DOWNLOADING & UPLOADING IMAGES

In this chapter we are going to see how you can download an image from internet, perform some image processing techniques on the image, and then again upload the processed image to a server.

Downloading an Image

In order to download an image from a website, we use java class named **URL**, which can be found under **java.net** package. Its syntax is given below:

```
String website = "http://tutorialspoint.com";
URL url = new URL(website);
```

Apart from the above method , there are other methods available in URL class which are listed below:

Sr.No.	Methods
1	**public String getPath** It returns the path of the URL.
2	**public String getQuery** It returns the query part of the URL.
3	**public String getAuthority** It returns the authority of the URL.
4	**public int getPort** It returns the port of the URL.
5	**public int getDefaultPort** It returns the default port for the protocol of the URL.
6	**public String getProtocol** It returns the protocol of the URL.
7	**public String getHost** It returns the host of the URL.

Example

The following example demonstrates the use of java URL class to download an image from the internet:

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import java.net.URL;

public class Download {

   public static void main(String[] args) throws Exception {

      try{
         String fileName = "digital_image_processing.jpg";
         String website = "http://tutorialspoint.com/java_dip/images/"+fileName;

         System.out.println("Downloading File From: " + website);

         URL url = new URL(website);
         InputStream inputStream = url.openStream();
         OutputStream outputStream = new FileOutputStream(fileName);
         byte[] buffer = new byte[2048];

         int length = 0;

         while ((length = inputStream.read(buffer)) != -1) {
            System.out.println("Buffer Read of length: " + length);
            outputStream.write(buffer, 0, length);
         }

         inputStream.close();
         outputStream.close();

      }catch(Exception e){
         System.out.println("Exception: " + e.getMessage());
      }
   }
}
```

Output

When you execute the given above, the following output is seen.

```
Downloading File From: http://tutorialspoint.com/java_dip/images/digital_image_p
rocessing.jpg
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 1369
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 960
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 416
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 960
Buffer Read of length: 2048
Buffer Read of length: 752
Buffer Read of length: 2048
```

```
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 2048
Buffer Read of length: 187
```

It would download the following image from the server.

Uploading an Image

Let us see how to upload an image to a webserver. We convert a BufferedImage to byte array in order to send it to server.

We use Java class **ByteArrayOutputStream**, which can be found under **java.io** package. Its syntax is given below:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
ImageIO.write(image, "jpg", baos);
```

In order to convert the image to byte array, we use **toByteArray** method of **ByteArrayOutputStream** class. Its syntax is given below:

```
byte[] bytes = baos.toByteArray();
```

Apart from the above method, there are other methods available in the ByteArrayOutputStream class as described briefly:

Sr.No.	Methods
1	**public void reset** This method resets the number of valid bytes of the byte array output stream to zero, so that all the accumulated output in the stream is discarded.
2	**public byte[] toByteArray** This method creates a newly allocated Byte array. Its size would be the current size of the output stream and the contents of the buffer will be copied into it. It returns the current contents of the output stream as a byte array.

3

public String toString

Converts the buffer content into a string. Translation will be done according to the default character encoding. It returns the String translated from the buffer's content.

4

public void write*intw*

It writes the specified array to the output stream.

5

public void write*byte[]b, intof, intlen*

It writes len number of bytes starting from offset off to the stream.

6

public void writeTo*OutputStreamoutSt*

It writes the entire content of this Stream to the specified stream argument.

Example

The following example demonstrates ByteArrayOutputStream to upload an image to the server:

Client Code

```
import javax.swing.*;
import java.net.*;
import java.awt.image.*;
import javax.imageio.*;
import java.io.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;

public class Client{
   public static void main(String args[]) throws Exception{

      Socket soc;
      BufferedImage img = null;
      soc=new Socket("localhost",4000);
      System.out.println("Client is running. ");

      try {

         System.out.println("Reading image from disk. ");
         img = ImageIO.read(new File("digital_image_processing.jpg"));
         ByteArrayOutputStream baos = new ByteArrayOutputStream();

         ImageIO.write(img, "jpg", baos);
         baos.flush();

         byte[] bytes = baos.toByteArray();
         baos.close();

         System.out.println("Sending image to server. ");

         OutputStream out = soc.getOutputStream();
```

```
            DataOutputStream dos = new DataOutputStream(out);

            dos.writeInt(bytes.length);
            dos.write(bytes, 0, bytes.length);

            System.out.println("Image sent to server. ");

            dos.close();
            out.close();

         }catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
            soc.close();
         }
         soc.close();
      }
}
```

Server Code

```
import java.net.*;
import java.io.*;
import java.awt.image.*;

import javax.imageio.*;
import javax.swing.*;

class Server {
   public static void main(String  args[]) throws Exception{
      ServerSocket server=null;
      Socket socket;
      server = new ServerSocket(4000);
      System.out.println("Server Waiting for image");

      socket = server.accept();
      System.out.println("Client connected.");

      InputStream in = socket.getInputStream();
      DataInputStream dis = new DataInputStream(in);

      int len = dis.readInt();
      System.out.println("Image Size: " + len/1024 + "KB");

      byte[] data = new byte[len];
      dis.readFully(data);
      dis.close();
      in.close();

      InputStream ian = new ByteArrayInputStream(data);
      BufferedImage bImage = ImageIO.read(ian);

      JFrame f = new JFrame("Server");
      ImageIcon icon = new ImageIcon(bImage);
      JLabel l = new JLabel();

      l.setIcon(icon);
      f.add(l);
      f.pack();
      f.setVisible(true);
   }
}
```

Output

Client Side Output

When you execute the client code, the following output appears on client side:

```
Client is running.
Reading image from disk.
Sending image to server.
Image sent to server.
```

Server Side Output

When you execute the server code, the following ouptut apppears on server side:

```
Server Waiting for image
Client connected.
Image Size: 29KB
```

After receiving the image, the server displays the image as shown below: