

# JAVA - Upload di immagini con ridimensionamento (proporzionato)

## FORM HTML

```
<form action="#" method="post" enctype="multipart/form-data">

    <input type="file" name="foto1" id="foto1" />

    ...n file

</form>
```

## SERVLET DI RISPOSTA AL FORM

Il metodo riportato qui di sotto carica tramite HTTP tutti gli oggetti file (input type=file) presenti nel form e li memorizza in un path temporaneo.

Successivamente, per tutti i file immagine, verifica se rispettano se le dimensioni (altezza e larghezza) massime consentite. In caso contrario verranno ridimensionate e rinominate altrimenti solo rinominate.

NB 10 \* 1024 \* 1024 indica che possono essere caricati massimo 10MB, se la dimensione della request supera tale dimensione viene generata un'eccezione (IOException)

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
```

```
String path = "PATH"
```

```
String size = 10 //MASSIMO 10MB
```

```
MultipartRequest mpr = new MultipartRequest(request, path, size * 1024 * 1024);
```

```
Enumeration enums = mpr.getFileNames();
```

```
ArrayList pageImg = new ArrayList();
```

```
int maxH = 640 //ALTEZZA MASSIMA DI UN IMMAGINE CARICABILE
```

```
int maxW = 480 //LARGHEZZA MASSIMA DI UN IMMAGINE CARICABILE
```

```
String randomName = Long.toString(System.currentTimeMillis());
```

```
ImageFile img = null;
```

```
File f = null;
```

```
for (int i = 1; enums.hasMoreElements(); i++){
```

```
    pageImg.add(mpr.getFilesystemName((String) enums.nextElement()));
```

```
}
```

```
for (int i=0; i < pageImg.size ; i++){
```

```
    if (pageImg.get(i) != null) {
```

```
        img = new ImageFile();
```

```
        f = new File(path + pageImg.get(i));
```

```
        System.out.println("    Altezza : " + img.getHeight());
```

```
        System.out.println("    Larghezza : " + img.getWidth());
```

```
        if (img.getHeight() > maxH) {
```

```
            System.out.println("    Ridimensionato");
```

```
            if (img.getWidth() > maxW) {
```

```
                img.resize(maxW, 0);
```

```

        }else{
            img.resize(0,maxH);
        }
        f.delete();
    }else{
        f.renameTo(new File(path + i + "_" + randomName + ".jpg"));
        System.out.println("    Non necessita di ridimensionamento");
    }
    pageImg.set(i, i + "_" + randomName + estensione);
}
}
}

```

## CLASSE IMAGE

E' necessario importare il package java.awt.\*, javax.imageio.\* e java.io.\*

```

public class ImageFile {

    BufferedImage image;
    String outFile, inFile;
    int active;

    public ImageFile() {
        active = 0;
    }

    public int getState(){
        return active;
    }

    public void setFile(String path, String out){
        active = 1;
        try {
            image = ImageIO.read(new File(path));
        } catch (Exception ex) {}
        outFile = new String(out);
        inFile = new String(out);
    }

    public void setOut (String out){
        this.outFile = out;
        if (getExtension(out) == null)
            outFile = outFile + "." + getExtension(inFile);
    }

    public int getWidth(){
        return image.getWidth();
    }

    public int getHeight(){
        return image.getHeight();
    }
}

```

```

public static String getExtension(String s) {
    String ext = null;
    int i = s.lastIndexOf('.');

    if (i > 0 && i < s.length() - 1) {
        ext = s.substring(i+1).toLowerCase();
    }
    return ext;
}

public void resize (int w, int h){
    double scale;
    if (h==0)
        scale = ((double)w / (double)image.getWidth());
    else scale = ((double)h / (double)image.getHeight());

    AffineTransformOp op = new
        AffineTransformOp(AffineTransform.getScaleInstance(scale, scale),
            AffineTransformOp.TYPE_NEAREST_NEIGHBOR);
    image = op.filter(image, null);
    try {
        ImageIO.write(image, getExtension(inFile), new File(outFile));
    } catch (Exception ex) {}
}
}
}

```