How to configure and troubleshoot <p:fileUpload> depends on PrimeFaces version.

All PrimeFaces versions

The below requirements apply to all PrimeFaces versions:

1. The enctype attribute of the <h:form> needs to be set to multipart/form-data. When this is absent, the ajax upload may just work, but the general browser behavior is unspecified and dependent on form composition and webbrowser make/version. Just always specify it to be on the safe side.
2. When using mode="advanced" (i.e. ajax upload, this is the default), then make sure that you've a <h:head> in the (master) template. This will ensure that the necessary JavaScript files are properly included. This is not required for mode="simple" (non-ajax upload), but this would break look'n'feel and functionality of all other PrimeFaces components, so you don't want to miss that anyway.
3. When using mode="simple" (i.e. non-ajax upload), then ajax must be disabled on any PrimeFaces command buttons/links by ajax="false", and you must use <p:fileUpload value> with <p:commandButton action> instead of <p:fileUpload fileUploadListener>.

So, if you want (auto) file upload with ajax support (mind the <h:head>!):

<h:form enctype="multipart/form-data">

<p:fileUpload fileUploadListener="#{bean.upload}" auto="true" />

</h:form>

public void upload(FileUploadEvent event) {

UploadedFile uploadedFile = event.getUploadedFile();

String fileName = uploadedFile.getFileName();

String contentType = uploadedFile.getContentType();

byte[] contents = uploadedFile.getContents();

// ...

}

Or if you want non-ajax file upload:

<h:form enctype="multipart/form-data">

<p:fileUpload mode="simple" value="#{bean.uploadedFile}" />

<p:commandButton value="Upload" action="#{bean.upload}" ajax="false" />

</h:form>

private UploadedFile uploadedFile; // +getter+setter

public void upload() {

String fileName = uploadedFile.getFileName();

String contentType = uploadedFile.getContentType();

byte[] contents = uploadedFile.getContents();

// ...

}

Do note that ajax-related attributes such as auto, allowTypes, update, onstart, oncomplete, etc are **ignored** in mode="simple". So it's needless to specify them in such case.

PrimeFaces 5.x

This does not require any additional configuration if you're using JSF 2.2 and your faces-config.xmlis also declared conform JSF 2.2 version. You do not need the PrimeFaces file upload filter at all.

If you're however not using JSF 2.2 yet and you can't upgrade it (should be effortless when already on a Servlet 3.0 compatible container), then you need to manually register the below PrimeFaces file upload filter in web.xml (it will parse the multi part request and fill the regular request parameter map so that FacesServlet can continue working as usual):

<filter>

<filter-name>PrimeFaces FileUpload Filter</filter-name>

<filter-class>org.primefaces.webapp.filter.FileUploadFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>PrimeFaces FileUpload Filter</filter-name>

<servlet-name>facesServlet</servlet-name>

</filter-mapping>

The <servlet-name> value of facesServlet must match exactly the value in the <servlet> entry of the javax.faces.webapp.FacesServlet in the same web.xml. So if it's e.g. Faces Servlet, then you need to edit it accordingly to match.

PrimeFaces 4.x

The same story as PrimeFaces 5.x applies on 4.x as well.

There's only a potential problem in getting the uploaded file content by UploadedFile#getContents(). This will return null when native API is used instead of Apache Commons FileUpload. You need to use UploadedFile#getInputStream() instead. See also [How to insert uploaded image from p:fileUpload as BLOB in MySQL?](http://stackoverflow.com/questions/8305633/how-to-insert-uploaded-image-from-pfileupload-as-blob-in-mysql/8310773#8310773)

Another potential problem with native API will manifest is when the upload component is present in a form on which a different "regular" ajax request is fired which does not process the upload component. See also [File upload doesn't work with AJAX in PrimeFaces 4.0/JSF 2.2.x - javax.servlet.ServletException: The request content-type is not a multipart/form-data](http://stackoverflow.com/questions/19262356/file-upload-doesnt-work-with-ajax-in-primefaces-4-0-jsf-2-2-x-javax-servlet-s).

Both problems can also be solved by switching to Apache Commons FileUpload. See PrimeFaces 3.x section for detail.

PrimeFaces 3.x

This version does not support JSF 2.2 / Servlet 3.0 native file upload. You need to manually install Apache Commons FileUpload and explicitly register the file upload filter in web.xml.

You need the following libraries:

* [commons-fileupload.jar](http://commons.apache.org/fileupload)
* [commons-io.jar](http://commons.apache.org/io)

Those must be present in the webapp's runtime classpath. When using Maven, make sure they are at least runtime scoped (default scope of compile is also good). When manually carrying around JARs, make sure they end up in /WEB-INF/lib folder.

The file upload filter registration detail can be found in PrimeFaces 5.x section here above. In case you're using PrimeFaces 4+ and you'd like to explicitly use Apache Commons FileUpload instead of JSF 2.2 / Servlet 3.0 native file upload, then you need next to the mentioned libraries and filter also the below context param in web.xml:

<context-param>

<param-name>primefaces.UPLOADER</param-name>

<param-value>commons</param-value><!-- Allowed values: auto, native and commons. -->

</context-param>

Troubleshooting

In case it still doesn't work, here are another possible causes unrelated to PrimeFaces configuration:

1. Only if you're using the PrimeFaces file upload filter: There's another Filter in your webapp which runs *before* the PrimeFaces file upload filter and has already consumed the request body by e.g. calling getParameter(), getParameterMap(), getReader(), etcetera. A request body can be parsed only once. When you call one of those methods before the file upload filter does its job, then the file upload filter will get an empty request body.

To fix this, you'd need to put the <filter-mapping> of the file upload filter *before* the other filter inweb.xml. If the request is not a multipart/form-data request, then the file upload filter will just continue as if nothing happened.

1. Only if you're using the PrimeFaces file upload filter: There's another Filter in your webapp which runs *before* the PrimeFaces file upload filter and has performed a [RequestDispatcher#forward()](http://docs.oracle.com/javaee/6/api/javax/servlet/RequestDispatcher.html" \l "forward%28javax.servlet.ServletRequest,%20javax.servlet.ServletResponse%29) call. Usually, URL rewrite filters such as [PrettyFaces](http://ocpsoft.org/prettyfaces/) do this. This triggers the FORWARD dispatcher, but filters listen by default on REQUEST dispatcher only.

To fix this, you'd need to either put the PrimeFaces file upload filter *before* the forwarding filter, or to reconfigure the PrimeFaces file upload filter to listen on FORWARD dispatcher:

<filter-mapping>

<filter-name>PrimeFaces FileUpload Filter</filter-name>

<servlet-name>facesServlet</servlet-name>

<dispatcher>FORWARD</dispatcher>

</filter-mapping>

*(note: you can specify multiple <dispatcher> entries if you need to, so having both REQUESTand FORWARD in one mapping is also legitimately valid)*

1. There's a nested <h:form>. This is illegal in HTML and the browser behavior is unspecified. More than often, the browser won't send the expected data on submit. Make sure that you are not nesting <h:form>. This is completely regardless of the form's enctype. Just do not nest forms at all.

If you're still having problems, well, debug the HTTP traffic. Open the webbrowser's developer toolset (press F12 in Chrome/Firebug23+/IE9+) and check the Net/Network section. If the HTTP part looks fine, then debug the JSF code. Put a breakpoint on [FileUploadRenderer#decode()](https://code.google.com/p/primefaces/source/browse/primefaces/trunk/src/main/java/org/primefaces/component/fileupload/FileUploadRenderer.java) and advance from there.