File Upload Service

Summary

Upload means sending files from one computer system to the other one. Generally, this term is used when sending a file from a small system to a large system. From the aspect of network user, uploading file is to send the file to the other computer set to receive file. If you want to share the image file with other user in the electronic board, upload the file to that board.

Then, the person at the opposite download the file. Here, download means the transfer of file from big computer to small computer. From the viewpoint of internet user, download means receiving file from other computer.

Description

To implement the file upload function, the MultiCommonsMultipartResolver should be defined in an empty configuration file. This guide recommends to use CommonsMultipartResolver provided by Apache Commens FileUpload. If updating and using CommonsMultipartResolver, lots of time and efforts will be required.

```
        <!-- Custom MultiFile resolver -->
<bean id="local.MultiCommonsMultipartResolver"
        class="egovframework.rte.util.web.resolver.MultiCommonsMultipartResolver">
        <property name="maxUploadSize" value="100000000" />
        <property name="maxInMemorySize" value="100000000" />
</bean>
```

[Recommended] To use **CommonsMultipartResolver** that processes the file upload using Apache Commens FileUpload API provided by spring, define as follows in the empty setting file.

```
<!-- MULTIPART RESOLVERS -->
<!-- regular spring resolver -->
<bean id="spring.RegularCommonsMultipartResolver"
        class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize" value="100000000" />
        <property name="maxInMemorySize" value="100000000" />
</bean>
```

In addition, designate the upload location of file with property of relevant controller and get the file upload location designated through setter method in controller. Following is the example of use.

fileUploadProperties.properties

```
# In case of windows NT
file.upload.path=C:\\temp

# In case of Unix
file.upload.path=/usr/file/upload
@Resource(name = "fileUploadProperties")
Properties fileUploadProperties;
..
..
..
        String uploadPath = fileUploadProperties
                        .getProperty("file.upload.path");
        File saveFolder = new File(uploadPath);
..
..
..
```

To upload file, designate the input form file of **JSP file** to file and designate the enctype of form to multipart/form-data. Example is as follows. At this tie, if using **CommonsMultipartResolver**, set the name of form different. If using the duplicate name, an error occurs.

```
<form method="post" action="<c:url value='/upload/genericMulti.do'/>"
enctype="multipart/form-data">
  <p>Type:
    <input type="text" name="type" value="genericFileMulti" size="60" />
  </p>
  <p>File1:
    <input type="file" name="file[]" size="60" />
  </p>
  <p>File2:
    <input type="file" name="file[]" size="60" />
  </p>
  <p>
    <input type="submit" value="Upload" />
  </p>
</form>
```

Next is the image that implemented the controller to upload the file.

```
@Controller("genericFileUploadController")
public class GenericFileUploadController {

@Resource(name = "multipartResolver")
CommonsMultipartResolver multipartResolver;

@Resource(name = "fileUploadProperties")
Properties fileUploadProperties;

@SuppressWarnings("unchecked")
@RequestMapping(value = "/upload/genericMulti.do")
public String multipartProcess(final HttpServletRequest request, Model model)
          throws Exception {

final long startTime = System.nanoTime();

/*
 * validate request type
 */
Assert.state(request instanceof MultipartHttpServletRequest,
                    "request !instanceof MultipartHttpServletRequest");
final MultipartHttpServletRequest multiRequest = (MultipartHttpServletRequest) request;

/*
 * validate text input
 */
Assert.state(request.getParameter("type").equals("genericFileMulti"),
                    "type != genericFileMulti");

/*
 * extract files
 */
final Map<String, MultipartFile> files = multiRequest.getFileMap();
Assert.notNull(files, "files is null");
Assert.state(files.size() > 0, "0 files exist");

/*
 * process files
 */
String uploadPath = fileUploadProperties
```

```
                    .getProperty("file.upload.path");
File saveFolder = new File(uploadPath);

// Directory generation
if (!saveFolder.exists() || saveFolder.isFile()) {
          saveFolder.mkdirs();
}

Iterator<Entry<String, MultipartFile>> itr = files.entrySet()
                    .iterator();
MultipartFile file;
List fileInfoList = new ArrayList();
String filePath;

while (itr.hasNext()) {
          Entry<String, MultipartFile> entry = itr.next();
          System.out.println("[" + entry.getKey() + "]");

          file = entry.getValue();
          if (!"".equals(file.getOriginalFilename())) {
                    filePath = uploadPath + "\\" + file.getOriginalFilename();
                    file.transferTo(new File(filePath));

                    FileInfoVO fileInfoVO = new FileInfoVO();
                    fileInfoVO.setFilePath(filePath);
                    fileInfoVO.setFileName(file.getOriginalFilename());
                    fileInfoVO.setFileSize(file.getSize());
                    fileInfoList.add(fileInfoVO);
          }
}

// Here, the related information is not stored on the DB and simply forwarded to success page to
reconfirm.
model.addAttribute("fileInfoList", fileInfoList);
model.addAttribute("uploadPath", uploadPath);

final long estimatedTime = System.nanoTime() - startTime;
System.out.println(estimatedTime + " " + getClass().getSimpleName());

return "success";

          }
}
```

Tomcat requires character set encoding through request.setCharacterEncoding() when web application receives parameter in the type of GET and POST.

Reference