Java Spring Framework Example

Among the following codes provided as the guide, file upload part is a **sample code** and has insufficient security.

As for the file upload, please use the code used within your project and refer the following code to handle the system link.

Link Guide by Environment

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

1. Uploading Images, Videos and Files

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", 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) { // jpeg is converted into jpg as jpg is more frequently used
                    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>();

            // Include the path to which the browser can access in uploadPath and transfer it.
            map.put("uploadPath", "uploads/" + saveFileName);

            return map;
        }

        /**
         * Write a file.
         */
        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();
            }
        }

        /**
         * When there is no directory, create a directory.
         */
        private static void makeDirectory(String dirPath) {
            File dir = new File(dirPath);
            if (!dir.exists()) {
                dir.mkdir();
            }
        }
}
```

2. Importing HWP, MS Word and Excel Documents

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", 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());

            // Create paths to store conversion results by file
            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);

            // Document conversion
            executeConverter(inputFileAbsPath, worksDirAbsPath);

            // Delete the original file once the conversion is completed.
            deleteFile(inputFileAbsPath);

            // Read the converted pb files and serialize them.
            // Since v2.3.0, the file name is changed from document.word.pb to document.pb
            String pbAbsPath = worksDirAbsPath + File.separator + "document.pb";
            Integer[] serializedData = serializePbData(pbAbsPath);

            // Delete pb file
            // Since v2.3.0, the file name is changed from document.word.pb to document.pb
            deleteFile(pbAbsPath);

            Map<String, Object> map = new HashMap<String, Object>();
            map.put("serializedData", serializedData);
            // Contain the path to which the browser can access in importPath and transfer it.
            // Modify according to the path in OUTPUT_DIR_REL_PATH.
            map.put("importPath", "uploads/output/" + yearMonth + "/" + uuid);

            return map;
        }

        /**
         * Run document conversion module.
         */
        public static int executeConverter(String inputFilePath, String outputFilePath) {
            String SEDOC_CONVERTER_DIR_ABS_PATH = "Absolute path of the directory in which the conversion
module exists";
                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";// For window server

                makeDirectory(TEMP_DIR_ABS_PATH);
                makeDirectory(FONT_DIR_ABS_PATH);

                // Configure conversion command
                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 seconds (the process shall be terminated if the
conversion is not completed within 20 seconds)

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

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

```
/**
 * Serialize the conversion result after running document module.
 */
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;
}

/**
 * Write a file.
 */
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();
    }
}

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

/**
 * When there is no directory, create a directory.
 */
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();
            }
        }
}
```

See also

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