[OCR] Java Spring Framework

1. OcrController.java

OcrController

```
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.extern.slf4j.Slf4j;
import org.springframework.core.io.FileSystemResource;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.util.LinkedMultiValueMap;
import org.springframework.util.MultiValueMap;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.io.FileInputStream;

import org.apache.commons.io.IOUtils;

@Slf4j
@RestController
public class OcrController {
    private static final ObjectMapper objectMapper = new ObjectMapper();
    private static final String ROOT_DIR = ".."; //  Root
    private static final String OCR_RESULT_DIR = ROOT_DIR + "/tmp/work"; // OCR

    private static final String OCR_API_URL = ""; // OCR API URL
    private static final String OCR_API_KEY = ""; // OCR API Key

    /**
     * OCR API .
     * @param file
     * @param page
     * @return
     */
    @PostMapping("/requestOCR")
    public ResultModel ocrRequest(
            @RequestParam("file") MultipartFile file,
            @RequestParam(value = "page", required = false, defaultValue = "0") int page) {
        ResultModel model = new ResultModel();
        model.setResult("false");

        try {
            OCRResult ocrResult = getOcrResult(file, page);
            String fileName = downloadImage(ocrResult.getMasked_image());

            String imagePath = String.format("%s/viewImage?fileName=%s", ServletUriComponentsBuilder.
fromCurrentContextPath().toUriString(), fileName);
            model.setResult(ocrResult);
            model.setImagePath(imagePath);
            return model;
        } catch (IOException e) {
            log.warn(e.getMessage(), e);
            return model;
        }
    }

    /**
     *    .
     * @param fileName
     * @return
     * @throws IOException
     */
```

```
    @GetMapping("/viewImage")
    public ResponseEntity<byte[]> getImageByteArray(String fileName) throws IOException {
        InputStream imageStream = new FileInputStream(String.format("%s/%s", OCR_RESULT_DIR, fileName));
        byte[] imageByteArray = IOUtils.toByteArray(imageStream);
        imageStream.close();
        return new ResponseEntity<>(imageByteArray, HttpStatus.OK);
    }

    /**
     * OCR result .
     * @param file
     * @param page
     * @return
     * @throws IOException
     */
    private OCRResult getOcrResult(MultipartFile file, int page) throws IOException {
        OCRResult ocrResult = requestOCR(file, page);

        if (ocrResult == null) {
            throw new IOException("OCR request failed.");
        }
        return ocrResult;
    }

    /**
     * OCR API .
     * @param file
     * @param page
     * @return
     * @throws IOException
     */
    private OCRResult requestOCR(MultipartFile file, int page) throws IOException {
        MultiValueMap<String, Object> body = new LinkedMultiValueMap<>();
        File f = new File("/C:/image/" + file.getOriginalFilename());
        file.transferTo(f);
        FileSystemResource fileSystemResource = new FileSystemResource(f);
        body.add("api_key", OCR_API_KEY);
        body.add("page_index", String.valueOf(page));
        body.add("image", fileSystemResource);
        body.add("type", "upload");
        body.add("coord", "origin");
        body.add("skew", "image");
        body.add("boxes_type", "all");
        body.add("save_mask", true);
        body.add("textout", true);
        body.add("recog_form", true);
        body.add("extract_table", true);

        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.MULTIPART_FORM_DATA);
        HttpEntity<MultiValueMap<String, Object>> requestEntity = new HttpEntity<>(body, headers);

        String serverUrl = OCR_API_URL + "/ocr";
        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> response = restTemplate.postForEntity(serverUrl, requestEntity, String.class);

        return this.objectMapper.readValue(this.objectMapper.readTree(response.getBody()).get("result").
toString(), OCRResult.class);
    }

    /**
     * OCR   .
     * @param fileName
     * @return
     * @throws IOException
     */
    private String downloadImage(String fileName) throws IOException {
        URL url = new URL(OCR_API_URL + "/out/" + fileName);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

```
        String formData = "api_key=" + OCR_API_KEY;
        connection.setDoOutput(true);

        try (OutputStream outputStream = connection.getOutputStream()) {
            byte[] formDataBytes = formData.getBytes("UTF-8");
            outputStream.write(formDataBytes, 0, formDataBytes.length);
        }

        int statusCode = connection.getResponseCode();
        String statusMessage = connection.getResponseMessage();
        if (statusCode == 200) {
            System.out.println("OCR image download ");
            String outFilePath = String.format("%s/%s", OCR_RESULT_DIR, fileName);
            try (InputStream inputStream = connection.getInputStream();
                 FileOutputStream outputStream = new FileOutputStream(outFilePath)) {
                byte[] buffer = new byte[1024];
                int bytesRead;
                while ((bytesRead = inputStream.read(buffer)) != -1) {
                    outputStream.write(buffer, 0, bytesRead);
                }
            }

            return fileName;
        } else {
            System.out.println("OCR image download : " + statusCode);
            throw new IOException(statusMessage);
        }
    }
}
```

2. OcrResult.java

OcrResult

```
import lombok.Data;

import java.util.List;

@Data
public class OCRResult {
    private List<List<Object>> block_boxes;
    private List<List<Object>> boxes;
    private List<List<Object>> char_boxes;
    private String csv_file_name;
    private double dur;
    private String fid;
    private String final_file_name;
    private String form_excel_file;
    private String full_text;
    private int height;
    private List<List<Object>> line_boxes;
    private String masked_image;
    private List<Object> matched_boxes;
    private List<Object> matched_forms;
    private int page_index;
    private double rotate;
    private String table_excel_file;
    private List<Object> table_list;
    private int total_pages;
    private int width;
}
```

3. ResultModel.java

ResultModel

```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@AllArgsConstructor
@NoArgsConstructor
@Data
public class ResultModel {
        Object result;
        String imagePath;
}
```

4. TableData.java

TableData

```
import lombok.Data;

import java.util.List;

@Data
public class TableData {
    private List<Double> boundary;
    private List<Object> cells;
    private double confidence;
    private String csv_file_name;
    private String excel_file_name;
    private String html_file_name;
    private int[] shape;
    private String table_id;
}
```