

# SpringBoot\_연동\_가이드

## Spring Boot (Java 21) 연동 가이드

프로젝트: EduMentor AI

목적: Python AI Engine과 Spring Boot Backend 통합

Java 버전: Java 21

Spring Boot 버전: 3.2.0

Python 서버: 단일 서버 (Port 8000)

### 📌 역할 분담

시스템	역할	주요 책임
Spring Boot	Backend + 인증 + DB 관리	PostgreSQL CRUD, JWT 인증, Python 호출
Python AI	AI Engine	RAG 파이프라인, LangGraph, Upstage API, ChromaDB

주의: Python 서버는 PostgreSQL을 직접 사용하지 않습니다. 모든 DB 작업은 Spring Boot가 담당합니다.

### 📋 시스템 구조



## 1. Spring Boot 프로젝트 생성

### 1.1 프로젝트 구조

```
spring-backend/
├── src/main/java/com/edumentor/
│   ├── EduMentorApplication.java
│   ├── config/
│   │   ├── SecurityConfig.java
│   │   ├── WebClientConfig.java
│   │   └── CorsConfig.java
│   ├── domain/
│   │   └── user/
```

```

├── User.java
├── UserRepository.java
├── UserService.java
├── material/
│   ├── Material.java
│   ├── MaterialRepository.java
│   └── MaterialService.java
├── qa/
│   ├── QASession.java
│   ├── QARepository.java
│   └── QAService.java
├── problem/
│   ├── Problem.java
│   ├── ProblemRepository.java
│   └── ProblemService.java
├── client/
│   ├── PythonClient.java
│   ├── dto/
│   │   ├── QARequest.java
│   │   ├── QAResponse.java
│   │   ├── ProblemRequest.java
│   │   └── ProblemResponse.java
│   ├── controller/
│   │   ├── MaterialController.java
│   │   ├── QAController.java
│   │   └── ProblemController.java
│   └── exception/
│       └── GlobalExceptionHandler.java
├── src/main/resources/
│   ├── application.yml
│   └── application-dev.yml
├── build.gradle
├── settings.gradle
└── gradlew / gradlew.bat

```

## 1.2 build.gradle

```

plugins {
    id 'java'
    id 'org.springframework.boot' version '3.2.0'
    id 'io.spring.dependency-management' version '1.1.4'
}

group = 'com.edumentor'
version = '1.0.0'

java {
    sourceCompatibility = '21'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
}

repositories {
    mavenCentral()
}

dependencies {
    // Spring Boot Web
    implementation 'org.springframework.boot:spring-boot-starter-web'

    // Spring Boot Security
    implementation 'org.springframework.boot:spring-boot-starter-security'

    // Spring Data JPA
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

    // PostgreSQL Driver

```

```
runtimeOnly 'org.postgresql:postgresql'

// WebFlux (WebClient)
implementation 'org.springframework.boot:spring-boot-starter-webflux'

// Validation
implementation 'org.springframework.boot:spring-boot-starter-validation'

// Lombok
compileOnly 'org.projectlombok:lombok'
annotationProcessor 'org.projectlombok:lombok'

// JWT
implementation 'io.jsonwebtoken:jjwt-api:0.12.3'
runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.12.3'
runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.12.3'

// Test
testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

### 1.3 settings.gradle

```
rootProject.name = 'edumentor-backend'
```

## 2. 설정 파일

### 2.1 application.yml

```
spring:
  application:
    name: edumentor-backend

# Database
datasource:
  url: jdbc:postgresql://localhost:5432/edumentor
  username: admin
  password: password
  driver-class-name: org.postgresql.Driver

jpa:
  hibernate:
    ddl-auto: update
  show-sql: true
  properties:
    hibernate:
      format_sql: true
      dialect: org.hibernate.dialect.PostgreSQLDialect

# File Upload (대용량 지원)
servlet:
  multipart:
    max-file-size: 300MB
    max-request-size: 300MB
    enabled: true
    file-size-threshold: 10MB

# Python AI Server (단일 서버)
python:
  ai-service:
    url: http://localhost:8000

# 🌟 File Storage 설정 (공유 볼륨)
```

```

file:
  upload-dir: /app/shared/uploads # Docker 공유 볼륨 경로
  allowed-extensions: pdf
  max-size: 314572800 # 300MB in bytes

# JWT
jwt:
  secret: your-secret-key-change-this-in-production
  expiration: 86400000 # 24 hours

# Server (대용량 파일 처리)
server:
  port: 8080
  tomcat:
    max-swallow-size: 320MB
    connection-timeout: 180000 # 3분

# Logging
logging:
  level:
    com.edumentor: DEBUG
    org.springframework.web: INFO

```

## 2.2 WebClient 설정

```

// config/WebClientConfig.java
package com.edumentor.config;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.reactive.function.client.WebClient;

import java.time.Duration;

@Configuration
public class WebClientConfig {

    @Value("${python.ai-service.url}")
    private String aiServiceUrl;

    @Bean
    public WebClient pythonWebClient() {
        return WebClient.builder()
            .baseUrl(aiServiceUrl) // http://localhost:8000
            .defaultHeader("Content-Type", "application/json")
            .build();
    }
}

```

## 2.3 FileStorage 설정 (신규)

```

// config/FileStorageConfig.java
package com.edumentor.config;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.annotation.Data;

import java.util.List;

@Configuration
@ConfigurationProperties(prefix = "file")
@Data
public class FileStorageConfig {

    /**
     * 파일 업로드 디렉토리 경로
     * Docker: /app/shared/uploads (Python과 공유)
     */
}

```

```

    * Local: ./uploads
    */
    private String uploadDir;

    /**
     * 허용된 파일 확장자 목록
     */
    private List<String> allowedExtensions;

    /**
     * 최대 파일 크기 (bytes)
     */
    private long maxSize;
}

```

## 2.4 FileStorageService (신규)

```

// service/FileStorageService.java
package com.edumentor.service;

import com.edumentor.config.FileStorageConfig;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import java.util.UUID;

@Slf4j
@Service
@RequiredArgsConstructor
public class FileStorageService {

    private final FileStorageConfig fileConfig;

    /**
     * 파일을 공유 디렉토리에 저장
     */
    public String storeFile(MultipartFile file) {
        validateFile(file);

        String originalFilename = StringUtils.cleanPath(file.getOriginalFilename());
        String fileExtension = getFileExtension(originalFilename);
        String storedFilename = UUID.randomUUID().toString() + "." + fileExtension;

        try {
            Path uploadPath = Paths.get(fileConfig.getUploadDir());
            if (!Files.exists(uploadPath)) {
                Files.createDirectories(uploadPath);
                log.info("Created upload directory: {}", uploadPath);
            }

            Path targetLocation = uploadPath.resolve(storedFilename);
            Files.copy(file.getInputStream(), targetLocation, StandardCopyOption.REPLACE_EXISTING);

            String absolutePath = targetLocation.toAbsolutePath().toString();
            log.info("File stored successfully: {}", absolutePath);

            return absolutePath;
        } catch (IOException ex) {
            log.error("Failed to store file: {}", originalFilename, ex);
            throw new RuntimeException("Failed to store file: " + originalFilename, ex);
        }
    }
}

```

```

private void validateFile(MultipartFile file) {
    if (file.isEmpty()) {
        throw new IllegalArgumentException("Cannot store empty file");
    }

    if (file.getSize() > fileConfig.getMaxSize()) {
        throw new IllegalArgumentException(
            String.format("File size exceeds maximum: %d bytes", fileConfig.getMaxSize())
        );
    }

    String filename = StringUtils.cleanPath(file.getOriginalFilename());
    String extension = getFileExtension(filename);

    if (!fileConfig.getAllowedExtensions().contains(extension.toLowerCase())) {
        throw new IllegalArgumentException(
            String.format("File extension not allowed: %s", extension)
        );
    }

    if (filename.contains("..")) {
        throw new IllegalArgumentException("Invalid path sequence in filename");
    }
}

private String getFileExtension(String filename) {
    int lastDotIndex = filename.lastIndexOf('.');
    if (lastDotIndex == -1) {
        throw new IllegalArgumentException("File has no extension");
    }
    return filename.substring(lastDotIndex + 1);
}

public String getFileType(String filename) {
    String extension = getFileExtension(filename).toLowerCase();
    if ("pdf".equals(extension)) {
        return "PDF";
    }
    throw new IllegalArgumentException("Unsupported file type: " + extension);
}
}

```

## 3. 도메인 엔티티

### 3.1 User 엔티티

```

// domain/user/User.java
package com.edumentor.domain.user;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.CreatedDate;

import java.time.LocalDateTime;

@Entity
@Table(name = "users")
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class User {

    @Id

```

```

@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@Column(unique = true, nullable = false, length = 50)
private String username;

@Column(nullable = false)
private String password;

@Column(nullable = false, length = 20)
@Enumerated(EnumType.STRING)
private Role role;

@CreatedDate
@Column(nullable = false, updatable = false)
private LocalDateTime createdAt;

public enum Role {
    INSTRUCTOR, STUDENT
}
}

```

## 3.2 Material 엔티티

```

// domain/material/Material.java
package com.edumentor.domain.material;

import com.edumentor.domain.user.User;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.CreatedDate;

import java.time.LocalDateTime;

@Entity
@Table(name = "learning_materials")
@Data
@Builder
@NoArgsConstructor
@NoArgsConstructor
public class Material {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, length = 200)
    private String title;

    @Column(nullable = false, length = 20)
    @Enumerated(EnumType.STRING)
    private FileType fileType;

    @Column(nullable = false, length = 500)
    private String filePath;

    private Integer pageCount;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "uploaded_by")
    private User uploadedBy;

    @Column(nullable = false, length = 20)
    @Enumerated(EnumType.STRING)
    private ParseStatus parseStatus;

    @CreatedDate
    @Column(nullable = false, updatable = false)

```

```

    private LocalDateTime createdAt;

    public enum FileType {
        PDF, PPT
    }

    public enum ParseStatus {
        PENDING, COMPLETED, FAILED
    }
}

```

### 3.3 QASession 엔티티

```

// domain/qa/QASession.java
package com.edumentor.domain.qa;

import com.edumentor.domain.material.Material;
import com.edumentor.domain.user.User;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.Type;
import org.springframework.data.annotation.CreatedDate;

import java.time.LocalDateTime;

@Entity
@Table(name = "qa_sessions")
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class QASession {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private User user;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "material_id")
    private Material material;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String question;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String answer;

    @Column(columnDefinition = "JSONB")
    private String sources; // JSON 형태로 저장

    private Integer responseTimeMs;

    @CreatedDate
    @Column(nullable = false, updatable = false)
    private LocalDateTime createdAt;
}

```

### 3.4 Problem 엔티티

```

// domain/problem/Problem.java
package com.edumentor.domain.problem;

```



```

import com.edumentor.domain.material.Material;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.CreatedDate;

import java.time.LocalDateTime;

@Entity
@Table(name = "practice_problems")
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Problem {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "material_id")
    private Material material;

    @Column(nullable = false, length = 20)
    @Enumerated(EnumType.STRING)
    private Difficulty difficulty;

    @Column(length = 50)
    private String problemType;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String question;

    @Column(columnDefinition = "TEXT")
    private String answer;

    @Column(columnDefinition = "JSONB")
    private String hints; // JSON 배열

    @Column(columnDefinition = "JSONB")
    private String testCases; // JSON 배열

    @CreatedDate
    @Column(nullable = false, updatable = false)
    private LocalDateTime createdAt;

    public enum Difficulty {
        BEGINNER, INTERMEDIATE, ADVANCED
    }
}

```

## 4. Python 클라이언트

### 4.1 DTO 클래스

```

// client/dto/MaterialUploadRequest.java (신규)
package com.edumentor.client.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder

```

```

@NoArgsConstructor
@AllArgsConstructor
public class MaterialUploadRequest {
    private Long materialId;
    private String filePath; // 📁 파일 경로만 전달
}

```

```

// client/dto/MaterialUploadResponse.java (신규)
package com.edumentor.client.dto;

import lombok.Data;

@Data
public class MaterialUploadResponse {
    private Long materialId;
    private String status;
    private Integer pageCount;
    private Integer chunkCount;
    private String message;
}

```

```

// client/dto/QARequest.java
package com.edumentor.client.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class QARequest {
    private Long materialId;
    private String question;
}

```

```

// client/dto/QAResponse.java
package com.edumentor.client.dto;

import lombok.Data;
import java.util.List;
import java.util.Map;

@Data
public class QAResponse {
    private String answer;
    private List<Source> sources;
    private Integer responseTimeMs;

    @Data
    public static class Source {
        private Integer page;
        private String excerpt;
    }
}

```

```

// client/dto/ProblemRequest.java
package com.edumentor.client.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder

```

```

@NoArgsConstructor
@AllArgsConstructor
public class ProblemRequest {
    private Long materialId;
    private String difficulty; // BEGINNER, INTERMEDIATE, ADVANCED
    private Integer problemCount;
}

```

```

// client/dto/ProblemResponse.java
package com.edumentor.client.dto;

import lombok.Data;
import java.util.List;
import java.util.Map;

@Data
public class ProblemResponse {
    private List<ProblemDto> problems;
    private String difficulty;
    private Integer generatedCount;
    private Integer rejectedCount;

    @Data
    public static class ProblemDto {
        private String question;
        private String answer;
        private List<String> hints;
        private Integer difficultyScore;
        private String problemType;
        private List<Map<String, String>> testCases;
    }
}

```

## 4.2 PythonClient 구현

```

// client/PythonClient.java
package com.edumentor.client;

import com.edumentor.client.dto.*;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;
import org.springframework.web.reactive.function.client.WebClient;
import reactor.core.publisher.Mono;

import java.time.Duration;

@Slf4j
@Component
@RequiredArgsConstructor
public class PythonClient {

    private final WebClient pythonWebClient;

    /**
     * 🚀 자료 업로드 (파일 경로만 전달)
     */
    public Mono<MaterialUploadResponse> uploadMaterial(MaterialUploadRequest request) {
        log.info("Sending material upload to Python: materialId={}, path={}",
            request.getMaterialId(), request.getFilePath());

        return pythonWebClient.post()
            .uri("/qa/upload")
            .bodyValue(request)
            .retrieve()
            .bodyToMono(MaterialUploadResponse.class)
            .timeout(Duration.ofSeconds(60))
            .doOnSuccess(response ->
                log.info("Upload completed: materialId={}, chunks={}",
                    request.getMaterialId(), response.getChunkCount())
            );
    }
}

```

```

    )
    .doOnError(error ->
        log.error("Upload failed: {}", error.getMessage())
    );
}

/**
 * QA 질문 요청
 */
public Mono<QAResponse> askQuestion(QARequest request) {
    log.info("Calling Python QA service: material={}, question={}",
        request.getMaterialId(), request.getQuestion());

    return pythonWebClient.post()
        .uri("/qa/ask")
        .bodyValue(request)
        .retrieve()
        .bodyToMono(QAResponse.class)
        .timeout(Duration.ofSeconds(3))
        .doOnSuccess(response ->
            log.info("QA response received in {}ms", response.getResponseTimeMs())
        )
        .doOnError(error ->
            log.error("QA service error: {}", error.getMessage())
        );
}

/**
 * 문제 생성 요청
 */
public Mono<ProblemResponse> generateProblems(ProblemRequest request) {
    log.info("Calling Python Problem service: material={}, difficulty={}, count={}",
        request.getMaterialId(), request.getDifficulty(), request.getProblemCount());

    return pythonWebClient.post()
        .uri("/problems/generate")
        .bodyValue(request)
        .retrieve()
        .bodyToMono(ProblemResponse.class)
        .timeout(Duration.ofSeconds(30))
        .doOnSuccess(response ->
            log.info("Generated {} problems", response.getGeneratedCount())
        )
        .doOnError(error ->
            log.error("Problem service error: {}", error.getMessage())
        );
}
}

```

## 5. 컨트롤러

### 5.1 Material 컨트롤러 (신규)

```

// controller/MaterialController.java
package com.edumentor.controller;

import com.edumentor.client.PythonClient;
import com.edumentor.client.dto.MaterialUploadRequest;
import com.edumentor.domain.material.Material;
import com.edumentor.domain.material.MaterialService;
import com.edumentor.service.FileStorageService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

```

```

import reactor.core.publisher.Mono;

@Slf4j
@RestController
@RequestMapping("/api/materials")
@RequiredArgsConstructor
public class MaterialController {

    private final FileStorageService fileStorageService;
    private final MaterialService materialService;
    private final PythonClient pythonClient;

    /**
     * 🚀 학습 자료 업로드 (최적화)
     *
     * 1. Spring Boot가 파일을 공유 볼륨에 저장
     * 2. PostgreSQL에 메타데이터 저장 (PENDING)
     * 3. Python에 파일 경로만 전달
     * 4. Python이 파일 읽어 파싱
     * 5. 완료 후 COMPLETED 업데이트
     */
    @PostMapping("/upload")
    public Mono<ResponseEntity<?>> uploadMaterial(
        @RequestParam("file") MultipartFile file,
        @RequestParam("title") String title,
        @AuthenticationPrincipal UserDetails userDetails
    ) {
        log.info("User {} uploading: {}", userDetails.getUsername(), title);

        try {
            // 1. 파일을 공유 볼륨에 저장
            String filePath = fileStorageService.storeFile(file);
            String fileType = fileStorageService.getFileType(file.getOriginalFilename());

            // 2. PostgreSQL에 메타데이터 저장
            Material material = materialService.createMaterial(
                userDetails.getUsername(),
                title,
                fileType,
                filePath,
                Material.ParseStatus.PENDING
            );

            log.info("Material saved: id={}, path={}", material.getId(), filePath);

            // 3. Python에 파일 경로만 전달
            MaterialUploadRequest request = MaterialUploadRequest.builder()
                .materialId(material.getId())
                .filePath(filePath)
                .build();

            return pythonClient.uploadMaterial(request)
                .doOnSuccess(response -> {
                    materialService.updateParseStatus(
                        material.getId(),
                        Material.ParseStatus.COMPLETED,
                        response.getPageCount()
                    );
                    log.info("Parsing completed: id={}", material.getId());
                })
                .map(response -> ResponseEntity.ok(material))
                .onErrorResume(error -> {
                    log.error("Parsing failed: {}", error.getMessage());
                    materialService.updateParseStatus(
                        material.getId(),
                        Material.ParseStatus.FAILED,
                        null
                    );
                    return Mono.just(ResponseEntity.internalServerError().build());
                })
                .thenReturn(ResponseEntity.ok(material));

        } catch (Exception e) {
            log.error("Upload failed: {}", e.getMessage());
            return Mono.just(ResponseEntity.internalServerError().build());
        }
    }
}

```

```

        log.error("Upload failed: {}", e.getMessage(), e);
        return Mono.just(ResponseEntity.badRequest().body(e.getMessage()));
    }
}

@GetMapping
public ResponseEntity<?> getMaterials(@AuthenticationPrincipal UserDetails userDetails) {
    var materials = materialService.getUserMaterials(userDetails.getUsername());
    return ResponseEntity.ok(materials);
}

@GetMapping("/{id}")
public ResponseEntity<?> getMaterial(@PathVariable Long id) {
    var material = materialService.getMaterialById(id);
    return ResponseEntity.ok(material);
}

@DeleteMapping("/{id}")
public ResponseEntity<?> deleteMaterial(
    @PathVariable Long id,
    @AuthenticationPrincipal UserDetails userDetails
) {
    materialService.deleteMaterial(id, userDetails.getUsername());
    return ResponseEntity.ok().build();
}
}

```

## 5.2 QA 컨트롤러

```

// controller/QAController.java
package com.edumentor.controller;

import com.edumentor.client.PythonClient;
import com.edumentor.client.dto.QARequest;
import com.edumentor.client.dto.QAResponse;
import com.edumentor.domain.qa.QAService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.*;
import reactor.core.publisher.Mono;

@Slf4j
@RestController
@RequestMapping("/api/qa")
@RequiredArgsConstructor
public class QAController {

    private final PythonClient pythonClient;
    private final QAService qaService;

    @PostMapping("/ask")
    public Mono<ResponseEntity<QAResponse>> askQuestion(
        @RequestBody QARequest request,
        @AuthenticationPrincipal UserDetails userDetails
    ) {
        log.info("User {} asking question for material {}",
            userDetails.getUsername(), request.getMaterialId());

        return pythonClient.askQuestion(request)
            .doOnSuccess(response -> {
                // DB에 저장 (비동기)
                qaService.saveSession(
                    userDetails.getUsername(),
                    request.getMaterialId(),
                    request.getQuestion(),
                    response
                );
            })
    }
}

```

```

        .map(ResponseEntity::ok)
        .onErrorResume(error -> {
            log.error("QA error: {}", error.getMessage());
            return Mono.just(ResponseEntity.internalServerError().build());
        });
    }

@GetMapping("/history")
public ResponseEntity<?> getQAHistory(
    @AuthenticationPrincipal UserDetails userDetails,
    @RequestParam(required = false) Long materialId
) {
    var history = qaService.getUserHistory(
        userDetails.getUsername(),
        materialId
    );
    return ResponseEntity.ok(history);
}
}

```

## 5.2 문제 생성 컨트롤러

```

// controller/ProblemController.java
package com.edumentor.controller;

import com.edumentor.client.PythonClient;
import com.edumentor.client.dto.ProblemRequest;
import com.edumentor.client.dto.ProblemResponse;
import com.edumentor.domain.problem.ProblemService;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.web.bind.annotation.*;
import reactor.core.publisher.Mono;

@Slf4j
@RestController
@RequestMapping("/api/problems")
@RequiredArgsConstructor
public class ProblemController {

    private final PythonClient pythonClient;
    private final ProblemService problemService;

    @PostMapping("/generate")
    public Mono<ResponseEntity<ProblemResponse>> generateProblems(
        @RequestBody ProblemRequest request,
        @AuthenticationPrincipal UserDetails userDetails
    ) {
        log.info("User {} generating {} {} problems for material {}",
            userDetails.getUsername(),
            request.getProblemCount(),
            request.getDifficulty(),
            request.getMaterialId());

        return pythonClient.generateProblems(request)
            .doOnSuccess(response -> {
                // DB에 저장 (비동기)
                problemService.saveProblems(
                    request.getMaterialId(),
                    response.getProblems()
                );
            })
            .map(ResponseEntity::ok)
            .onErrorResume(error -> {
                log.error("Problem generation error: {}", error.getMessage());
                return Mono.just(ResponseEntity.internalServerError().build());
            });
    }
}

```



```

@GetMapping("/list")
public ResponseEntity<?> getProblemsByMaterial(
    @RequestParam Long materialId,
    @RequestParam(required = false) String difficulty
) {
    var problems = problemService.getProblemsByMaterial(
        materialId,
        difficulty
    );
    return ResponseEntity.ok(problems);
}
}

```

## 6. 서비스 레이어

### 6.1 QA 서비스

```

// domain/qa/QAService.java
package com.edumentor.domain.qa;

import com.edumentor.client.dto.QAResponse;
import com.edumentor.domain.material.MaterialRepository;
import com.edumentor.domain.user.UserRepository;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Slf4j
@Service
@RequiredArgsConstructor
public class QAService {

    private final QARepository qaRepository;
    private final UserRepository userRepository;
    private final MaterialRepository materialRepository;
    private final ObjectMapper objectMapper;

    @Transactional
    public void saveSession(
        String username,
        Long materialId,
        String question,
        QAResponse response
    ) {
        var user = userRepository.findByUsername(username)
            .orElseThrow(() -> new RuntimeException("User not found"));

        var material = materialRepository.findById(materialId)
            .orElseThrow(() -> new RuntimeException("Material not found"));

        try {
            var session = QASession.builder()
                .user(user)
                .material(material)
                .question(question)
                .answer(response.getAnswer())
                .sources(objectMapper.writeValueAsString(response.getSources()))
                .responseTimeMs(response.getResponseTimeMs())
                .build();

            qaRepository.save(session);
            log.info("Saved QA session for user {}", username);
        } catch (Exception e) {

```



```

        log.error("Failed to save QA session: {}", e.getMessage());
    }
}

public List<QASession> getUserHistory(String username, Long materialId) {
    var user = userRepository.findByUsername(username)
        .orElseThrow(() -> new RuntimeException("User not found"));

    if (materialId != null) {
        return qaRepository.findByUserAndMaterial_IdOrderByCreatedAtDesc(
            user, materialId
        );
    } else {
        return qaRepository.findByUserOrderByCreatedAtDesc(user);
    }
}
}

```

## 6.2 문제 서비스

```

// domain/problem/ProblemService.java
package com.edumentor.domain.problem;

import com.edumentor.client.dto.ProblemResponse;
import com.edumentor.domain.material.MaterialRepository;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Slf4j
@Service
@RequiredArgsConstructor
public class ProblemService {

    private final ProblemRepository problemRepository;
    private final MaterialRepository materialRepository;
    private final ObjectMapper objectMapper;

    @Transactional
    public void saveProblems(
        Long materialId,
        List<ProblemResponse.ProblemDto> problemDtos
    ) {
        var material = materialRepository.findById(materialId)
            .orElseThrow(() -> new RuntimeException("Material not found"));

        try {
            for (var dto : problemDtos) {
                var problem = Problem.builder()
                    .material(material)
                    .difficulty(Problem.Difficulty.valueOf(dto.getDifficulty()))
                    .problemType(dto.getProblemType())
                    .question(dto.getQuestion())
                    .answer(dto.getAnswer())
                    .hints(objectMapper.writeValueAsString(dto.getHints()))
                    .testCases(objectMapper.writeValueAsString(dto.getTestCases()))
                    .build();

                problemRepository.save(problem);
            }

            log.info("Saved {} problems for material {}", problemDtos.size(), materialId);
        } catch (Exception e) {
            log.error("Failed to save problems: {}", e.getMessage());
        }
    }
}

```

```
public List<Problem> getProblemsByMaterial(Long materialId, String difficulty) {
    if (difficulty != null) {
        return problemRepository.findByMaterial_IdAndDifficulty(
            materialId,
            Problem.Difficulty.valueOf(difficulty)
        );
    } else {
        return problemRepository.findByMaterial_Id(materialId);
    }
}
}
```

---

## 7. 테스트

### 7.1 통합 테스트

```
// controller/QAControllerTest.java
package com.edumentor.controller;

import com.edumentor.client.PythonClient;
import com.edumentor.client.dto.QARequest;
import com.edumentor.client.dto.QAResponse;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import reactor.test.StepVerifier;

import java.time.Duration;

@SpringBootTest
class QAControllerTest {

    @Autowired
    private PythonClient pythonClient;

    @Test
    void testQARequest() {
        var request = QARequest.builder()
            .materialId(1L)
            .question("JPA Entity란 무엇인가요?")
            .build();

        StepVerifier.create(pythonClient.askQuestion(request))
            .expectNextMatches(response ->
                response.getAnswer() != null &&
                !response.getAnswer().isEmpty() &&
                response.getResponseTimeMs() < 2000
            )
            .verifyComplete();
    }
}
```

---

## 8. 실행

### 8.1 빌드 및 실행

```
# 빌드
./gradlew clean build

# 실행
java -jar build/libs/edumentor-backend-1.0.0.jar

# 또는
./gradlew bootRun
```

## 8.2 API 테스트

```
# 1. QA 테스트
curl -X POST http://localhost:8080/api/qa/ask \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer {token}" \
  -d '{
    "materialId": 1,
    "question": "JPA Entity란 무엇인가요?"
  }'

# 2. 문제 생성 테스트
curl -X POST http://localhost:8080/api/problems/generate \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer {token}" \
  -d '{
    "materialId": 1,
    "difficulty": "BEGINNER",
    "problemCount": 3
  }'
```

---

### 체크리스트

- ☐ Spring Boot 프로젝트 생성 (Java 21)
- ☐ PostgreSQL 연동
- ☐ WebClient 설정 (Python 서버 연동)
- ☐ 엔티티 작성 (User, Material, QASession, Problem)
- ☐ Repository 작성
- ☐ PythonClient 구현
- ☐ 컨트롤러 작성 (QA, Problem)
- ☐ 서비스 레이어 작성
- ☐ 예외 처리
- ☐ 통합 테스트
- ☐ API 문서화 (Swagger)

---

작성일: 2025-10-28