방지를 위해 사용자 입력을 그대로 출력하면 안 됩니다.

```
const escapeHtml = (text) => {
    const map = {
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        '"': '&quot;',
        "'": '&#039;'
    };
    return text.replace(/[&<>"']/g, m => map[m]);
};
```

변환 예시:

입력	출력
`<script>`	`<script>`
`"Hello"`	`"Hello"`

`removeThinkTags()` - AI 사고 과정 제거 함수

AI가 `<think>...</think>` 태그 안에 사고 과정을 넣어서 응답할 때, 이를 제거합니다.

```
const removeThinkTags = (text) => {
    if (!text) return '';
    return text.replace(/<think>[\s\S]*?<\/think>/g, '').trim();
};
```

변환 예시:

```
입력: "<think>사용자가 통행료를 물어봤으니...</think>서울-부산 통행료는 35,000원입니다."
출력: "서울-부산 통행료는 35,000원입니다."
```

`formatMarkdown()` - 마크다운 렌더링 함수

AI 응답에 포함된 마크다운을 HTML로 변환합니다.

```
const formatMarkdown = (text) => {
    if (!text) return '';

    // 1. 특수 공백 정규화
    text = text.replace(/[\u00A0\u2000-\u200B\u202F\u205F\u3000]/g, ' ');

    // 2. 제목 처리 (### → <h4>, ## → <h3>, # → <h2>)
    text = text.replace(/^###\s*(.+)/gm, '<h4>$1</h4>');
    text = text.replace(/^##\s*(.+)/gm, '<h3>$1</h3>');
    text = text.replace(/^#\s+(.+)/gm, '<h2>$1</h2>');

    // 3. 볼드/이탤릭 처리
    text = text.replace(/\*\*(.+?)\*\*/g, '<strong>$1</strong>');
    text = text.replace(/\*([^*]+)\*/g, '<em>$1</em>');

    // 4. 줄바꿈 처리
    text = text.replace(/\n/g, '<br>');

    // ... 추가 처리 ...

    return text;
};
```

변환 예시:

마크다운	HTML
`## 제목`	`<h3>제목</h3>`
`**굵게**`	`<strong>굵게</strong>`
`*기울임*`	`<em>기울임</em>`

`sendMessage()` - 메시지 전송 함수 (핵심!)

```
const sendMessage = (message) => {
    // [1] 빈 메시지 체크
    if (!message || message.trim() === '') {
        alert('메시지를 입력해주세요.');
        return;
    }

    // [2] 중복 요청 방지
    if (isLoading) {
        console.log('[ChatBot] 답변 생성 중입니다.');
        return;
    }

    // [3] 화면에 사용자 메시지 표시
    appendUserMessage(message);

    // [4] 입력창 비활성화
    isLoading = true;
    inputChat.disabled = true;

    // [5] 대화 히스토리에 추가
    conversationHistory.push({
        role: "user",
        content: message
    });

    // [6] API 요청 데이터 구성
    const requestData = {
        history: conversationHistory,     // 전체 대화 이력
        filter: { must: ["homepage"] },   // 필터 조건
        stream: false,                    // 스트리밍 비사용
        message_id: generateUUID(),       // 메시지 고유 ID
        session_id: sessionId,            // 세션 ID
        user_id: userId,                  // 사용자 ID
        current_time: new Date().toISOString(),  // 현재 시간
        think_mode: true                  // AI 사고 모드
    };

    // [7] API 호출
    fetch('/portal/chatbot/requestChatbot.do', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded;charset=UTF-8'
        },
        body: 'requestData=' + encodeURIComponent(JSON.stringify(requestData))
    })
    .then(response => {
        // HTTP 상태 코드별 처리
        if (response.status === 403) {
```

```
            throw new Error('[오류 403] 접근 권한이 없습니다.');
        }
        if (response.status === 404) {
            throw new Error('[오류 404] API를 찾을 수 없습니다.');
        }
        if (response.status >= 500) {
            throw new Error('[오류] 서버 오류가 발생했습니다.');
        }
        return response.json();
    })
    .then(data => {
        // [8] 성공 응답 처리
        if (data.success && data.response) {
            let cleanResponse = removeThinkTags(data.response);
            appendBotMessage(cleanResponse, true);

            // 대화 히스토리에 AI 응답 추가
            conversationHistory.push({
                role: "assistant",
                content: cleanResponse
            });

            // 추천 질문 표시
            if (data.suggested_questions) {
                appendSuggestedQuestions(data.suggested_questions);
            }
        }
    })
    .catch(error => {
        // [9] 오류 처리
        appendBotMessage(error.message, false);
    })
    .finally(() => {
        // [10] 입력창 재활성화
        isLoading = false;
        inputChat.disabled = false;
    });
};
```

`appendUserMessage()` - 사용자 메시지 표시

```
const appendUserMessage = (message) => {
    const chatContainer = $('.ai-chat');

    const messageDiv = document.createElement('div');
    messageDiv.className = 'chat-client';
    messageDiv.innerHTML = `
        <p class="text-wrap">${escapeHtml(message)}</p>
    `;

    chatContainer.appendChild(messageDiv);
    chatContainer.scrollTop = chatContainer.scrollHeight;  // 스크롤 맨 아래로
};
```

`appendBotMessage()` - 챗봇 메시지 표시

```
const appendBotMessage = (message, showCopyButton = false) => {
    const chatContainer = $('.ai-chat');
    const messageId = 'bot-msg-' + Date.now();

    // 오류 메시지 감지
    const isError = message.startsWith('[오류') ||
                    message.startsWith('[네트워크 오류]');

    // 오류 시 빨간색 스타일 적용
    const errorStyle = isError ? `
        background: #fff0f0;
        border: 1px solid #ff6b6b;
        color: #c92a2a;
    ` : '';

    // 복사 버튼 (로딩 메시지가 아닐 때만)
    const copyButtonHtml = showCopyButton ? `
        <button class="btn-copy" data-message-id="${messageId}">복사</button>
    ` : '';

    const messageDiv = document.createElement('div');
    messageDiv.className = 'chat-admin';
    messageDiv.id = messageId;
    messageDiv.innerHTML = `
        <span class="profile">...</span>
        <div class="text-wrap" style="${errorStyle}">${formatMarkdown(message)}</div>
        ${copyButtonHtml}
    `;

    chatContainer.appendChild(messageDiv);

    // 복사 버튼 이벤트 바인딩
    if (showCopyButton) {
        const copyBtn = messageDiv.querySelector('.btn-copy');
        copyBtn.addEventListener('click', () => copyMessageById(messageId));
    }

    return messageId;
};
```

`init()` - 초기화 함수

```
const init = () => {
    // 전송 버튼 클릭 이벤트
    const sendButton = $('.button-send');
    if (sendButton) {
        sendButton.addEventListener('click', () => {
            sendMessage($('#input-chat').value.trim());
        });
    }

    // Enter 키 이벤트
    const inputChat = $('#input-chat');
    if (inputChat) {
        inputChat.addEventListener('keypress', (e) => {
            if (e.key === 'Enter') {
                e.preventDefault();
                sendMessage(inputChat.value.trim());
            }
        });
    }

    // ESC 키로 챗봇 종료
    document.addEventListener('keydown', (e) => {
        if (e.key === 'Escape') {
            const chatLayer = $('.chat-wrap.show');
            if (chatLayer && confirm('챗봇을 종료하시겠습니까?')) {
                chatLayer.querySelector('.button-close').click();
            }
        }
    });
};
```

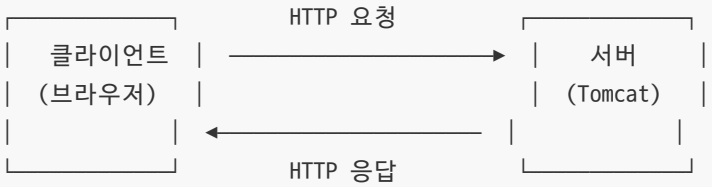
---

6. API 통신 기초 개념

> 전자정부 프레임워크 개발자를 위한 네트워크 통신 기초

6.1 HTTP 요청이란?

HTTP는 웹에서 데이터를 주고받는 규약(프로토콜)입니다.



6.2 HTTP 메서드

메서드	용도	예시
GET	데이터 조회	페이지 로딩, 목록 조회
POST	데이터 전송	폼 제출, API 호출
PUT	데이터 수정	전체 데이터 업데이트
DELETE	데이터 삭제	레코드 삭제

이 프로젝트에서 사용: POST (챗봇 API 호출)

6.3 HTTP 헤더

HTTP 요청/응답에 포함되는 메타 정보입니다.

```
POST /portal/chatbot/requestChatbot.do HTTP/1.1
Host: www.ex.co.kr
Content-Type: application/x-www-form-urlencoded;charset=UTF-8  ← 데이터 형식
X-API-Key: sk-prod-xxxxx                                       ← 인증 키
```

주요 헤더 설명:

헤더	설명	예시
`Content-Type`	요청 본문의 데이터 형식	`application/json`
`Accept`	원하는 응답 형식	`application/json`
`X-API-Key`	API 인증 키	`sk-prod-xxxxx`

6.4 HTTP 상태 코드

코드	의미	설명
200	OK	성공
400	Bad Request	잘못된 요청
403	Forbidden	권한 없음 (로그인 필요)
404	Not Found	URL을 찾을 수 없음
500	Internal Server Error	서버 오류
502	Bad Gateway	게이트웨이 오류
503	Service Unavailable	서비스 불가

6.5 JSON 형식

> JSON: JavaScript Object Notation, 데이터 교환 형식

```
{
  "이름": "값",
  "숫자": 123,
  "배열": [1, 2, 3],
  "객체": {
    "중첩키": "중첩값"
  },
  "불린": true,
  "널": null
}
```

Java에서 JSON 처리:

```
// JSON 문자열 → Java 객체 (파싱)
Gson gson = new Gson();
Map<String, Object> data = gson.fromJson(jsonString, Map.class);

// Java 객체 → JSON 문자열 (직렬화)
String json = gson.toJson(data);
```

6.6 SSE (Server-Sent Events)

서버가 클라이언트에게 실시간으로 데이터를 보내는 기술입니다.

```
일반 HTTP:
클라이언트 —요청—▶ 서버
클라이언트 ◀—응답— 서버 (한 번에 전체 데이터)

SSE:
클라이언트 —요청—▶ 서버
클라이언트 ◀—데이터1— 서버
클라이언트 ◀—데이터2— 서버
클라이언트 ◀—데이터3— 서버
클라이언트 ◀—완료— 서버
```

SSE 데이터 형식:

```
data:{"response":"안","done":false}

data:{"response":"녕","done":false}

data:{"response":"!","done":true}
```

Content-Type: `text/event-stream`

---

7. 설정 및 환경변수

7.1 application.properties 설정

```
# 챗봇 API 설정
chatbot.api.url=http://172.16.160.69:38084/v1/client/chat
chatbot.api.key=sk-prod-실제키값을여기에입력

# 타임아웃 설정 (필요시)
# chatbot.connect.timeout=30000
# chatbot.read.timeout=60000
```

7.2 설정값 변경 방법

1. `application.properties` 파일 수정
2. WAS 재시작
3. 변경사항 확인

주의: API 키는 절대 소스코드에 하드코딩하지 마세요!

7.3 외부 API 서버 정보

항목	값
URL	`http://172.16.160.69:38084/v1/client/chat`
포트	38084
프로토콜	HTTP
인증	X-API-Key 헤더

---

8. 트러블슈팅 가이드

8.1 자주 발생하는 오류

오류 1: "unexpected token '<'"

원인: 서버가 JSON 대신 HTML을 반환함 (로그인 페이지 등)

해결:
1. 로그인 세션 확인
2. API URL 경로 확인
3. 서버 로그 확인

```
[확인 방법]
브라우저 개발자 도구 → Network 탭 → 요청 선택 → Response 탭
→ HTML이 보이면 이 오류
```

오류 2: "Failed to fetch"

원인: 네트워크 연결 실패

해결:
1. 서버 상태 확인
2. 방화벽 설정 확인
3. URL 확인

오류 3: HTTP 403 Forbidden

원인: API 키 인증 실패 또는 권한 없음

해결:
1. `application.properties`의 API 키 확인
2. API 키 유효기간 확인
3. 서버 측 인증 설정 확인

오류 4: HTTP 500 Internal Server Error

원인: 서버 내부 오류

해결:
1. 서버 로그 확인 (`catalina.out` 등)
2. 요청 데이터 형식 확인
3. 외부 AI 서버 상태 확인

오류 5: 타임아웃 (응답 없음)

원인: 외부 API 서버 응답 지연

해결:
1. 외부 AI 서버 상태 확인
2. 네트워크 지연 확인
3. 타임아웃 값 조정

```
// ChatPostService.java에서 타임아웃 수정
conn.setConnectTimeout(60000);  // 연결 타임아웃 60초로 증가
conn.setReadTimeout(120000);    // 읽기 타임아웃 2분으로 증가
```

8.2 로그 확인 방법

Java 백엔드 로그 (콘솔/로그 파일):

```
[requset ChatbotAPI] ...
[requestDataJson] {"history":[...],...}
[Response Code] 200
[Content-Type] application/json
```

JavaScript 프론트엔드 로그 (브라우저 개발자 도구 → Console):

```
[ChatBot API] 요청 시작...
[ChatBot API] 요청 데이터: {...}
[ChatBot API] 응답 상태: 200
[ChatBot API] 응답 데이터: {...}
```

8.3 디버깅 체크리스트

```
□ 1. 브라우저 개발자 도구 열기 (F12)
□ 2. Network 탭에서 요청 확인
   □ 요청 URL이 올바른가?
   □ 요청 메서드가 POST인가?
   □ 요청 데이터가 올바른가?
   □ 응답 상태 코드가 200인가?
   □ 응답 데이터가 JSON인가?
□ 3. Console 탭에서 오류 메시지 확인
□ 4. 서버 로그 확인
□ 5. 외부 AI 서버 상태 확인
```

---

9. 자주 묻는 질문 (FAQ)

Q1. API 키는 어디서 관리하나요?

A: `application.properties` 파일에서 관리합니다.

```
chatbot.api.key=sk-prod-실제키값
```

절대 소스코드에 직접 넣지 마세요!

Q2. 새로운 카테고리를 추가하려면?

A: `ChatBotPopupController.java`의 `getCategoryList()` 메서드를 수정합니다.

```
categoryList.add(createCategory("NEW", "새 카테고리", "new", 5));
```

> 권장: 실제로는 DB에서 조회하도록 수정하세요.

Q3. 대화 이력은 어디에 저장되나요?

A: 브라우저 메모리(JavaScript 변수)에 저장됩니다.

```
let conversationHistory = [];  // chat.js
```

- 새로고침하면 초기화됩니다.
- 서버에는 저장되지 않습니다.
- 영구 저장이 필요하면 DB 연동이 필요합니다.

Q4. 타임아웃 시간을 변경하려면?

A: `ChatPostService.java`에서 수정합니다.

```
conn.setConnectTimeout(30000);  // 연결 타임아웃 (밀리초)
conn.setReadTimeout(60000);     // 읽기 타임아웃 (밀리초)
```

Q5. 부서별로 다른 필터를 적용하려면?

A: `Roles.java` 열거형을 활용합니다.

```
// 부서 코드로 Role 가져오기
Roles role = Roles.fromDeptCode("dept_disaster");

// 필터 값 사용
String filterValue = role.getFilterValue();  // "disaster"
```

요청 데이터의 `filter.must` 배열에 해당 값을 넣으면 됩니다.

Q6. 프론트엔드에서 API 키가 노출되나요?

A: 아니요, 노출되지 않습니다.

```
브라우저 → [API 키 없이 요청] → WAS → [API 키 추가] → 외부 서버
```

API 키는 `ChatPostService.java`에서만 사용되며, 브라우저에는 전달되지 않습니다.

Q7. SSE 스트리밍을 활성화하려면?

A: 현재 `stream: false`로 설정되어 있습니다.

```
// ChatBotController.java
requestData.put("stream", false);  // true로 변경
```

단, 프론트엔드에서 SSE 응답을 처리하는 로직이 추가로 필요합니다.

Q8. 마크다운 스타일을 변경하려면?

A: `chat.js`의 `formatMarkdown()` 함수를 수정합니다.

```
// 예: 제목 색상 변경
text = text.replace(/^##\s*(.+)/gm,
    '<h3 style="color:#FF0000;">$1</h3>');  // 빨간색으로 변경
```

부록: 용어 정리

용어	설명
API	Application Programming Interface, 프로그램 간 통신 규약
JSON	JavaScript Object Notation, 데이터 교환 형식
HTTP	HyperText Transfer Protocol, 웹 통신 규약
SSE	Server-Sent Events, 서버→클라이언트 실시간 통신
UUID	Universally Unique Identifier, 고유 식별자
XSS	Cross-Site Scripting, 웹 보안 취약점
POJO	Plain Old Java Object, 단순 자바 객체
DTO	Data Transfer Object, 데이터 전달 객체

수정 이력

버전	날짜	작성자	내용
1.0	2025-01-19	-	최초 작성
