

# Java Servlets 3.0

## Lesson 9: Multipart File Upload

## Lesson Objective

- In this lesson, we will learn:
  - Introduction to Multipart File Upload
  - Use of Multipart Config in Servlets



Copyright © Capgemini 2015. All Rights Reserved

2

### Lesson Objectives:

This lesson introduces Multipart File Upload. The lesson contents are:

#### Lesson 09: Multipart File Upload

9.1: Introduction to Multipart File Upload

9.2: Use of Multipart Config in Servlets

9.1: Introduction to Multipart File Upload

## Multipart File Upload

- File upload is a common requirement for web applications
- Prior to Servlet 3.0, implementing File upload required use of external libraries and complex input processing
- With the advent of Servlet 3.0 , file(s) could be uploaded
- Make use of following annotation and methods
  - @Multipart Config
  - getParts() and getPart() method



Copyright © Capgemini 2015. All Rights Reserved

3

Multipart File Upload:

File uploading is now a common requirement for web applications.

It has now become easier since Servlet 3.0

9.2: Use of Multipart Config in Servlets

## Multi Part Config

- The basic annotation used during File Upload is `@MultipartConfig`
  - Describes where the uploaded file will get stored
  - Describes maximum size allowed for upload
  - Specifies maximum size allowed for multipart / form-data requested
- For example, the `@MultipartConfig` annotation could be constructed as follows:
  - `@MultipartConfig(location="/tmp", fileSizeThreshold=1024*1024, maxFileSize=1024*1024*5, maxRequestSize=1024*1024*5*5)`



Copyright © Capgemini 2015. All Rights Reserved

4

The `@MultipartConfig` annotation supports the following optional attributes:

**location:** An absolute path to a directory on the file system. The location attribute does not support a path relative to the application context. This location is used to store files temporarily while the parts are processed or when the size of the file exceeds the specified `fileSizeThreshold` setting. The default location is `""`.

**fileSizeThreshold:** The file size in bytes after which the file will be temporarily stored on disk. The default size is 0 bytes.

**MaxFileSize:** The maximum size allowed for uploaded files, in bytes. If the size of any uploaded file is greater than this size, the web container will throw an exception (`IllegalStateException`). The default size is unlimited.

**maxRequestSize:** The maximum size allowed for a multipart/form-data request, in bytes. The web container will throw an exception if the overall size of all uploaded files exceeds this threshold. The default size is unlimited.

9.2: Use of Multipart Config in Servlets

## Part Interface

- Part interface represents a part or form item that was received within a multipart/form-data POST request
- Provides getParts() and getPart() Methods
- Servlet 3.0 supports two additional `HttpServletRequest` methods:
  - `Collection<Part> getParts()`
  - `Part getPart(String name)`



Copyright © Capgemini 2015. All Rights Reserved

5

Part interface is now a part of `javax.servlet.http` package, and used to represent data received within a multipart / form – data.

Some important methods are `getInputStream()`, `write(String fileName)` that we can use to read and write file.

The `request.getParts()` method returns collections of all Part objects. If there are more than one input of type file, multiple Part objects are returned. Since Part objects are named, the `getPart(String name)` method can be used to access a particular Part. Alternatively, the `getParts()` method, which returns an `Iterable<Part>`, can be used to get an Iterator over all the Part objects.

9.2: Use of Multipart Config in Servlets

## Part Interface

- The `javax.servlet.http.Part` interface is a simple one, providing methods that allow introspection of each Part. The methods do the following:
  - Retrieve the name, size, and content-type of the Part
  - Query the headers submitted with a Part
  - Delete a Part
  - Query the headers submitted with a Part
  - Write a Part out to disk




Copyright © Capgemini 2015. All Rights Reserved 6

For example, the Part interface provides the `write(String filename)` method to write the file with the specified name. The file can then be saved in the directory specified with the location attribute of the `@MultipartConfig` annotation or, in the case of the file-upload example, in the location specified by the Destination field in the form.

9.2: Use of Multipart Config in Servlets


## File Upload Application

- Application illustrates how to implement and use the file upload feature
- Refer: FileUploadServlet.java



**Total parts : 1**

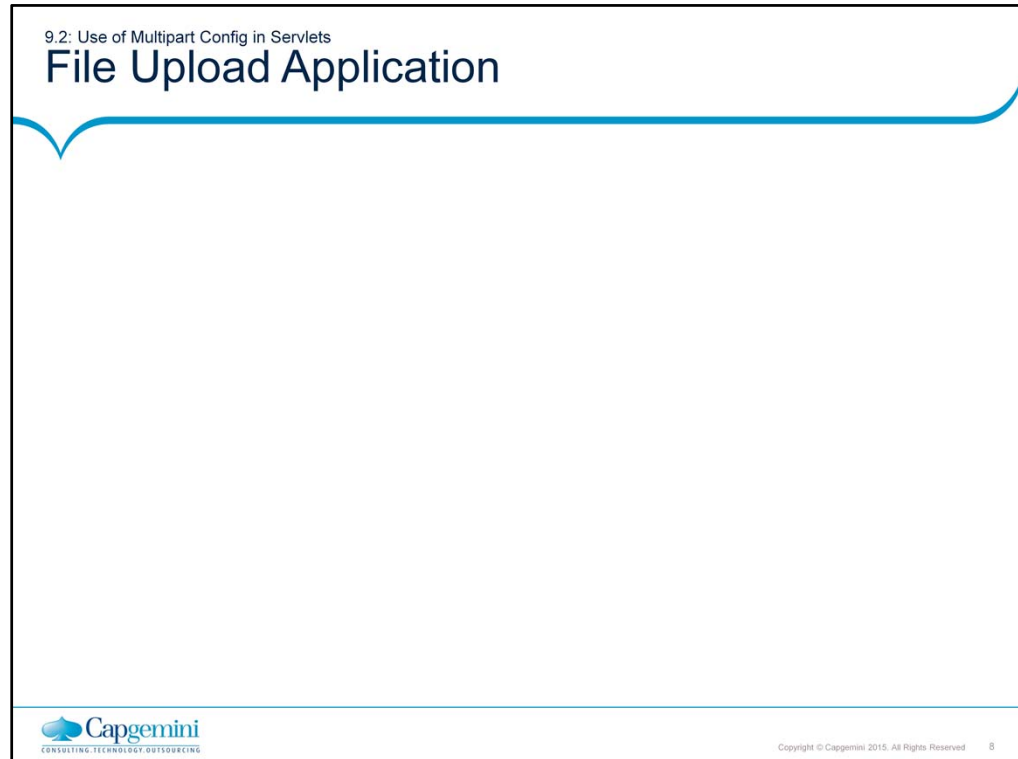
Name : file1  
 Content Type : application/vnd.openxmlformats-officedocument.wordprocessingml.document  
 Size : 4097205  
 Content-Disposition : form-data; name="file1"; filename="D:\Yukti Data\Yukti\Java\gate\Advanced Java-Module 3\Servlets - 3.0-Upgrade\Connection Pooling with WildFly 8.docx"  
 Content-Type : application/vnd.openxmlformats-officedocument.wordprocessingml.document

 **Capgemini**  
CONSULTING TECHNOLOGY OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved 7

The input type as file, enables a user to browse the local file system to select the file. When the file is selected, it is sent to the server as a part of a POST request. During this process two mandatory restrictions are applied to the form with input type file: The enctype attribute must be set to a value of multipart/form-data. Its method must be POST.

When the form is specified in this manner, the entire request is sent to the server in encoded form. The servlet then handles the request to process the incoming file data and to extract a file from the stream. The destination is the path to the location where the file will be saved on your computer. Pressing the Upload button at the bottom of the form posts the data to the servlet, which saves the file in the specified destination.



A POST request method is used when the client needs to send data to the server as part of the request, such as when uploading a file or submitting a completed form. In contrast, a GET request method sends a URL and headers only to the server, whereas POST requests also include a message body. This allows arbitrary-length data of any type to be sent to the server. A header field in the POST request usually indicates the message body's Internet media type.

When submitting a form, the browser streams the content in, combining all parts, with each part representing a field of a form. Parts are named after the input elements and are separated from each other with string delimiters named boundary.

This is what submitted data from the file-upload form looks like, after selecting sample.txt as the file that will be uploaded to the tmp-directory on the local file system:



## Summary

- In this lesson, we have learnt:
  - Introduction to Multipart File Upload
  - Use of Multipart Config in Servlets



Add the notes here.

## Review Question

- Question 1: Which of the following attribute can be used to specify maximum size allowed for uploaded files, in bytes?
  - Option 1: fileSizeThreshold
  - Option 2: MaxFileSize
  - Option 3: maxRequestSize
  - Option 4: locationsize



Add the notes here.

## Review Question

- Question 2: Where is the file stored after uploading?
  - Option 1: Stored in current working directory
  - Option 2: Directory as that of where server is installed
  - Option 3: Path specified given in location attribute of @MultipartConfig annotation
  - Option 4: Can specify any arbitrary location on server



Add the notes here.