How can I upload files to a server using JSP/Servlet?

Asked 14 years, 9 months ago Modified 1 year, 2 months ago Viewed 656k times

720

How can I upload files to server using JSP/Servlet?

I tried this:

```
<form action="upload" method="post">
    <input type="text" name="description" />
    <input type="file" name="file" />
    <input type="submit" />
</form>
```

However, I only get the file name, not the file content. When I add `enctype="multipart/form-data"` to the `<form>`, then `request.getParameter()` returns `null`.

During research I stumbled upon Apache Common FileUpload. I tried this:

```
FileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload upload = new ServletFileUpload(factory);
List items = upload.parseRequest(request); // This line is where it died.
```

Unfortunately, the servlet threw an exception without a clear message and cause. Here is the stacktrace:

```
SEVERE: Servlet.service() for servlet UploadServlet threw exception
javax.servlet.ServletException: Servlet execution threw an exception
    at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilter
    at
org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.ja
    at
org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:233
    at
org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:191
    at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:127)
    at
org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:102)
    at
org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:109)
    at
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:298)
    at
org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:852)
    at
```

```
org.apache.coyote.http11.Http11Protocol$Http11ConnectionHandler.process(Http11Prot
    at org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:489)
    at java.lang.Thread.run(Thread.java:637)
```

jsp servlets file-upload

Share
Improve this question
Follow

edited Oct 5, 2023 at 16:17

BalusC
1

asked Mar 11, 2010 at 4:07

Thang Pham
38.7k ● 78 ● 207 ● 289

Perhaps this article will be helpful: baeldung.com/upload-file-servlet – Adam Gerard Jan 15, 2019 at 4:31

1 @Adam: They copied from my answer and added a sleuth of advertising on top of it in an attempt to earn money with it. Yeah, great article .. – BalusC Feb 24, 2021 at 11:24

1 No, actually nothing was copied. I wrote the first draft of that article along with the supplemental code. The core reference documentation can be found here: commons.apache.org/proper/commons-fileupload/using.html (and is linked to and cited in the article). Examples are partly reprised from the core reference document (which is the point of reference documentation - i.e. to be a point of reference) but not in their entirety (note that the reference docs don't go into much detail). Thanks! – Adam Gerard Mar 29, 2021 at 15:33

check this sandny.com/2017/05/18/servlet-file-upload – ricky Jul 26, 2021 at 16:50

14 Answers

Sorted by: Highest score (default)

1259

+200

Introduction

To browse and select a file for upload you need a HTML `<input type="file">` field in the form. As stated in the HTML specification you have to use the `POST` method and the `enctype` attribute of the form has to be set to `"multipart/form-data"`.

```
<form action="upload" method="post" enctype="multipart/form-data">
    <input type="text" name="description" />
    <input type="file" name="file" />
    <input type="submit" />
</form>
```

After submitting such a form, the binary multipart form data is available in the request body in a different format than when the `enctype` isn't set.

Before Servlet 3.0 (Dec 2009), the Servlet API didn't natively support `multipart/form-data`. It supports only the default form enctype of `application/x-www-form-urlencoded`. The `request.getParameter()` and consorts would all return

`null` when using multipart form data. This is where the well known Apache Commons FileUpload came into the picture.

Don't manually parse it!

You can in theory parse the request body yourself based on `ServletRequest#getInputStream()`. However, this is a precise and tedious work which requires precise knowledge of RFC2388. You shouldn't try to do this on your own or copypaste some homegrown library-less code found elsewhere on the Internet. Many online sources have failed hard in this, such as roseindia.net. See also uploading of pdf file. You should rather use a real library which is used (and implicitly tested!) by millions of users for years. Such a library has proven its robustness.

When you're already on Servlet 3.0 or newer, use native API

If you're using at least Servlet 3.0 (Tomcat 7, Jetty 9, JBoss AS 6, GlassFish 3, etc, they exist already since 2010), then you can just use standard API provided `HttpServletRequest#getPart()` to collect the individual multipart form data items (most Servlet 3.0 implementations *actually* use Apache Commons FileUpload under the covers for this!). Also, normal form fields are available by `getParameter()` the usual way.

First annotate your servlet with `@MultipartConfig` in order to let it recognize and support `multipart/form-data` requests and thus get `getPart()` to work:

```
@WebServlet("/upload")
@MultipartConfig
public class UploadServlet extends HttpServlet {
    // ...
}
```

Then, implement its `doPost()` as follows:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    String description = request.getParameter("description"); // Retrieves
<input type="text" name="description">
    Part filePart = request.getPart("file"); // Retrieves <input type="file"
name="file">
    String fileName =
Paths.get(filePart.getSubmittedFileName()).getFileName().toString(); // MSIE
fix.
    InputStream fileContent = filePart.getInputStream();
    // ... (do your job here)
}
```

Note the `Path#getFileName()`. This is a MSIE fix as to obtaining the file name. This browser incorrectly sends the full file path along the name instead of only the file name.

In case you want to upload multiple files via either `multiple="true"`,

```
<input type="file" name="files" multiple="true" />
```

or the old-fashioned way with multiple inputs,

```
<input type="file" name="files" />
<input type="file" name="files" />
<input type="file" name="files" />
...
```

then you can collect them as below (unfortunately there is no such method as `request.getParts("files")`):

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    // ...
    List<Part> fileParts = request.getParts().stream().filter(part ->
"files".equals(part.getName()) && part.getSize() >
0).collect(Collectors.toList()); // Retrieves <input type="file" name="files"
multiple="true">

    for (Part filePart : fileParts) {
        String fileName =
Paths.get(filePart.getSubmittedFileName()).getFileName().toString(); // MSIE
fix.
        InputStream fileContent = filePart.getInputStream();
        // ... (do your job here)
    }
}
```

When you're not on Servlet 3.1 yet, manually get submitted file name

Note that `Part#getSubmittedFileName()` was introduced in Servlet 3.1 (Tomcat 8, Jetty 9, WildFly 8, GlassFish 4, etc, they exist since 2013 already). If you're not on Servlet 3.1 yet (really?), then you need an additional utility method to obtain the submitted file name.

```
private static String getSubmittedFileName(Part part) {
    for (String cd : part.getHeader("content-disposition").split(";")) {
        if (cd.trim().startsWith("filename")) {
            String fileName = cd.substring(cd.indexOf('=') +
1).trim().replace("\"", "");
```

```
            return fileName.substring(fileName.lastIndexOf('/') +
1).substring(fileName.lastIndexOf('\\') + 1); // MSIE fix.
        }
    }
    return null;
}
```

```
String fileName = getSubmittedFileName(filePart);
```

Note the MSIE fix as to obtaining the file name. This browser incorrectly sends the full file path along the name instead of only the file name.

When you're not on Servlet 3.0 yet, use Apache Commons FileUpload

If you're not on Servlet 3.0 yet (isn't it about time to upgrade? it's released over a decade ago!), the common practice is to make use of Apache Commons FileUpload to parse the multpart form data requests. It has an excellent User Guide and FAQ (carefully go through both). There's also the O'Reilly ("cos") `MultipartRequest`, but it has some (minor) bugs and isn't actively maintained anymore for years. I wouldn't recommend using it. Apache Commons FileUpload is still actively maintained and currently very mature.

In order to use Apache Commons FileUpload, you need to have at least the following files in your webapp's `/WEB-INF/lib`:

- `commons-fileupload.jar`
- `commons-io.jar`

Your initial attempt failed most likely because you forgot the commons IO.

Here's a kickoff example how the `doPost()` of your `UploadServlet` may look like when using Apache Commons FileUpload:

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    try {
        List<FileItem> items = new ServletFileUpload(new
DiskFileItemFactory()).parseRequest(request);
        for (FileItem item : items) {
            if (item.isFormField()) {
                // Process regular form field (input
type="text|radio|checkbox|etc", select, etc).
                String fieldName = item.getFieldName();
                String fieldValue = item.getString();
                // ... (do your job here)
            } else {
                // Process form file field (input type="file").
```

```
                String fieldName = item.getFieldName();
                String fileName = FilenameUtils.getName(item.getName());
                InputStream fileContent = item.getInputStream();
                // ... (do your job here)
            }
        }
    } catch (FileUploadException e) {
        throw new ServletException("Cannot parse multipart request.", e);
    }

    // ...
}
```

It's very important that you don't call `getParameter()`, `getParameterMap()`, `getParameterValues()`, `getInputStream()`, `getReader()`, etc on the same request beforehand. Otherwise the servlet container will read and parse the request body and thus Apache Commons FileUpload will get an empty request body. See also a.o. ServletFileUpload#parseRequest(request) returns an empty list.

Note the `FilenameUtils#getName()`. This is a MSIE fix as to obtaining the file name. This browser incorrectly sends the full file path along the name instead of only the file name.

Alternatively you can also wrap this all in a `Filter` which parses it all automagically and put the stuff back in the parametermap of the request so that you can continue using `request.getParameter()` the usual way and retrieve the uploaded file by `request.getAttribute()`. You can find an example in this blog article.

Workaround for GlassFish3 bug of `getParameter()` still returning `null`

Note that Glassfish versions older than 3.1.2 had a bug wherein the `getParameter()` still returns `null`. If you are targeting such a container and can't upgrade it, then you need to extract the value from `getPart()` with help of this utility method:

```
private static String getValue(Part part) throws IOException {
    BufferedReader reader = new BufferedReader(new
InputStreamReader(part.getInputStream(), "UTF-8"));
    StringBuilder value = new StringBuilder();
    char[] buffer = new char[1024];
    for (int length = 0; (length = reader.read(buffer)) > 0;) {
        value.append(buffer, 0, length);
    }
    return value.toString();
}
```

```
String description = getValue(request.getPart("description")); // Retrieves
<input type="text" name="description">
```

Saving uploaded file (don't use `getRealPath()` nor `part.write()`!)

Head to the following answers for detail on properly saving the obtained `InputStream` (the `fileContent` variable as shown in the above code snippets) to disk or database:

- Recommended way to save uploaded files in a servlet application
- How to upload an image and save it in database?
- How to convert Part to Blob, so I can store it in MySQL?

Serving uploaded file

Head to the following answers for detail on properly serving the saved file from disk or database back to the client:

- Load images from outside of webapps / webcontext / deploy folder using <h:graphicImage> or <img> tag
- How to retrieve and display images from a database in a JSP page?
- Simplest way to serve static data from outside the application server in a Java web application
- Abstract template for static resource servlet supporting HTTP caching

Ajaxifying the form

Head to the following answers how to upload using Ajax (and jQuery). Do note that the servlet code to collect the form data does not need to be changed for this! Only the way how you respond may be changed, but this is rather trivial (i.e. instead of forwarding to JSP, just print some JSON or XML or even plain text depending on whatever the script responsible for the Ajax call is expecting).

- How can I upload files to a server using JSP/Servlet and Ajax?
- Send a file as multipart through XMLHttpRequest
- HTML5 drag and drop file upload to Java Servlet

edited Oct 5, 2023 at 16:29

answered Mar 11, 2010 at 12:27

BalusC
1

Ah sorry, I was seeing `request.getParts("file")` and was confused x_x – Kagami Sascha Rosylight Nov 5, 2016 at 22:02

With Servlet 3.0, if a `MultipartConfig` condition is violated (eg: `maxFileSize`), calling `request.getParameter()` returns null. Is this on purpose? What if I get some regular (text) parameters before calling `getPart` (and checking for an `IllegalStateException`)? This causes a `NullPointerException` to be thrown before I have a chance to check for the `IllegalStateException`. – theyuv Nov 20, 2016 at 14:34

@BalusC I created a post related to this, do you have an idea how I could retrieve extra infos from File API webKitDirectory. More details here stackoverflow.com/questions/45419598/… – Rapster Jul 31, 2017 at 23:52

1 Yeah, if someone tries to use the code in 3.0 section with tomcat 7, they might face issue in `String fileName = Paths.get(filePart.getSubmittedFileName()).getFileName().toString(); // MSIE fix.` part similar to me – raviraja Sep 24, 2018 at 13:24

2 @aaa: that can happen when you converted bytes to characters by using a `Reader` and/or `Writer` for unclear reason. Do not do that. Use `InputStream` / `OutputStream` over all place during reading and writing an uploaded file without massaging bytes into characters. A PDF file is not a character based text file or so. It is a binary file. – BalusC Apr 1, 2021 at 10:01

26

If you happen to use Spring MVC, this is how to (I'm leaving this here in case someone find it useful):

Use a form with `enctype` attribute set to "`multipart/form-data`" (the same as BalusC's answer):

```
<form action="upload" method="post" enctype="multipart/form-data">
    <input type="file" name="file" />
    <input type="submit" value="Upload"/>
</form>
```

In your controller, map the request parameter `file` to `MultipartFile` type as follows:

```
@RequestMapping(value = "/upload", method = RequestMethod.POST)
public void handleUpload(@RequestParam("file") MultipartFile file) throws
IOException {
    if (!file.isEmpty()) {
            byte[] bytes = file.getBytes(); // alternatively,
file.getInputStream();
            // application logic
    }
}
```

You can get the filename and size using `MultipartFile`'s `getOriginalFilename()` and `getSize()`.

I've tested this with Spring version `4.1.1.RELEASE`.

Share

Improve this answer

Follow

edited Oct 27, 2021 at 20:44

Peter Mortensen

31.6k • 22 • 109 • 133

answered Aug 26, 2015 at 12:39

Amila

5,213 • 1 • 29 • 47

If I'm not mistaken, this requires that you configure a bean in your server's application config... – Kenny Worden Jul 19, 2018 at 21:36

12

Without components or external libraries in Tomcat 6 or Tomcat 7

Enabling upload in the **web.xml** file:

Manually Installing PHP, Tomcat and Httpd Lounge.

```
<servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
    <multipart-config>
      <max-file-size>3145728</max-file-size>
      <max-request-size>5242880</max-request-size>
    </multipart-config>
    <init-param>
        <param-name>fork</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>xpoweredBy</param-name>
        <param-value>false</param-value>
    </init-param>
    <load-on-startup>3</load-on-startup>
</servlet>
```

As you can see:

```
<multipart-config>
  <max-file-size>3145728</max-file-size>
  <max-request-size>5242880</max-request-size>
</multipart-config>
```

Uploading files using JSP. files:

In the HTML file

```
<form method="post" enctype="multipart/form-data" name="Form" >

  <input type="file" name="fFoto" id="fFoto" value="" /></td>
  <input type="file" name="fResumen" id="fResumen" value=""/>
```

In the JSP File or *Servlet*

```
InputStream isFoto = request.getPart("fFoto").getInputStream();
InputStream isResu = request.getPart("fResumen").getInputStream();
ByteArrayOutputStream baos = new ByteArrayOutputStream();
byte buf[] = new byte[8192];
int qt = 0;
while ((qt = isResu.read(buf)) != -1) {
  baos.write(buf, 0, qt);
}
String sResumen = baos.toString();
```

Edit your code to servlet requirements, like *max-file-size*, *max-request-size* and other options that you can to set...

Share

Improve this answer

Follow

edited Oct 29, 2021 at 11:22

answered Jan 25, 2014 at 5:44

joseluisbz

1,626 ● 2 ● 38 ● 63

11

You need the `common-io.1.4.jar` file to be included in your `lib` directory, or if you're working in any editor, like NetBeans, then you need to go to project properties and just add the JAR file and you will be done.

To get the `common.io.jar` file just google it or just go to the Apache Tomcat website where you get the option for a free download of this file. But remember one thing: download the binary ZIP file if you're a Windows user.

Share

Improve this answer

Follow

edited Nov 3, 2012 at 12:25

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered May 17, 2012 at 11:11

Pranav

237 ● 3 ● 7

Can't find `.jar` but `.zip`. Do you mean `.zip`? – Malwinder Singh Oct 9, 2014 at 11:42

9

I am using a common Servlet for **every** HTML form whether it has attachments or not.

This Servlet returns a `TreeMap` where the keys are JSP name parameters and values are user inputs and saves all attachments in a fixed directory and later you rename the directory of your choice. Here *Connections* is our custom interface having a connection object.

```
public class ServletCommonfunctions extends HttpServlet implements
        Connections {

    private static final long serialVersionUID = 1L;
```

```
    public ServletCommonfunctions() {}

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

    public SortedMap<String, String> savefilesindirectory(
            HttpServletRequest request, HttpServletResponse response)
            throws IOException {

        // Map<String, String> key_values = Collections.synchronizedMap(new
        // TreeMap<String, String>());
        SortedMap<String, String> key_values = new TreeMap<String, String>();
        String dist = null, fact = null;
        PrintWriter out = response.getWriter();
        File file;
        String filePath = "E:\\FSPATH1\\2KL06CS048\\";
        System.out.println("Directory Created   ???????????"
            + new File(filePath).mkdir());
        int maxFileSize = 5000 * 1024;
        int maxMemSize = 5000 * 1024;

        // Verify the content type
        String contentType = request.getContentType();
        if ((contentType.indexOf("multipart/form-data") >= 0)) {
            DiskFileItemFactory factory = new DiskFileItemFactory();
            // Maximum size that will be stored in memory
            factory.setSizeThreshold(maxMemSize);
            // Location to save data that is larger than maxMemSize.
            factory.setRepository(new File(filePath));
            // Create a new file upload handler
            ServletFileUpload upload = new ServletFileUpload(factory);
            // maximum file size to be uploaded.
            upload.setSizeMax(maxFileSize);
            try {
                // Parse the request to get file items.
                @SuppressWarnings("unchecked")
                List<FileItem> fileItems = upload.parseRequest(request);
                // Process the uploaded file items
                Iterator<FileItem> i = fileItems.iterator();
                while (i.hasNext()) {
                    FileItem fi = (FileItem) i.next();
                    if (!fi.isFormField()) {
                        // Get the uploaded file parameters
                        String fileName = fi.getName();
                        // Write the file
                        if (fileName.lastIndexOf("\\") >= 0) {
                            file = new File(filePath
                                + fileName.substring(fileName
                                        .lastIndexOf("\\")));
                        } else {
                            file = new File(filePath
                                + fileName.substring(fileName
                                        .lastIndexOf("\\") + 1));
                        }
                        fi.write(file);
                    } else {
                        key_values.put(fi.getFieldName(), fi.getString());
                    }
                }
```

```
            } catch (Exception ex) {
                System.out.println(ex);
            }
        }
        return key_values;
    }
}
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 20:55

Peter Mortensen

31.6k • 22 • 109 • 133

answered Jan 8, 2013 at 5:50

Nagappa L M

1,480 • 5 • 20 • 33

@buhake sindi hey what should be the filepath if i m using live server or i live my project by uploading files to the server – AmanS Oct 20, 2013 at 4:00

2 Any directory in live server.if you write a code to create a directory in servlet then directory will be created in live srver – Nagappa L M Oct 20, 2013 at 9:27

7

For Spring MVC

I managed to have a simpler version that worked for taking form input, both data and images.

```
<form action="/handleform" method="post" enctype="multipart/form-data">
    <input type="text" name="name" />
    <input type="text" name="age" />
    <input type="file" name="file" />
    <input type="submit" />
</form>
```

Controller to handle

```
@Controller
public class FormController {
    @RequestMapping(value="/handleform",method= RequestMethod.POST)
    ModelAndView register(@RequestParam String name, @RequestParam int age,
@RequestParam MultipartFile file)
            throws ServletException, IOException {

        System.out.println(name);
        System.out.println(age);
        if(!file.isEmpty()){
            byte[] bytes = file.getBytes();
            String filename = file.getOriginalFilename();
            BufferedOutputStream stream =new BufferedOutputStream(new
FileOutputStream(new File("D:/" + filename)));
            stream.write(bytes);
            stream.flush();
            stream.close();
        }
        return new ModelAndView("index");
```

```
    }
}
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 21:09

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jul 15, 2017 at 19:42

Shivangi Gupta

866 ● 8 ● 21

Can you please share select image form db mysql and show it on jsp/html? – Onic Team Mar 13, 2019 at 12:27

6

Another source of this problem occurs if you are using Geronimo with its embedded Tomcat. In this case, after many iterations of testing Commons IO and commons-fileupload, the problem arises from a parent classloader handling the commons-xxx JAR files. This has to be prevented. The crash always occurred at:

```
fileItems = uploader.parseRequest(request);
```

Note that the List type of fileItems has changed with the current version of commons-fileupload to be specifically `List<FileItem>` as opposed to prior versions where it was generic `List`.

I added the source code for commons-fileupload and Commons IO into my Eclipse project to trace the actual error and finally got some insight. First, the exception thrown is of type Throwable not the stated FileIOException nor even Exception (these will not be trapped). Second, the error message is obfuscatory in that it stated class not found because axis2 could not find commons-io. Axis2 is not used in my project at all, but it exists as a folder in the Geronimo repository subdirectory as part of standard installation.

Finally, I found one place that posed a working solution which successfully solved my problem. You must hide the JAR files from the parent loader in the deployment plan. This was put into the *geronimo-web.xml* file with my full file shown below.

Pasted from *http://osdir.com/ml/user-geronimo-apache/2011-03/msg00026.html*:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<web:web-app xmlns:app="http://geronimo.apache.org/xml/ns/j2ee/application-2.0"
xmlns:client="http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0"
xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"
xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"
xmlns:ejb="http://openejb.apache.org/xml/ns/openejb-jar-2.2"
xmlns:log="http://geronimo.apache.org/xml/ns/loginconfig-2.0"
xmlns:name="http://geronimo.apache.org/xml/ns/naming-1.2"
xmlns:pers="http://java.sun.com/xml/ns/persistence"
xmlns:pkgen="http://openejb.apache.org/xml/ns/pkgen-2.1"
xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0"
```

```
xmlns:web="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1">
    <dep:environment>
        <dep:moduleId>
            <dep:groupId>DataStar</dep:groupId>
            <dep:artifactId>DataStar</dep:artifactId>
            <dep:version>1.0</dep:version>
            <dep:type>car</dep:type>
        </dep:moduleId>

        <!-- Don't load commons-io or fileupload from parent classloaders -->
        <dep:hidden-classes>
            <dep:filter>org.apache.commons.io</dep:filter>
            <dep:filter>org.apache.commons.fileupload</dep:filter>
        </dep:hidden-classes>
        <dep:inverse-classloading/>

    </dep:environment>
    <web:context-root>/DataStar</web:context-root>
</web:web-app>
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 20:15

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 10, 2013 at 15:15

Geoffrey Malafsky

164 ● 2 ● 5

The link is (effectively) broken (redirects to `https://osdir.com/`) - the HTTPS version does as well. – Peter Mortensen Oct 27, 2021 at 20:17

0

Here's an example using apache commons-fileupload:

```
// apache commons-fileupload to handle file upload
DiskFileItemFactory factory = new DiskFileItemFactory();
factory.setRepository(new File(DataSources.TORRENTS_DIR()));
ServletFileUpload fileUpload = new ServletFileUpload(factory);

List<FileItem> items = fileUpload.parseRequest(req.raw());
FileItem item = items.stream()
  .filter(e ->
  "the_upload_name".equals(e.getFieldName()))
  .findFirst().get();
String fileName = item.getName();

item.write(new File(dir, fileName));
log.info(fileName);
```

Share Improve this answer Follow

answered May 21, 2015 at 16:49

dessalines

7,235 ● 5 ● 51 ● 66

You first have to set the enctype attribute of the form to "multipart/form-data"

0

This is shown below.

```
<form action="Controller" method="post" enctype="multipart/form-data">
     <label class="file-upload"> Click here to upload an Image </label>
     <input type="file" name="file" id="file" required>
</form>
```

And then, in the Servlet "Controller" add the Annotation for a Multi-part to indicate multipart data is processed in the servlet.

After doing this, retrieve the part sent through the form and then retrieve the file name (with path)of the submitted file. Use this to create a new file in the desired path and write the parts of the file to the newly created file to recreate the file.

As shown below:

```
@MultipartConfig

public class Controller extends HttpServlet {

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

    private void addProduct(HttpServletRequest request, HttpServletResponse
response) {
        Part filePart = request.getPart("file");
        String imageName =
Paths.get(filePart.getSubmittedFileName()).getFileName().toString();

        String imageSavePath = "specify image path to save image"; //path to
save image
        FileOutputStream outputStream = null;
        InputStream fileContent = null;

        try {
            outputStream = new FileOutputStream(new File(imageSavePath +
File.separator + imageName));
            // Creating a new file with file path and the file name
            fileContent = filePart.getInputStream();
            // Getting the input stream
            int readBytes = 0;
            byte[] readArray = new byte[1024];
            // Initializing a byte array with size 1024

            while ((readBytes = fileContent.read(readArray)) != -1) {
                outputStream.write(readArray, 0, readBytes);
            } // This loop will write the contents of the byte array unitl the
end to the output stream
        } catch (Exception ex) {
            System.out.println("Error Writing File: " + ex);
        } finally {
            if (outputStream != null) {
                outputStream.close();
```

```
                // Closing the output stream
            }
            if (fileContent != null) {
                fileContent.close();
                // Closing the input stream
            }
        }
    }
}
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 21:16

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Nov 24, 2020 at 10:49

Lakindu Hewawasam

750 ● 6 ● 11

1 This solution is different. Other solution used a library to handle files where as this does tf his with no 3rd party jar files. – Lakindu Hewawasam Nov 25, 2020 at 0:43

1 This is already covered by the currently accepted answer. Have you read it? The native API already existed since December 2009. Your way of closing the streams is by the way also legacy. Since Java 7 which was introduced in July 2011, you can use try-with-resources statement instead of fiddling with nullchecks in finally. – BalusC Nov 25, 2020 at 8:57

The simplest way I could come up with for files and input controls, without a billion libraries:

-1

```
  <%
      if (request.getContentType() == null)
          return;
      // For input type=text controls
      String v_Text =
          (new BufferedReader(new
InputStreamReader(request.getPart("Text1").getInputStream()))).readLine();

      // For input type=file controls
      InputStream inStr = request.getPart("File1").getInputStream();
      char charArray[] = new char[inStr.available()];
      new InputStreamReader(inStr).read(charArray);
      String contents = new String(charArray);
  %>
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 19:58

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Apr 26, 2020 at 5:36

access_granted

1,897 ● 22 ● 28

What is `<%` for? ASP.NET (C#)? Can you clarify? Please respond by editing (changing) your answer, not here in comments (***without*** "Edit:", "Update:", or similar - the answer should appear as if it was written today). – Peter Mortensen Oct 27, 2021 at 21:10

-1

Use:

```
DiskFileUpload upload = new DiskFileUpload();
```

From this object you have to get the file items and fields, and then you can store into the server like the following:

```
String loc = "./webapps/prjct name/server folder/" + contentid + extension;
File uploadFile = new File(loc);
item.write(uploadFile);
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 20:36

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Mar 23, 2015 at 12:37

Mahender Reddy Yasa

171 ● 7 ● 20

-1

You can upload a file using JSP /servlet.

```
<form action="UploadFileServlet" method="post">
    <input type="text" name="description" />
    <input type="file" name="file" />
    <input type="submit" />
</form>
```

On the other hand, on the server side, use the following code.

```
package com.abc..servlet;

import java.io.File;
---------
--------


/**
 * Servlet implementation class UploadFileServlet
 */
public class UploadFileServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public UploadFileServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.sendRedirect("../jsp/ErrorPage.jsp");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
```

```
response) throws ServletException, IOException {
        // TODO Auto-generated method stub

        PrintWriter out = response.getWriter();
        HttpSession httpSession = request.getSession();
        String filePathUpload = (String) httpSession.getAttribute("path") !=
null ? httpSession.getAttribute("path").toString() : "" ;

        String path1 = filePathUpload;
        String filename = null;
        File path = null;
        FileItem item = null;


        boolean isMultipart = ServletFileUpload.isMultipartContent(request);

        if (isMultipart) {
            FileItemFactory factory = new DiskFileItemFactory();
            ServletFileUpload upload = new ServletFileUpload(factory);
            String FieldName = "";
            try {
                List items = upload.parseRequest(request);
                Iterator iterator = items.iterator();
                while (iterator.hasNext()) {
                     item = (FileItem) iterator.next();

                        if (fieldname.equals("description")) {
                            description = item.getString();
                        }
                    }
                    if (!item.isFormField()) {
                        filename = item.getName();
                        path = new File(path1 + File.separator);
                        if (!path.exists()) {
                            boolean status = path.mkdirs();
                        }
                        /* Start of code fro privilege */

                        File uploadedFile = new File(path + Filename);  // for
copy file
                        item.write(uploadedFile);
                        }
                    } else {
                        f1 = item.getName();
                    }

                } // END OF WHILE
                response.sendRedirect("welcome.jsp");
            } catch (FileUploadException e) {
                e.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

Share

Improve this answer

edited Oct 27, 2021 at 20:42

Peter Mortensen

answered Feb 4, 2014 at 10:00

Mitul Maheshwari

What do you mean by *"Start of code fro privilege"* (seems incomprehensible)? Please respond by editing (changing) your answer, not here in comments (***without*** "Edit:", "Update:", or similar - the answer should appear as if it was written today). – Peter Mortensen Oct 27, 2021 at 20:43

-2

HTML page

```
<html>
    <head>
        <title>File Uploading Form</title>
    </head>

    <body>
        <h3>File Upload:</h3>
        Select a file to upload: <br />
        <form action="UploadServlet" method="post"
              enctype="multipart/form-data">

            <input type="file" name="file" size="50" />
            <br />
            <input type="submit" value="Upload File" />
        </form>
    </body>
</html>
```

Servlet file

```
// Import required java libraries
import java.io.*;
import java.util.*;

import javax.servlet.ServletConfig;
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.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.apache.commons.io.output.*;

public class UploadServlet extends HttpServlet {

    private boolean isMultipart;
    private String filePath;
    private int maxFileSize = 50 * 1024;
    private int maxMemSize = 4 * 1024;
    private File file;

    public void init() {
```

```
        // Get the file location where it would be stored.
        filePath =
               getServletContext().getInitParameter("file-upload");
    }

    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
               throws ServletException, java.io.IOException {

        // Check that we have a file upload request
        isMultipart = ServletFileUpload.isMultipartContent(request);
        response.setContentType("text/html");
        java.io.PrintWriter out = response.getWriter();
        if (!isMultipart) {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet upload</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<p>No file uploaded</p>");
            out.println("</body>");
            out.println("</html>");
            return;
        }

        DiskFileItemFactory factory = new DiskFileItemFactory();
        // Maximum size that will be stored in memory
        factory.setSizeThreshold(maxMemSize);
        // Location to save data that is larger than maxMemSize.
        factory.setRepository(new File("c:\\temp"));

        // Create a new file upload handler
        ServletFileUpload upload = new ServletFileUpload(factory);
        // maximum file size to be uploaded.
        upload.setSizeMax(maxFileSize);

        try {
            // Parse the request to get file items.
            List fileItems = upload.parseRequest(request);

            // Process the uploaded file items
            Iterator i = fileItems.iterator();

            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet upload</title>");
            out.println("</head>");
            out.println("<body>");
            while (i.hasNext())
            {
                FileItem fi = (FileItem)i.next();
                if (!fi.isFormField())
                {
                     // Get the uploaded file parameters
                     String fieldName = fi.getFieldName();
                     String fileName = fi.getName();
                     String contentType = fi.getContentType();
                     boolean isInMemory = fi.isInMemory();
                     long sizeInBytes = fi.getSize();

                     // Write the file
                     if (fileName.lastIndexOf("\\") >= 0) {
```

```
                        file = new File(filePath +
                        fileName.substring(fileName.lastIndexOf("\\")));
                    }
                    else {
                        file = new File(filePath +
                        fileName.substring(fileName.lastIndexOf("\\") + 1));
                    }
                    fi.write(file);
                    out.println("Uploaded Filename: " + fileName + "<br>");
                }
            }
            out.println("</body>");
            out.println("</html>");
        }
        catch(Exception ex) {
            System.out.println(ex);
        }
    }

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
            throws ServletException, java.io.IOException {

        throw new ServletException("GET method used with " +
                getClass().getName() + ": POST method required.");
    }
}
```

File *web.xml*

Compile the above servlet UploadServlet and create the required entry in the *web.xml* file as follows.

```
<servlet>
   <servlet-name>UploadServlet</servlet-name>
   <servlet-class>UploadServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>UploadServlet</servlet-name>
   <url-pattern>/UploadServlet</url-pattern>
</servlet-mapping>
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 20:58

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jul 15, 2016 at 7:14

Himanshu Patel

243 ● 2 ● 16

-4

Sending multiple files for file, we have to use `enctype="multipart/form-data"`.

And to send multiple files, use `multiple="multiple"` in the *input* tag:

```
<form action="upload" method="post" enctype="multipart/form-data">
    <input type="file" name="fileattachments"  multiple="multiple"/>
    <input type="submit" />
</form>
```

Share

Improve this answer

Follow

edited Oct 27, 2021 at 20:00

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Oct 24, 2013 at 9:35

rohan kamat

581 ● 5 ● 12

2 How would we go about doing getPart("fileattachments") so we get an array of Parts instead? I don't think getPart for multiple files will work? – CyberMew Sep 29, 2015 at 6:38

What do you mean by *"Sending multiple files for file"* (seems incomprehensible)? Please respond by editing (changing) your answer, not here in comments (***without*** "Edit:", "Update:", or similar - the question/answer should appear as if it was written today). – Peter Mortensen Oct 27, 2021 at 20:00

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.