

[파일 업로드 구현] oreilly 의 cos 라이브러리 download(www.servlets.com/cos)

cos-26Dec2008.zip 파일 다운로드 후, \lib\cos.jar 파일을

WEB-INF\lib 폴더 또는 Tomcat 설치폴더\lib 폴더에 복사, Tomcat 재시작

(참고 - WEB-INF\lib 폴더에 저장하면 이 웹 어플리케이션에서만 사용가능.

Tomcat 설치폴더\lib 폴더에 복사 하면, 이 톰캣 상에 구동되는 모든 웹 어플리케이션에서만 사용가능)

/WebContent/upload/fileUpload.html

```
<html>
<body>
<form action="fileupload.jsp" method="post"
enctype="multipart/form-data">

파일: <input type="file" name="f"> <br>
설명: <input type="text" name="desc"> <br>
<input type="submit" value="업로드">
</form></body></html>
```

파일 전송 시, 반드시
form 태그 속성은 다음과
같아야 함.

method="post"
enctype="multipart/
form-data"

/WebContent/upload/fileUpload.jsp

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ page import="com.oreilly.servlet.MultipartRequest,
com.oreilly.servlet.multipart.DefaultFileRenamePolicy, java.io.*"%>
<html><body>

<%
String savePath = application.getRealPath("/upload");
int sizeLimit = 5*1024*1024;//파일 업로드 사이즈 제한

MultipartRequest mr = new MultipartRequest(request,
savePath, sizeLimit, "UTF-8",
new DefaultFileRenamePolicy() );

File file = mr.getFile("f");
out.println("저장 경로 :" + savePath + "<br/>");
out.println("파일명: " + file.getName() + "<br/>");
out.println("파일크기: " + file.length() + "<br/>");
out.println("설명: " + mr.getParameter("desc") + "<br/>");
%>
</body></html>
```

new MultipartRequest(request,파일저장경로,업로드최대사이즈,"UTF-

8",동일명의 파일 이미 존재하는 경우 rename정책);←이 코드만으로 파일 업로드됨

파일: C:\Wj\workspace\ eclipse-jee-neon-3-win32\workspace\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\ex\upload
설명: 업로드 예제
업로드

업로드할 파일 선택

<< eclipse-jee-neon-3-win32 eclipse >>

구성 새 폴더

컴퓨터

BOOTC

\$Recy

99 Util

Boot

Docur

Intel

j

eclip

erlin

이름

configuration

dropins

features

p2

plugins

readme

.eclipseproduct

artifacts.xml

수정함

2017-07

2017-07

2017-07

2017-07

2017-07

2017-07

2017-01

2017-07

저장 경로 :C:\Wj\workspace\ eclipse-jee-neon-3-win32\workspace\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\ex\upload
파일명: eclipse.exe
파일크기: 326640
설명: 업로드 예제

[필터]

반복적인 작업을 매번 서블릿/JSP 에서 수행하는 것이 불편한 경우, 클라이언트의 요청이 서블릿에 전달되기 전에 처리하는 Preprocessor, 응답이 서블릿에서 나온 다음에 처리하는 Postprocessor 의 역할 (예) 인증필터, 이미지 변환필터, 데이터 압축필터, 암호화필터)

javax.servlet.Filter 인터페이스

```
void init(FilterConfig filterconfig)
    : 필터가 서비스에 들어가기 전에 한번만 호출.
void doFilter(ServletRequest req, ServletResponse resp,
              FilterChain chain)
    : 요청과 응답이 필터 체인을 통과할 때마다 호출.
    필터에서 처리된 결과는 필터 체인에서 다음 필터로 전달.
void destroy()
    : 필터가 서비스에서 삭제되지 전에 웹컨테이너에 의해서 호출.
```

필터 예제

1) Filter 인터페이스를 구현한 MyFilter 클래스를 작성한다.

```
package filter;
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
```

```

@WebFilter("*.jsp") //모든 jsp 파일 요청마다 호출되는 필터 등록
public class MyFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig)
                                throws ServletException {
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {

        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();
        out.print("My Filter  <br> ");

        long startTime = System.currentTimeMillis();
        chain.doFilter(request, response); //
        long stopTime = System.currentTimeMillis();

        System.out.println((stopTime - startTime) + "
milliseconds");
    }
    @Override
    public void destroy() {
    }
}

```

2) `@WebFilter("*.jsp")` 대신 아래와 같이 필터를 등록할 수도 있다.

WEB-INF/web.xml

```

<web-app>
...중략
<filter>
    <filter-name>myfilter</filter-name>
    <filter-class>filter.MyFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>myfilter</filter-name>
    <url-pattern>*.jsp</url-pattern>
</filter-mapping>
</web-app>

```

모든 jsp 파일 요청마다 MyFilter 가 호출됨