

Java Servlet Example

- [1. Servlet Configuration](#)
- [2. Uploading Images, Videos and Files](#)
- [3. Importing HWP, MS Word and Excel Documents](#)

1. Servlet Configuration



Note

You may use `@WebServlet` annotation in Servlet 3.0 or above.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
id="WebApp_ID" version="3.1">
  <display-name>synapeditor</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>ImportServlet</servlet-name>
    <servlet-class>com.synap.synapeditor.ImportServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ImportServlet</servlet-name>
    <url-pattern>/importDoc</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>UploadServlet</servlet-name>
    <servlet-class>com.synap.synapeditor.UploadServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>UploadServlet</servlet-name>
    <url-pattern>/uploadFile</url-pattern>
  </servlet-mapping>
</web-app>
```

2. Uploading Images, Videos and Files



Caution

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.

upload

```
import com.google.gson.GsonBuilder;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
```

```

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

public class UploadServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;
    private static final String UPLOAD_DIR_REL_PATH = "uploads";

    private static final int MEMORY_THRESHOLD = 1024 * 1024 * 3;
    private static final int MAX_FILE_SIZE = 1024 * 1024 * 40;
    private static final int MAX_REQUEST_SIZE = 1024 * 1024 * 50;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException {

        if (ServletFileUpload.isMultipartContent(request)) {

            DiskFileItemFactory factory = new DiskFileItemFactory();
            factory.setSizeThreshold(MEMORY_THRESHOLD);
            factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

            ServletFileUpload upload = new ServletFileUpload(factory);
            upload.setHeaderEncoding("UTF-8");
            upload.setFileSizeMax(MAX_FILE_SIZE);
            upload.setSizeMax(MAX_REQUEST_SIZE);

            String ROOT_ABS_PATH = request.getSession().getServletContext().getRealPath("");
            String UPLOAD_DIR_ABS_PATH = ROOT_ABS_PATH + File.separator + UPLOAD_DIR_REL_PATH;

            makeDirectory(UPLOAD_DIR_ABS_PATH);

            String storeFileName = "";
            try {
                List<FileItem> formItems = upload.parseRequest(request);

                if (formItems != null && formItems.size() > 0) {
                    for (FileItem item : formItems) {
                        if (!item.isFormField()) {
                            String ext = item.getName().substring(item.getName().lastIndexOf('.'));
                            storeFileName = UUID.randomUUID().toString() + ext;
                            String storeFileAbsPath = UPLOAD_DIR_ABS_PATH + File.separator +
storeFileName;

                            item.write(new File(storeFileAbsPath)); // Save the file
                        }
                    }
                }
            } catch (Exception ex) {
                ex.printStackTrace();
            }

            Map map = new HashMap();
            map.put("uploadPath", "uploads/" + storeFileName);

            PrintWriter out = response.getWriter();
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");

            out.print(new GsonBuilder().create().toJson(map));
            out.flush();

        }

    }

    /**

```

```

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

```

3. Importing HWP, MS Word and Excel Documents



Caution

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.

import

```

import com.google.gson.GsonBuilder;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Timer;
import java.util.TimerTask;
import java.util.UUID;
import java.util.zip.InflaterInputStream;

public class ImportServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;
    private static final String UPLOAD_DIR_REL_PATH = "import";

    private static final int MEMORY_THRESHOLD = 1024 * 1024 * 3;
    private static final int MAX_FILE_SIZE = 1024 * 1024 * 40;
    private static final int MAX_REQUEST_SIZE = 1024 * 1024 * 50;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException {

        if (ServletFileUpload.isMultipartContent(request)) {

            DiskFileItemFactory factory = new DiskFileItemFactory();
            factory.setSizeThreshold(MEMORY_THRESHOLD);
            factory.setRepository(new File(System.getProperty("java.io.tmpdir")));

            ServletFileUpload upload = new ServletFileUpload(factory);
            upload.setHeaderEncoding("UTF-8");
            upload.setFileSizeMax(MAX_FILE_SIZE);
            upload.setSizeMax(MAX_REQUEST_SIZE);

```

```

String ROOT_ABS_PATH = request.getSession().getServletContext().getRealPath("");
String UPLOAD_DIR_ABS_PATH = ROOT_ABS_PATH + File.separator + UPLOAD_DIR_REL_PATH;

makeDirectory(UPLOAD_DIR_ABS_PATH);

String storeFileAbsPath = "";
try {
    List<FileItem> formItems = upload.parseRequest(request);

    if (formItems != null && formItems.size() > 0) {
        for (FileItem item : formItems) {
            if (!item.isFormField()) {
                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 storeFileName = UUID.randomUUID().toString() + ext;
                    storeFileAbsPath = UPLOAD_DIR_ABS_PATH + File.separator + storeFileName;
                    System.out.println(storeFileAbsPath);

                    item.write(new File(storeFileAbsPath)); // Save the file
                }
            }
        }
    } catch (Exception ex) {
        ex.printStackTrace();
    }

    // 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 outputDirAbsPath = UPLOAD_DIR_ABS_PATH + File.separator + yearMonth + File.
separator + uuid;

    makeDirectory(outputDirAbsPath);

    // Execute conversion
    executeConverter(storeFileAbsPath, outputDirAbsPath);

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

    // 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 = outputDirAbsPath + File.separator + "document.pb";
    Integer[] serializedData = serializePbData(pbAbsPath);

    // Delete pb file
    deleteFile(pbAbsPath);

    Map map = new HashMap();
    map.put("serializedData", serializedData);
    map.put("importPath", UPLOAD_DIR_REL_PATH + "/" + yearMonth + "/" + uuid);

    PrintWriter out = response.getWriter();
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");

    out.print(new GsonBuilder().create().toJson(map));
    out.flush();

```

```

    }

}

/**
 * Run document conversion module.
 */
protected static int executeConverter(String inputFilePath, String outputPath) {
    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,
outputPath, 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.
 */
protected 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;
}

```

```

    }

    /**
     * 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](#)