Java Spring Framework

.

.

- Java Spring Framework
- Java Servlet
- ASP.NET (C#)
- ASP(Classic)
- PHP
- PHP4
- Django
- Ruby On Rails
- Wordpress plugin

1. , ,

upload

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
import java.io.*;
import java.util.*;


@Controller
public class UploadController {
    static String IMAGE_UPLOAD_DIR_REL_PATH = "uploads";

    @RequestMapping(value = "/uploadFile.do", method = RequestMethod.POST)
        @ResponseBody
        public Map<String, Object> uploadFile(HttpServletRequest request, @RequestParam("file")
MultipartFile file)
                throws IOException {
            String ROOT_ABS_PATH = request.getSession().getServletContext().getRealPath("");
            String UPLOAD_DIR_ABS_PATH = ROOT_ABS_PATH + File.separator + IMAGE_UPLOAD_DIR_REL_PATH;

            makeDirectory(UPLOAD_DIR_ABS_PATH);

            String fileName = file.getOriginalFilename();
            String ext = "";
                String contentType = file.getContentType();
                if(contentType != null) {
                    ext = "." + contentType.substring(contentType.lastIndexOf('/') + 1);
                } else if (fileName.lastIndexOf('.') > 0) {
                    ext = fileName.substring(fileName.lastIndexOf('.'));
                }
                if (ext.indexOf(".jpeg") > -1) { // jpg  jpeg jpg
                    ext = ".jpg";
                }
            String saveFileName = UUID.randomUUID().toString() + ext;
            String saveFileAbsPath = UPLOAD_DIR_ABS_PATH + File.separator + saveFileName;

            writeFile(saveFileAbsPath, file.getBytes());

            Map<String, Object> map = new HashMap<String, Object>();

            //    uploadPath  .
            map.put("uploadPath", "uploads/" + saveFileName);
```

```
            return map;
        }

        /**
         *  .
         */
        private static void writeFile(String path, byte[] bytes) throws IOException {
            OutputStream os = null;
            try {
                os = new FileOutputStream(path);
                os.write(bytes);
            } finally {
                if (os != null) os.close();
            }
        }

        /**
         *     .
         */
        private static void makeDirectory(String dirPath) {
            File dir = new File(dirPath);
            if (!dir.exists()) {
                dir.mkdir();
            }
        }
}
```

2. HWP, MS ,

import

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
import java.io.*;
import java.util.*;
import java.util.zip.InflaterInputStream;

@Controller
public class ImportController {
    static String DOC_UPLOAD_DIR_REL_PATH = "uploads" + File.separator + "docs";
        static String OUTPUT_DIR_REL_PATH = "uploads" + File.separator + "output";

    @RequestMapping(value = "/importDoc.do", method = RequestMethod.POST)
    @ResponseBody
    public Map<String, Object> importDoc(HttpServletRequest request, @RequestParam("file") MultipartFile
importFile)
                throws IOException {
            String ROOT_ABS_PATH = request.getSession().getServletContext().getRealPath("");
            String UPLOAD_DIR_ABS_PATH = ROOT_ABS_PATH + File.separator + DOC_UPLOAD_DIR_REL_PATH;

            makeDirectory(UPLOAD_DIR_ABS_PATH);

            String fileName = importFile.getOriginalFilename();
            String inputFileAbsPath = UPLOAD_DIR_ABS_PATH + File.separator + fileName;

            writeFile(inputFileAbsPath, importFile.getBytes());

            //
            Calendar cal = Calendar.getInstance();
            String yearMonth = String.format("%04d%02d", cal.get(Calendar.YEAR), cal.get(Calendar.MONTH)+1);
```

```
            String uuid = UUID.randomUUID().toString();
            String worksDirAbsPath = ROOT_ABS_PATH + File.separator + OUTPUT_DIR_REL_PATH + File.separator +
yearMonth + File.separator + uuid;

            makeDirectory(worksDirAbsPath);

            //
            executeConverter(inputFileAbsPath, worksDirAbsPath);

            //    .
            deleteFile(inputFileAbsPath);

            //  pb  serialzie
            // v2.3.0   document.word.pb document.pb
            String pbAbsPath = worksDirAbsPath + File.separator + "document.pb";
            Integer[] serializedData = serializePbData(pbAbsPath);

            // pb
            // v2.3.0   document.word.pb document.pb
            deleteFile(pbAbsPath);

            Map<String, Object> map = new HashMap<String, Object>();
            map.put("serializedData", serializedData);
            //    importPath  .
            // OUTPUT_DIR_REL_PATH    .
            map.put("importPath", "uploads/output/" + yearMonth + "/" + uuid);

            return map;
        }

        /**
         *    .
         */
        public static int executeConverter(String inputFilePath, String outputFilePath) {
            String SEDOC_CONVERTER_DIR_ABS_PATH = "   ";
                String FONT_DIR_ABS_PATH = SEDOC_CONVERTER_DIR_ABS_PATH + File.separator + "fonts";
                String TEMP_DIR_ABS_PATH = SEDOC_CONVERTER_DIR_ABS_PATH + File.separator + "temp";
                String SEDOC_CONVERTER_ABS_PATH = SEDOC_CONVERTER_DIR_ABS_PATH + File.separator +
"sedocConverter_exe";
                // String SEDOC_CONVERTER_ABS_PATH = SEDOC_CONVERTER_DIR_ABS_PATH + File.separator +
"sedocConverter.exe";// window server

                makeDirectory(TEMP_DIR_ABS_PATH);
                makeDirectory(FONT_DIR_ABS_PATH);

                //
                String[] cmd = {SEDOC_CONVERTER_ABS_PATH, "-f", FONT_DIR_ABS_PATH, inputFilePath,
outputFilePath, TEMP_DIR_ABS_PATH};
                try {
                    Timer t = new Timer();
                    Process proc = Runtime.getRuntime().exec(cmd);

                    TimerTask killer = new TimeoutProcessKiller(proc);
                    t.schedule(killer, 20000); // 20 ( 20     )

                    int exitValue = proc.waitFor();
                    killer.cancel();

                    return exitValue;
                } catch (Exception e) {
                    e.printStackTrace();
                    return -1;
                }
        }

        /**
         *       Serialize .
         */
        public static Integer[] serializePbData(String pbFilePath) throws IOException {
            List<Integer> serializedData = new ArrayList<Integer>();
            FileInputStream fis = null;
```

```
    InflaterInputStream ifis = null;
    Integer[] data = null;

    try {
        fis = new FileInputStream(pbFilePath);
        fis.skip(16);

        ifis = new InflaterInputStream(fis);
        byte[] buffer = new byte[1024];

        int len;
        while ((len = ifis.read(buffer)) != -1) {
            for (int i = 0; i < len; i++) {
                serializedData.add(buffer[i] & 0xFF);
            }
        }

        data = serializedData.toArray(new Integer[serializedData.size()]);
    } finally {
        if (ifis != null) ifis.close();
        if (fis != null) fis.close();
    }

    return data;
}

/**
 *  .
 */
private static void writeFile(String path, byte[] bytes) throws IOException {
    OutputStream os = null;
    try {
        os = new FileOutputStream(path);
        os.write(bytes);
    } finally {
        if (os != null) os.close();
    }
}

/**
 *  .
 */
private static void deleteFile(String path) {
        File file = new File(path);
    if (file.exists()) {
                file.delete();
        }
}

/**
 *     .
 */
private static void makeDirectory(String dirPath) {
    File dir = new File(dirPath);
    if (!dir.exists()) {
        dir.mkdir();
    }
}

private static class TimeoutProcessKiller extends TimerTask {
    private Process p;

    public TimeoutProcessKiller(Process p) {
        this.p = p;
    }

    @Override
    public void run() {
        p.destroy();
    }
}
```

```
}
```

- Java Spring Framework
- Java Servlet
- ASP.NET (C#)
- ASP(Classic)
- PHP
- PHP4
- Django
- Ruby On Rails
- Wordpress plugin