

# 14장. 파일 업로드

# 파일업로드를 위한 기본 폼

1. 웹 브라우저를 통해서 파일을 전송하기 위해서 폼 구성
  - 1) form 태그의 속성들 중 input 태그가 있는데 이들 중 type 속성의 속성값이 file인 `<input type="file">` 태그는 파일을 선택할 수 있는 창을 제공
  - 2) form 태그의 속성들 중 method의 속성값은 "post"를 enctype의 속성값은 "multipart/form-data"을 사용  
`<form name="formName" method="post" enctype="multipart/form-data">`  
`<input type="file" name="selectfile">`  
`</form>`
2. `<form method="post">`의 형태로 전송한 폼에 담겨진 파라미터들은 request 객체를 통해서 해당되는 값을 얻음
3. form 엘리먼트에는 enctype을 multipart/form-data라는 attribute를 기재해야 하는데 이를 기재하지 않으면 웹 서버로 데이터를 전송할 때 파일 내용을 전송하는 것이 아니라 파일의 경로명만 전송되며 이 속성이 제대로 동작하도록 하려면 form의 전송 방식이 **post** 방식이어야만 동작.
  - ▶ enctype="multipart/form-data"로 전송한 폼에 담겨진 파라미터들에 대한 이름과 값을 얻어내고  
`<input type="file">`로 지정된 파일을 서버상의 한 폴더에 업로드하기 위해 특별한 컴포넌트가 필요.
4. cos.jar를 사용
  - ① cos.jar 파일 다운로드 <http://www.servlets.com>
  - ② cos-05Nov.zip파일의 압축 해제  
cosWlib폴더 안에 있는 cos.jar 파일을 톰캣홈 lib폴더에 복사
  - ③ 안전한 서비스를 위해 cos.jar파일을 톰캣홈Wcommonswlib폴더에 복사
  - ④ 폴더에 업로드될 파일을 모아놓을 저장소인 fileSave폴더를 만듦

# 파일 업로드하고 폼 데이터를 분석 도구

★ 파일업로드 및 폼요소처리 MultipartRequest 클래스

- ① COS 라이브러리에서가장핵심적인역할을하는클래스로 파일 업로드 담당
- ② MultipartRequest의생성자

```
MultipartRequest(javax.servlet.http.HttpServletRequest request,  
                java.lang.String saveDirectory,  
                int maxPostSize,  
                java.lang.String encoding,  
                FileRenamePolicy policy)
```

인자	설명
request	MultipartRequest와 연결될 request 객체를 의미한다.
saveDirectory	서버 측에 파일이 실질적으로 저장될 경로를 의미한다.
maxPostSize	한번에 업로드 할 수 있는 최대 파일 크기를 의미한다.
encoding	파일의 인코딩 방식을 의미한다.
policy	파일 이름 중복 처리를 위한 인자를 의미한다.

# MultipartRequest 클래스의 메소드

Return type	Method
java.lang.String	getContentType(java.lang.String name):업로드된 파일의 콘텐츠 타입을 반환. 업로드된 파일이 없으면 null을 반환한다
java.io.File	getFile(java.lang.String name):서버 상에 업로드된 파일의 파일객체를 반환. 업로드된 파일이 없다면 null을 반환한다.
java.util.Enumeration	getFileNames():폼요소 중 input 태그속성이 file로 된 파라미터의 이름들을 반환. upload된 파일이 없으면 비어있는 Enumeration을 반환한다.
java.lang.String	getFileName(java.lang.String name):사용자가 지정해서 서버 상에 실제로 업로드된 파일명을 반환한다
java.lang.String	getOriginalFileName(java.lang.String name):사용자가 지정해서 서버 상에 업로드된 파일명을 반환. 이 때의 파일명은 파일중복을 고려한 파일명 변경전의 이름을 말한다.
java.lang.String	getParameter(java.lang.String name):스트링으로 주어진 이름에 대한 값을 반환. 값없이 파라미터가 전송되었거나, 해당되는 이름의 파라미터가 전송이 안 되었을 경우 null을 반환한다.
java.util.Enumeration	getParameterNames():모든 파라미터 이름을 Enumeraton으로 반환
java.lang.String[]	getParameterValues(java.lang.String name):주어진 이름에 대한 값을 스트링 배열로 반환. 파라미터가 전송되지 않았을 때는 null을 반환

# 폼 데이터 분석 및 파일 업로드 구현

1. 파일을 선택할 수 있는 fileSelect.jsp 페이지를 코딩하고 저장  
`<form name="fileForm" method="post" enctype="multipart/form-data" action="fileUpload.jsp">`
2. 폼 데이터를 분석하고 파일 업로드를 구현할 fileUpload.jsp 페이지를 코딩하고 저장
  - ▶ `multi = new MultipartRequest( request  
                                    ,realFolder  
                                    ,maxSize  
                                    ,encType  
                                    ,new DefaultFileRenamePolicy());`

# fileSelect.jsp 예시

```
<%@ page contentType="text/html; charset=euc-kr"%>
<html>
  <head>
    <title>파일 업로드 예제 </title>
    <meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
  </head>
  <body>
    <form name="fileForm" method="post" enctype="multipart/form-data" action="fileUpload.jsp">
      작성자: <input type="text" name="user"> <br>
      제 목: <input type="text" name="title"> <br>
      파일명: <input type="file" name="uploadFile"> <br>
             <input type="submit" value="파일 올리기"> <br>
    </form>
  </body>
</html>
```

# fileUpload.jsp 예시1

```
<%@page import="com.oreilly.servlet.multipart.DefaultFileRenamePolicy"%>
<%@page import="com.oreilly.servlet.MultipartRequest"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
import="java.util.*,java.io.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insert title here</title>
</head>
<body>
    <% request.setCharacterEncoding("utf-8");
    int maxSize = 5 * 1024 * 1024;
    String fileSave = "/fileSave";
    String realPath = getServletContext().getRealPath(fileSave);
    MultipartRequest multi = new MultipartRequest(request,realPath, maxSize,"utf-8", new
    DefaultFileRenamePolicy());
    // Enumeration en = multi.getFileNames();
    // while(en.hasMoreElements()) {
    //input 태그의 속성이 file인 태그의 name 속성값 :파라미터이름
    //String name = (String)en.nextElement();
    //서버에 저장된 파일 이름
    String name1 = "uploadFile";
    String filename = multi.getFilesystemName(name1);
    //전송전 원래의 파일 이름
    String original = multi.getOriginalFileName(name1);
```

# fileUpload.jsp 예시2

//전송된 파일의 내용 타입

```
String type = multi.getContentType(name1);  
//전송된 파일속성이 file인 태그의 name 속성값을 이용해 파일객체생성  
File file = multi.getFile(name1);  
out.println("파라미터 이름 : " + name1 + "<br>");  
out.println("실제 파일 이름 : " + original + "<br>");  
out.println("저장된 파일 이름 : " + filename + "<br>");  
out.println("파일 타입 : " + type + "<br>");  
if(file!=null){  
    out.println("크기 : " + file.length() + "<br>");  
}
```

```
String name = multi.getParameter("name");  
String title = multi.getParameter("title");
```

%>

작성자 : <%=name %><p>

제목 : <%=title %>

</body>

</html>



# Style.css

```
a:link,a:visited,a:active {  
    text-decoration : none;  
}  
a:hover {  
    text-decoration : underline; color : red;  
}  
TABLE, TD,TR, TH{  
    font-size:9pt;  
}  
p{  
    font-size:9pt;  
}
```

# 썸네일 이미지의 개요

1. 썸네일(Thumbnail) 이미지란 원본이미지와 별도로 제공하는 작은 이미지
2. 이미지파일에 대해 원래크기의 이미지와 작은 사이즈의 이미지가 동시에 제공되어야 하는 경우
  - ① 원래이미지를 웹 디자이너가 작은 사이즈로 만들어주는 작업을 수행
  - ② 원래크기의 이미지를 올릴 때(업로드 할 때) 자동으로 작은 사이즈의 이미지를 생성해 준다면 웹 디자이너가 일일이 원본그림에 대한 썸네일 이미지를 만들어 줄 필요가 없음
3. 썬 마이크로시스템즈(Sun Microsystems)에서 제공하는 JAI(Java Advance Imaging) API를 다운로드 받아서 작성
4. JAI(Java Advance Imaging) API 다운로드 및 설치 -<http://java.sun.com/products/java-media/jai/>
5. <http://www.oracle.com/technetwork/java/current-142188.html>
  - ▶ 썸네일(Thumbnail) 이미지의 생성하기
  - ▶ JAI(Java Advance Imaging) API
    - ① JAI(Java Advance Imaging) API 100여개의 이상의 이미지 처리 오퍼레이션을 제공하며, Byte, UShort, Short, 32-bit int, floats/double, n-banded images 이미지 포맷과 데이터 타입과 Image File 입출력으로 BMP, GIF, FPX, JPEG, PNG, PNM, TIFF. 포맷을 제공
    - ② JAI 클래스는 오퍼레이션들의 객체를 사용하기 쉽게 해줌
      - import javax.media.jai.JAI;
      - RenderedOp im = JAI.create("fileload", param);
      - create()메소드는 RenderedOp 객체를 리턴

# JAI download1



The screenshot shows a web browser window with the address bar displaying `http://java.sun.com/javase/technologies/desktop/media/`. The page is the Oracle Sun Developer Network (SDN) for Java Media APIs. The header includes the Oracle logo, the SDN name, and navigation links for Sun, Java, Solaris, Communities, and My SDN Account. A secondary navigation bar lists APIs, Downloads, Products, Support, Training, and Participate, along with a search bar. The main content area is titled "Java Media APIs" and features a sub-header "Many technologies, one platform" with a link to "Get Java SE". Below this is a tabbed interface with "Technologies" selected, showing a list of technologies including Desktop, Accessibility, Security, Tools, Web Services, and Real-Time. The "Desktop" tab is active, displaying a "Desktop Overview" section that describes support for advanced media capabilities. This section includes links to "Java 2D" and "Java Bindings for OpenGL (JOGL)". The "JOGL" link is highlighted, leading to a detailed description of JOGL as a Java programming language binding for the OpenGL 3D graphics API. It mentions integration with AWT and Swing, and provides access to the latest OpenGL routines. A red button labeled "» Download JOGL" is visible, with a note that it is distributed through the project [joгл](#). Below the JOGL section, the "Java Advanced Imaging (JAI)" section is partially visible, describing a set of object-oriented interfaces for image manipulation, with a red button labeled "» Download JAI" and a note that it is an optional Java SE API.

ORACLE® Sun Developer Network (SDN)

Sun ▾ Java ▾ Solaris ▾ Communities My SDN Account ▾

APIs Downloads Products Support Training Participate » search tips Search

SDN Home > Java Technology > Java SE >

## Java Media APIs

Many technologies, one platform  
Java SE technologies provide the functionality to develop and run applications  
» Get Java SE

Overview Technologies Documentation Community Support Downloads

At a Glance | Core | Database | Desktop | Accessibility | Security | Tools | Web Services | Real-Time

» Desktop Overview » Documentation » Articles

Support for advanced media capabilities on the Java platform is provided via a suite of optional packages for graphics, image I/O, image processing, and time-based media (video, audio, etc.).

### Java 2D

The Java 2D API is a set of classes for advanced 2D graphics and imaging, encompassing line art, text,

### Java Bindings for OpenGL (JOGL)

JOGL is a Java programming language binding for the OpenGL 3D graphics API. It supports integration with the Java platform's AWT and Swing widget sets while providing a minimal and easy-to-use API that handles many of the issues associated with building multithreaded OpenGL applications. JOGL provides access to the latest OpenGL routines (OpenGL 2.0 with vendor extensions) as well as platform-independent access to hardware-accelerated offscreen rendering ("pbuffers"). JOGL also provides some of the most popular features introduced by other Java bindings for OpenGL like GL4Java, LWJGL and Magician, including a composable pipeline model which can provide faster debugging for Java-based OpenGL applications than the analogous C program. » Read More

» Download JOGL  
*Distributed through project [joгл](#)*

### Java Advanced Imaging (JAI)

The Java Advanced Imaging API provides a set of object-oriented interfaces that support a simple, high-level programming model which lets you manipulate images easily. » Read More

» Download JAI  
*Optional Java SE API*

» Java SE Site Map

Update Your Java Runtime Environment

Related Resources  
» Java for Business

# JAI download2

Oracle Technology Network

Products and Services | Solutions | Downloads | Store | Support | Training | Partners | About | Oracle Technology Network

Oracle Technology Network > Java

**Downloads**

Java Advanced Imaging Downloads

[Java Advanced Imaging API 1.1.3 Download Page](#)

[Java Advanced Imaging API 1.1.2 01 Download Page](#)

[Java Advanced Imaging API 1.1.2 Download Page](#)

[Java Advanced Imaging API 1.1.1 01 Download Page](#)

[Java Advanced Imaging API 1.1.1 Download Page](#)

[Java Advanced Imaging API 1.0.2 Download Page](#)

Java Advanced Imaging-Image I/O Tools Downloads

[Java Advanced Imaging Image I/O Tools 1.1 Download](#)

[Java Advanced Imaging Image I/O Tools 1.0 01 Download](#)

[Java Advanced Imaging Image I/O Tools 1.0 Download](#)

Java Advanced Imaging Third Party Utilities

[JAI Utilities](#)

**Java SDKs and Tools**

[Java SE](#)

[Java EE and Glassfish](#)

[Java ME](#)

[JavaFX](#)

[Java Card](#)

[NetBeans IDE](#)

**Java Resources**

[New to Java?](#)

[APIs](#)

[Code Samples & Apps](#)

[Developer Training](#)


[Documentation](#)

[Java.com](#)

[Java.net](#)

[Student Developers](#)

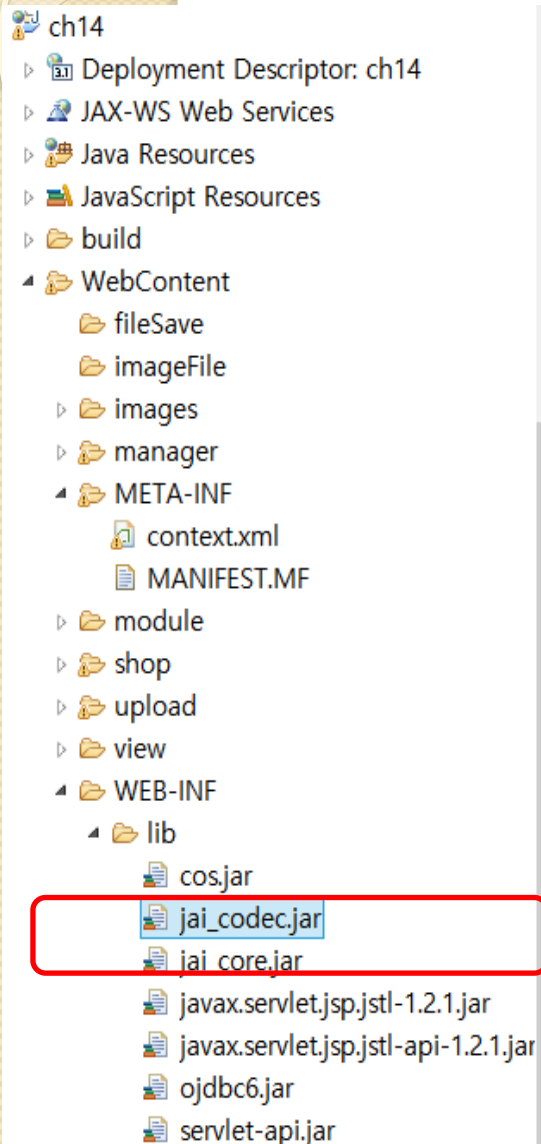
[Tutorials](#)



# JAI download3

	<a href="#">jai-1_1_3-lib-solaris-amd64.jar.zip</a>	18-Sep-2006 13:22	1M
	<a href="#">jai-1_1_3-lib-solaris-amd64.tar.gz</a>	18-Sep-2006 13:22	3M
	<a href="#">jai-1_1_3-lib-solaris-i586-jdk.bin</a>	18-Sep-2006 13:22	3M
	<a href="#">jai-1_1_3-lib-solaris-i586-jre.bin</a>	18-Sep-2006 13:22	3M
	<a href="#">jai-1_1_3-lib-solaris-i586.jar.zip</a>	18-Sep-2006 13:22	3M
	<a href="#">jai-1_1_3-lib-solaris-i586.tar.gz</a>	18-Sep-2006 13:22	3M
	<a href="#">jai-1_1_3-lib-solaris-sparc-jdk.bin</a>	18-Sep-2006 13:23	23M
	<a href="#">jai-1_1_3-lib-solaris-sparc-jre.bin</a>	18-Sep-2006 13:23	23M
	<a href="#">jai-1_1_3-lib-solaris-sparc.jar.zip</a>	18-Sep-2006 13:23	23M
	<a href="#">jai-1_1_3-lib-solaris-sparc.tar.gz</a>	18-Sep-2006 13:23	23M
	<a href="#">jai-1_1_3-lib-solaris-sparcv9-jdk.bin</a>	18-Sep-2006 13:23	24M
	<a href="#">jai-1_1_3-lib-solaris-sparcv9-jre.bin</a>	18-Sep-2006 13:23	24M
	<a href="#">jai-1_1_3-lib-solaris-sparcv9.jar.zip</a>	18-Sep-2006 13:23	24M
	<a href="#">jai-1_1_3-lib-solaris-sparcv9.tar.gz</a>	18-Sep-2006 13:24	26M
	<a href="#">jai-1_1_3-lib-windows-i586-jdk.exe</a>	18-Sep-2006 13:24	9M
	<a href="#">jai-1_1_3-lib-windows-i586-jre.exe</a>	18-Sep-2006 13:24	9M
	<a href="#">jai-1_1_3-lib-windows-i586.exe</a>	18-Sep-2006 13:24	9M

# JAI download4



```
5      type="javax.sql.DataSource"
6      username="scott"
7      password="tiger"
8      driverClassName="oracle.jdbc.driver.OracleDriver"
9      factory="org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory"
10     url="jdbc:oracle:thin:@127.0.0.1:1521:orcl"
11     maxActive="100"
12     maxIdle="10"/>
13 <Resource
14     name="jdbc/MySql"
15     auth="Container"
16     type="javax.sql.DataSource"
17     username="root"
18     password="mysql"
19     driverClassName="com.mysql.jdbc.Driver"
20     factory="org.apache.tomcat.dbcp.dbcp2.BasicDataSourceFactory"
21     url="jdbc:mysql://localhost:3306/test"
22     maxActive="100"
23     maxIdle="10"/>
24 </Context>
```

context.xml

Markers Properties Servers Data Source Explorer Console

Android

# 작업 전에 image folder 생성

thumbnailForm.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"%>
<html>
  <head> <title>썸네일 이미지 폼</title> </head>
  <body>
    <center> <h3>썸네일 이미지 폼 예제</h3>
    <form action="thumbnail.jsp" method="post" enctype="multipart/form-data">
      이미지 파일 : <input type="file" name="filename"> <p>
      <input type="submit" value="전송">
    </form>
  </center>
</body>
</html>
```

# Thumbnail.jsp I

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"%>
<%@ page import="java.awt.Graphics2D"%>
<%@ page import="java.awt.image.renderable.ParameterBlock"%>
<%@ page import="java.awt.image.BufferedImage"%>
<%@ page import="javax.media.jai.JAI"%>
<%@ page import="javax.media.jai.RenderedOp"%>
<%@ page import="javax.imageio.ImageIO"%>
<%@ page import="com.oreilly.servlet.MultipartRequest"%>
<%@ page import="com.oreilly.servlet.multipart.DefaultFileRenamePolicy"%>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*"%>
<%
    String imagePath=getServletContext().getRealPath("/fileSave");
    int size = 2*1024*1024 ; String filename="";
    try{
        MultipartRequest multi=new MultipartRequest(request, imagePath,size,"utf-8", new
        DefaultFileRenamePolicy());
        Enumeration files=multi.getFileNames();
        String file =(String)files.nextElement();
        filename=multi.getFilesystemName(file);
    }catch(Exception e){
        e.printStackTrace();
    }
}
```



## Thumbnail.jsp 2

```
// 이 클래스에 변환할 이미지를 담는다.  
// (이미지는 ParameterBlock을 통해서만 담을수 있다.)  
ParameterBlock pb=new ParameterBlock();  
// 서버에 저장된 원본파일의 경로로 파라미터블록에 추가  
//위에서 가져온 파일이름을 받아서 이미지패스에 지정한 폴더 속에 파일을 만들어줌  
pb.add(imagePath+"/"+filename);  
// 자이로 파라미터블록을 로드하여 RenderedOp 에 삽입  
RenderedOp rOp=JAI.create("fileload",pb);  
// 불러온 이미지를 BufferedImage에 담는다.  
BufferedImage bi= rOp.getAsBufferedImage();  
// thumb라는 이미지 버퍼를 생성, 버퍼의 사이즈는 100*100으로 설정.  
BufferedImage thumb=new BufferedImage(100,100, BufferedImage.TYPE_INT_RGB);  
Graphics2D g=thumb.createGraphics();  
//버퍼사이즈 100*100으로 맞춰 그리자  
g.drawImage(bi,0,0,100,100,null);  
/*출력할 위치와 파일이름을 설정하고 섬네일 이미지를 생성한다. 저장하는 타입을 jpg로 설정.*/  
//그 변형한 파일을 파일명 변경시킨다  
File file=new File(imagePath+"/sm_"+filename);  
//버퍼공간의 영역에 변경한 이미지 파일명을 불러와 jpg속성으로 출력시킨다  
ImageIO.write(thumb,"jpg",file);
```

%>

## Thumbnail.jsp 3

```
<html>
<head>
  <title> 이미지 썸네일 예제 </title>
</head>
<body>
  -원본 이미지-<br>
   <p>
  -썸네일 이미지-<br>
  
</body>
</html>
```