# JAXRS- File Upload:

These are different Form content types defined by W3C. If you want to send simple text/ ASCII data, then x-www-form-urlencoded will work. This is the default.

But if you have to send non-ASCII text or large binary data, the form-data is for that.

Refer to this link for further reading: [Forms in HTML documents](http://www.w3.org/TR/html401/interact/forms.html#h-17.13.4.1)

### Form content types

The [enctype](https://www.w3.org/TR/html401/interact/forms.html" \l "adef-enctype) attribute of the [FORM](https://www.w3.org/TR/html401/interact/forms.html#edef-FORM) element specifies the [content type](https://www.w3.org/TR/html401/types.html#type-content-type) used to encode the [form data set](https://www.w3.org/TR/html401/interact/forms.html#form-data-set) for submission to the server. User agents must support the content types listed below. Behavior for other content types is unspecified.

Please also consult the section on [escaping ampersands in URI attribute values](https://www.w3.org/TR/html401/appendix/notes.html#ampersands-in-uris).

#### application/x-www-form-urlencoded

This is the default content type. Forms submitted with this content type must be encoded as follows:

1. Control names and values are escaped. Space characters are replaced by `+', and then reserved characters are escaped as described in [[RFC1738]](https://www.w3.org/TR/html401/references.html#ref-RFC1738), section 2.2: Non-alphanumeric characters are replaced by `%HH', a percent sign and two hexadecimal digits representing the ASCII code of the character. Line breaks are represented as "CR LF" pairs (i.e., `%0D%0A').
2. The control names/values are listed in the order they appear in the document. The name is separated from the value by `=' and name/value pairs are separated from each other by `&'.

#### multipart/form-data

***Note.***Please consult[*[RFC2388]*](https://www.w3.org/TR/html401/references.html#ref-RFC2388)for additional information about file uploads, including backwards compatibility issues, the relationship between "multipart/form-data" and other content types, performance issues, etc.

Please consult the appendix for information about[*security issues for forms*](https://www.w3.org/TR/html401/appendix/notes.html#forms-security).

The content type "application/x-www-form-urlencoded" is inefficient for sending large quantities of binary data or text containing non-ASCII characters. The content type "multipart/form-data" should be used for submitting forms that contain files, non-ASCII data, and binary data.

The content "multipart/form-data" follows the rules of all multipart MIME data streams as outlined in [[RFC2045]](https://www.w3.org/TR/html401/references.html#ref-RFC2045). The definition of "multipart/form-data" is available at the [[IANA]](https://www.w3.org/TR/html401/references.html#ref-IANA) registry.

A "multipart/form-data" message contains a series of parts, each representing a [successful control](https://www.w3.org/TR/html401/interact/forms.html#successful-controls). The parts are sent to the processing agent in the same order the corresponding controls appear in the document stream. Part boundaries should not occur in any of the data; how this is done lies outside the scope of this specification.

As with all multipart MIME types, each part has an optional "Content-Type" header that defaults to "text/plain". User agents should supply the "Content-Type" header, accompanied by a "charset" parameter.

Each part is expected to contain:

1. a "Content-Disposition" header whose value is "form-data".
2. a name attribute specifying the [control name](https://www.w3.org/TR/html401/interact/forms.html#control-name) of the corresponding control. Control names originally encoded in non-ASCII [character sets](https://www.w3.org/TR/html401/charset.html) may be encoded using the method outlined in [[RFC2045]](https://www.w3.org/TR/html401/references.html#ref-RFC2045).

Thus, for example, for a control named "mycontrol", the corresponding part would be specified:

Content-Disposition: form-data; name="mycontrol"

As with all MIME transmissions, "CR LF" (i.e., `%0D%0A') is used to separate lines of data.

Each part may be encoded and the "Content-Transfer-Encoding" header supplied if the value of that part does not conform to the default (7BIT) encoding (see [[RFC2045]](https://www.w3.org/TR/html401/references.html#ref-RFC2045), section 6)

If the contents of a file are submitted with a form, the file input should be identified by the appropriate [content type](https://www.w3.org/TR/html401/types.html#type-content-type) (e.g., "application/octet-stream"). If multiple files are to be returned as the result of a single form entry, they should be returned as "multipart/mixed" embedded within the "multipart/form-data".

The user agent should attempt to supply a file name for each submitted file. The file name may be specified with the "filename" parameter of the 'Content-Disposition: form-data' header, or, in the case of multiple files, in a 'Content-Disposition: file' header of the subpart. If the file name of the client's operating system is not in US-ASCII, the file name might be approximated or encoded using the method of [[RFC2045]](https://www.w3.org/TR/html401/references.html#ref-RFC2045). This is convenient for those cases where, for example, the uploaded files might contain references to each other (e.g., a TeX file and its ".sty" auxiliary style description).

The following example illustrates "multipart/form-data" encoding. Suppose we have the following form:

<FORM action="http://server.com/cgi/handle"

enctype="multipart/form-data"

method="post">

<P>

What is your name? <INPUT type="text" name="submit-name"><BR>

What files are you sending? <INPUT type="file" name="files"><BR>

<INPUT type="submit" value="Send"> <INPUT type="reset">

</FORM>

If the user enters "Larry" in the text input, and selects the text file "file1.txt", the user agent might send back the following data:

Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x

Content-Disposition: form-data; name="submit-name"

Larry

--AaB03x

Content-Disposition: form-data; name="files"; filename="file1.txt"

Content-Type: text/plain

... contents of file1.txt ...

--AaB03x--

If the user selected a second (image) file "file2.gif", the user agent might construct the parts as follows:

Content-Type: multipart/form-data; boundary=AaB03x

--AaB03x

Content-Disposition: form-data; name="submit-name"

Larry

--AaB03x

Content-Disposition: form-data; name="files"

Content-Type: multipart/mixed; boundary=BbC04y

--BbC04y

Content-Disposition: file; filename="file1.txt"

Content-Type: text/plain

... contents of file1.txt ...

--BbC04y

Content-Disposition: file; filename="file2.gif"

Content-Type: image/gif

Content-Transfer-Encoding: binary

...contents of file2.gif...

--BbC04y--

--AaB03x--

The "multipart/mixed" (and other similar types) media type was explicitly designed to formalize the transport of related collections of body entities in the same message ... the classic use case being email attachments. These media types were explicitly designed to be more general, so they do not impose any use-case-specific restrictions on things like requiring certain headers with certain parameters. The definitions are in RFC 2046 through 2049.