

动态web开发：

一、首先，都需要先创建web项目，

创建web项目：打开开发工具 - - - -> New - - - -> Web Project - - - -> 进入
创建页面 输入项目名

等信息

完成任意实现之前一定要先配置好tomcat服务器，实现之后一定要发布到服务上

二、其次有三种方法实现。

★方法1、创建实现servlet接口implements Servlet 所在

包 import javax.servlet.ServletException; 做为了解

★方法2、继承extends GenericServlet 或者 继承HttpServlet 需要手
动配置应用服务

配置：在WebRoot目录下的WEB-INF目录里的web.xml文件

打开web.xml文件配置内容如下：

1、必须要配置标签：<servlet>和<servlet-mapping>配置位置在<display-
name></display-name>下面<welcome-file-list>上面

2、配置内容：

<servlet>

<servlet-name>web02</servlet-name> Servlet服务应用名 可
自定义

<servlet-class>it.s511.SerWeb02</servlet-class> 应用对应处
理响应请求的类

</servlet>

<servlet-mapping> 配置映射 路径三选一即可

<servlet-name>web02</servlet-name> 映射的应用 其名同应用
名

<url-pattern>/xxx/yyy</url-pattern> 映射的访问路径 方法1

/xxx/yyy

`<url-pattern>/demo/*</url-pattern>` 映射的访问路径 方法2

/demo/*

`<url-pattern>*.do</url-pattern>` 映射的访问路径 方法3

*.do 或 *.action

`</servlet-mapping>`

配置完成之后访问网页的地址：

方法1：

http://IP地址:8080/项目名/xxx/yyyy

<http://192.168.0.108:8080/ServletWeb/ser/ssweb03>

方法2：

<http://IP地址:8080/项目名/demo/任意多个或一个字符>

http://192.168.0.108:8080/ServletWeb/demo/888

方法3：

<http://IP地址:8080/项目名/任意名.do>

<http://192.168.0.108:8080/ServletWeb/reg.do>

方法3扩展：

http://192.168.0.108:8080/ServletWeb/reg.action

3、可选配置：在`<servlet-class>it.s511.SerWeb02</servlet-class>`之后可配

servlet启动参数

对应的可以在对应处理响应请求的类的**init()**方法中获取相应的数据

```
<init-param>

    <param-name>p</param-name>

    <param-value>asdsd</param-value>

</init-param>
```

<load-on-startup>10</load-on-startup> 使servlet随服务启动

对应类的**init**方法:

@Override

```
    public void init() throws ServletException {

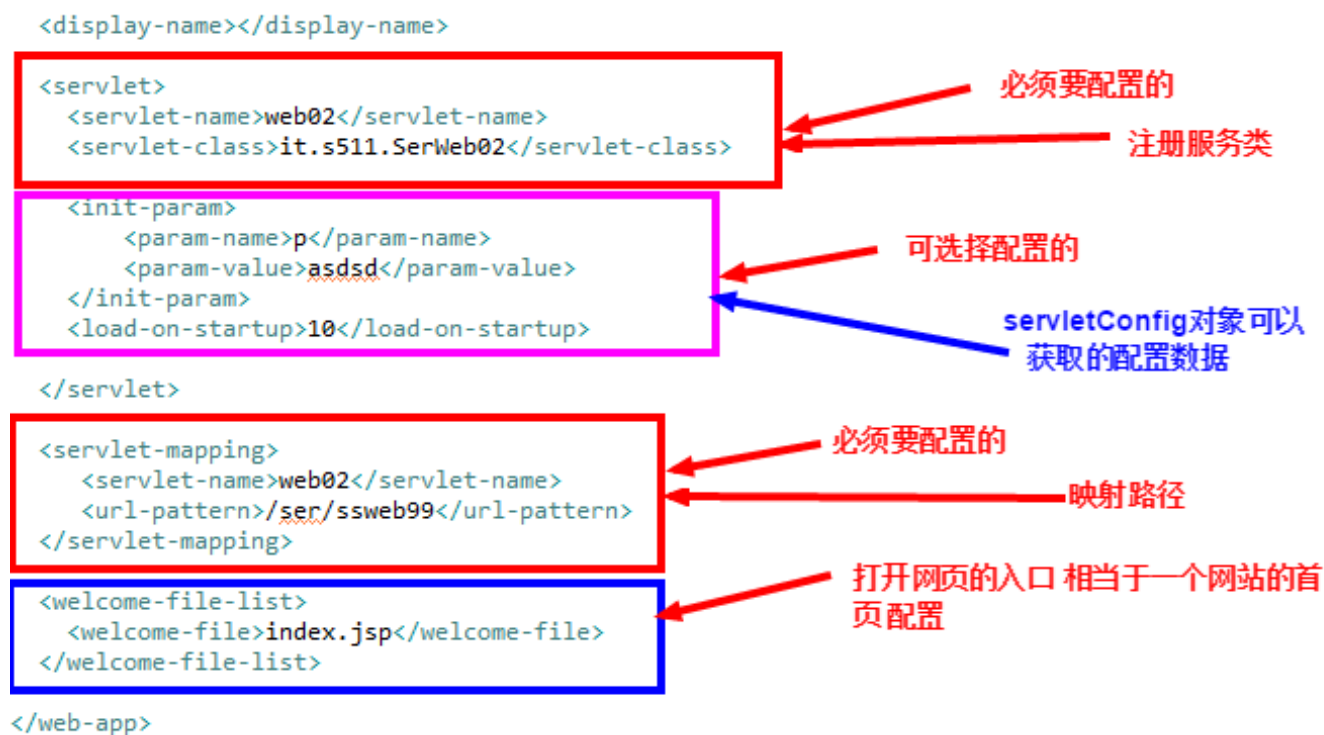
        super.init();

        System.out.print("服务器启动了"); 对应<b>load-on-startupinit-param
```

通servletConfig对象获取 可选配置数据

```
String username = servletConfig.getInitParameter("username");
String password = servletConfig.getInitParameter("password");
```

下面是手动配置的**web.xml**



★ ServletContext对象的使用

作用：在tomcat服务器加载web应用程序时候，会为 每个 web应用程序去创建一个唯一的与之对应的 servletContext对象，这个对象就代表着各自的那个 web应用程序。

//获得 servletContext

```
ServletContext servletContext = getServletContext();
```

使用场景:实现项目内数据的共享.

//在A服务类存数据

```
servletContext.setAttribute("name", "小丽");
```

//在B服务类取出数据

```
String name = (String) servletContext.getAttribute("name");
```

//获得web应用程序中一些资源的具体的路径 绝对的路径

```
String realPath = getServletContext().getRealPath("/5.jpg");
```

★ 获得web应用范围内全局的初始化参数信息

在web.xml文件中，可以配置 web应用程序全局的初始化参数信息。

<!-- WEB应用全局初始化参数信息 -->

```
<context-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</context-param>
```

通过servletContext获取

```
String value = servletContext.getInitParameter("encoding");
```

★ 使用ServletContext去统计网站的访问次数

@Override

```
public void init() throws ServletException {

    int count=0;

    getServletContext().setAttribute("count", count);

}
```

```
protected void doGet(HttpServletRequest request,
HttpServletRequest response) throws ServletException,
IOException {
```

```
    //维护 计数器的值
```

```
    //取出原有的值
```

```
    int count =(int)
```

```
getServletContext().getAttribute("count");
```

```
    //加 1
```

```
    count++;
```

//再存进去

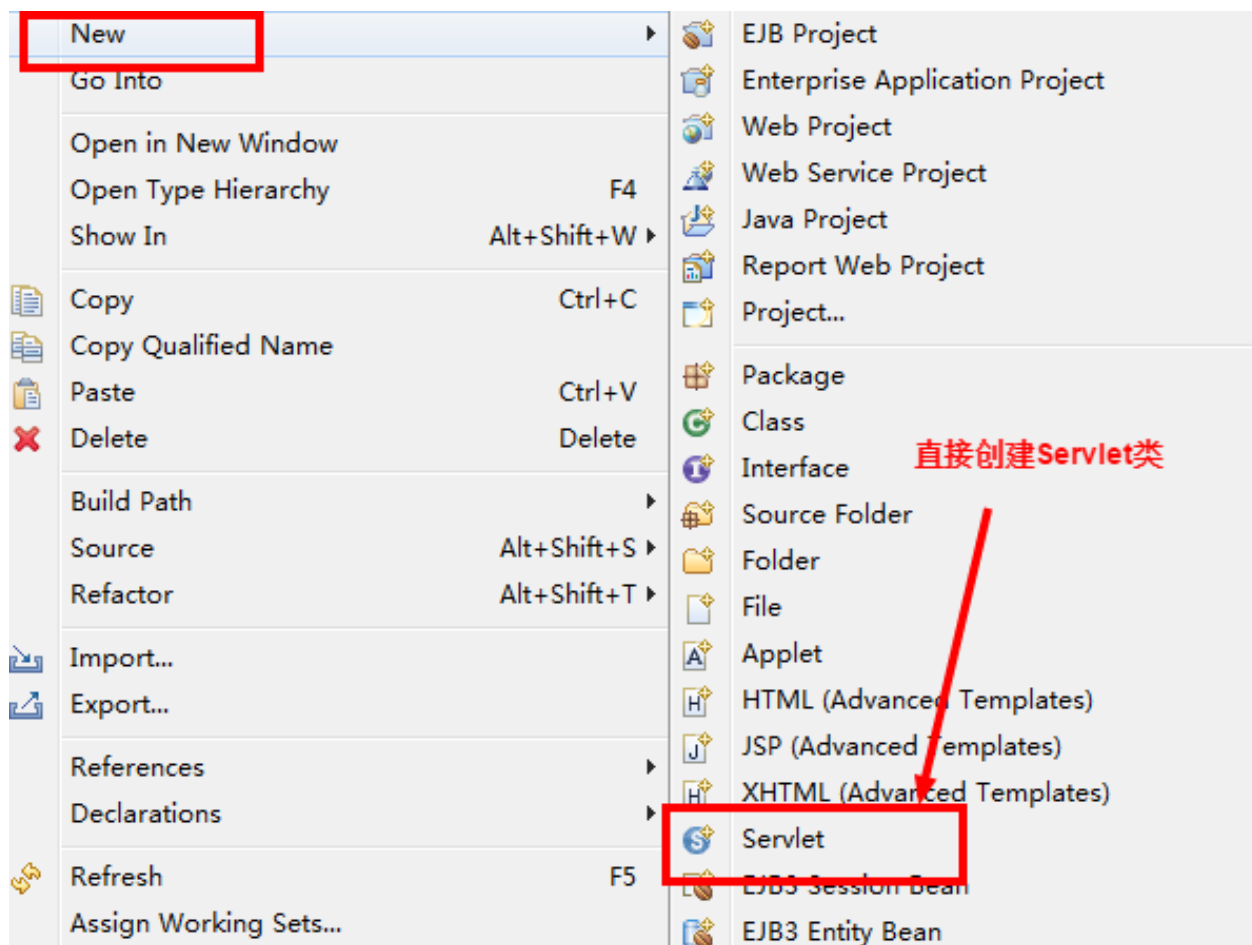
```
getServletContext().setAttribute("count", count);
```

//解决乱码

```
response.setContentType("text/html;charset=utf-8");
```

```
response.getWriter().print("网站被访问了 : " + count+"  
次 ");  
}
```

★方法3、创建完web项目之后直接New----->Strvlet - - >进行创建类选择



创建时可选

Source folder: ServletWeb/src Browse...

Package: it.s511 Browse...

☐ Enclosing type: Browse...

Name: **AutoWeb** 对应服务的类名

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: **javax.servlet.http.HttpServlet** 自动继承的类 Browse...

Interfaces: Add...
Remove

Template to use: 1] Default template for Servlet

Which method stubs would you like to create?

☐ Inherited abstract methods ☒ doGet() 可选的方法 一般选这四个即可
☐ Constructors from superclass ☒ doPost()
☒ init() and destroy() ☐ doPut()
☐ doDelete() ☐ getServletInfo()

自动配置文件的修改说明

☒ Generate/Map web.xml file

Servlet/JSP Class Name: it.s511.AutoWeb

Servlet/JSP Name: **AutoWeb** 自动完成的配置

Servlet/JSP Mapping URL: /servlet/AutoWeb

File Path of web.xml: /ServletWeb/WebRoot/WEB-INF Browse...

Display Name: 可以不要说明

Description:

创建该类型类，将不在去手动配置web.xml文件，会自动给配置好
同时该类继承了**extends** HttpServlet 需要的包

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

方法说明：

destroy() 服务销毁调用的方法

init() 服务初始化方法

doGet(HttpServletRequest request, HttpServletResponse response)

Get请求的方法处理

doPost(HttpServletRequest request, HttpServletResponse response)

Post请求的方法处理

参数: request 请求对象获取请求数据

response 响应对像向web响应数据

★发起请求的html相关

<!-- 表单对应 的映射路径 及 请求方式get 或 post -->

<form action="servlet/AutoSer" method="get"> 需要注意
的

用户名: <input type="text" name="u_name"></br>

密码: <input type="text" name="u_pwd"></br>

<input type="submit" value="提交">

</form>

web.xml内映射

<servlet-mapping>

<servlet-name>AutoSer</servlet-name>

<url-pattern>/servlet/AutoSer</url-pattern> 需要注意的

</servlet-mapping>

★访问项目下某个html的方法

<http://localhost:8080/AutoServlet/index.html>

http://IP:8080/项目名/xxx.html

在项目的下的WebRoot目录下有一个[index.html](#)

★请求响应处理

GET请求

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //通过请求对象获取请求数据 方法参数一定是要与html里的请求参数名一致
    String name = request.getParameter("u_name");
    String pwd = request.getParameter("u_pwd");
    //get请求方式解决请求乱码
    name = new String(name.getBytes("iso-8859-1"), "utf-8");
    response.setContentType("text/html"); //响应给客户端的数据类型
    response.setCharacterEncoding("utf-8"); //设置响应给客户端的数据 字符编码
    PrintWriter out = response.getWriter();
    out.println("<HTML>");
    out.println("  <HEAD><TITLE>A Servlet</TITLE></HEAD>");
    out.println("  <BODY>");
    out.println("主个是GET请求");
    out.println("name是: " + name);
    out.println("pwd是: " + pwd);
    out.println("  </BODY>");
    out.println("</HTML>");
    out.flush();
    out.close();
}
```

POST请求

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //post请求方法解决请求乱码 必须提前对请求数据的声明编码方式
    request.setCharacterEncoding("utf-8");
    //通过请求对象获取请求数据 方法参数一定是要与html里的请求参数名一致
    String name = request.getParameter("u_name");
    String pwd = request.getParameter("u_pwd");
    response.setContentType("text/html"); //响应给客户端的数据类型
    response.setCharacterEncoding("utf-8"); //设置响应给客户端的数据 字符编码
    PrintWriter out = response.getWriter();
    /**
     *      response.getWriter().write(""); 向客户端响应方式的另外一种写法
     */
    out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");
    out.println("<HTML>");
    out.println("  <HEAD><TITLE>A Servlet</TITLE></HEAD>");
    out.println("  <BODY>");
    out.println("name = " + name);
    out.println("pwd = " + pwd);
    out.println("这是一个 POST 请求方式");
    out.println("  </BODY>");
    out.println("</HTML>");
    out.flush();
    out.close();
}
```



★具体的方法介绍：

★处理请求的方法

//post请求方法解决请求乱码

```
request.setCharacterEncoding("utf-8");
```

//通过请求对象获取请求数据 方法参数一定是要与html里的请求参数名一致

```
String name = request.getParameter("u_name");
```

//get请求方式解决请求乱码(需要对获取的数据进行转码)

```
name = new String(name.getBytes("iso-8859-1"), "utf-8");
```

注意：1、为什么要处理请求数据的字符编码，当客户端请求的数据要不做任何处理的响应给客户端，那么此时就需要对客户请求的数据先做字符编码处理，否则将出现响应给客户端的数据乱码

2、post请求必须先确定请求的数据编码格式，才可以去请求数据

```
request.setCharacterEncoding("utf-8");
```

//通过请求对象获取请求数据 方法参数一定是要与html里的请求参数名一致

```
String name = request.getParameter("u_name");
```

```
String pwd = request.getParameter("u_pwd");
```

★处理响应的方法：

```
ServletOutputStream out = response.getOutputStream();    //得到字节流
```

```
response.setContentType("text/html;charset=UTF-8");    //响应给客户端的数据类型
```

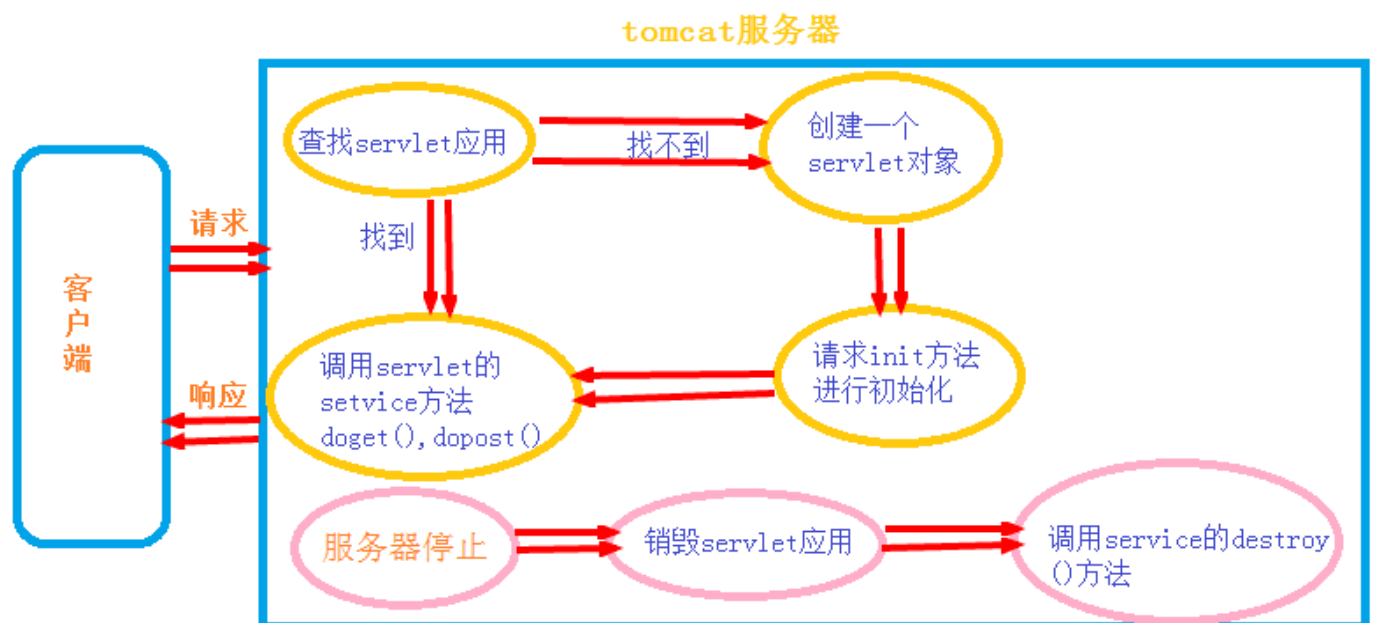
`response.setCharacterEncoding("utf-8");` //设置响应给客户端的数据 字符编码

`PrintWriter out = response.getWriter();` //字符流 向客户端响应方式的另外一种写法

`response.getWriter().write("");` //向客户端响应方式的另外一种写法

★Servlet生命周期

直到服务器停止才会被销毁掉



★文件的上传下载处理

下载实现方式一：（资源所在地与html下载标签不同）

1、创建一个html在里面定义一个下载文件的标签 ★注意

`下载MP4` href是对应处理下载类的web.xml配置的映射路径

2、在项目的WebRoot目录下创建一个新的文件夹res,放资源文件1.mp5 ★注意

3、处理下载代码实现在dopost()方法中实现 doget()调用dopost()方法

3.1、文件下载设置为Content-Disposition类型

```
response.setContentType("Content-Disposition");
```

3.2、找到下载文件的真实路径：根据相对路径，使用容器的getRealPath进行获取

```
String filepath = getServletContext().getRealPath("res/1.mp5");
```

3.3、读取文件流，写入响应对象的输出流

```
FileInputStream fis = new FileInputStream(filepath);
```

3.4、获取响应输出流

```
OutputStream os = response.getOutputStream();
```

3.5、读写流的输出操作

```
int len;  
byte[] b = new byte[1024];  
while((len=fis.read(b))!=-1){  
    os.write(b,0,len);  
}  
fis.close();  
os.close();
```

```

public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doPost(request, response);
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //1、文件下载改为Content-Disposition类型
    response.setContentType("Content-Disposition");
    //2、找到下载文件的真实路径：根据相对路径，使用容器的getRealPath进行获取
    String filePath = getServletContext().getRealPath("res/c5.mp4");

    //3、读取文件流，写入响应对象的输出流
    FileInputStream fis = new FileInputStream(filePath);
    //获取响应输出流
    OutputStream os = response.getOutputStream();
    //流的输出操作
    int len;
    byte[] b = new byte[1024];
    while((len=fis.read(b))!=-1){
        os.write(b,0,len);
    }
    fis.close();
    os.close();
}

```

下载实现方式二：（资源所在地与html下载标签不同）

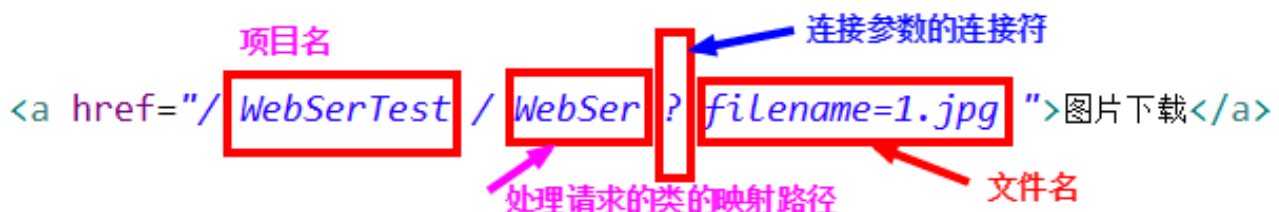
1、创建一个html在里面定义一个下载文件的标签 ★注意

<h1>文件下载</h1>

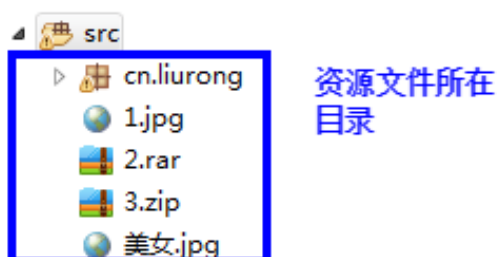
```

<a href="/WebSerTest/WebSer?filename=1.jpg">图片下载</a>
<a href="/WebSerTest/WebSer?filename=美女.jpg">图片下载</a>
<a href="/WebSerTest/WebSer?filename=2.rar">rar文件下载</a>
<a href="/WebSerTest/WebSer?filename=3.zip">zip文件下载</a>

```



2、在项目SRC目录下放资源文件 ★注意



3、处理下载代码实现在doPost()方法中实现 doget()调用doPost()方法

3.1、获得请求参数

```
String filename = request.getParameter("filename");  
filename = new String(filename.getBytes("iso-8859-1"),  
"UTF-8");
```

3.2、处理业务 根据文件名得到文件流

```
InputStream is =  
this.getServletContext().getResourceAsStream("/WEB-INF/classes/"+filename);  
filename = URLEncoder.encode(filename, "UTF-8");  
//filename = new String(filename.getBytes("UTF-8"),"iso-8859-1");//火狐可用
```

3.3、告知浏览器要下载文件

```
response.setHeader("Content-disposition",  
"attachment;filename="+filename);
```

3.4、告知浏览器下载文件的格式

```
String mimeType =  
this.getServletContext().getMimeType(filename);  
response.setHeader("Content-Type", mimeType);
```

3.5、得到输出流

```
ServletOutputStream out = response.getOutputStream();
```

3.6、响应数据

```
int len = 0;  
byte[] b = new byte[1024];  
while((len=is.read(b))!=-1){  
    out.write(b, 0, len);  
}  
out.close();  
is.close();
```

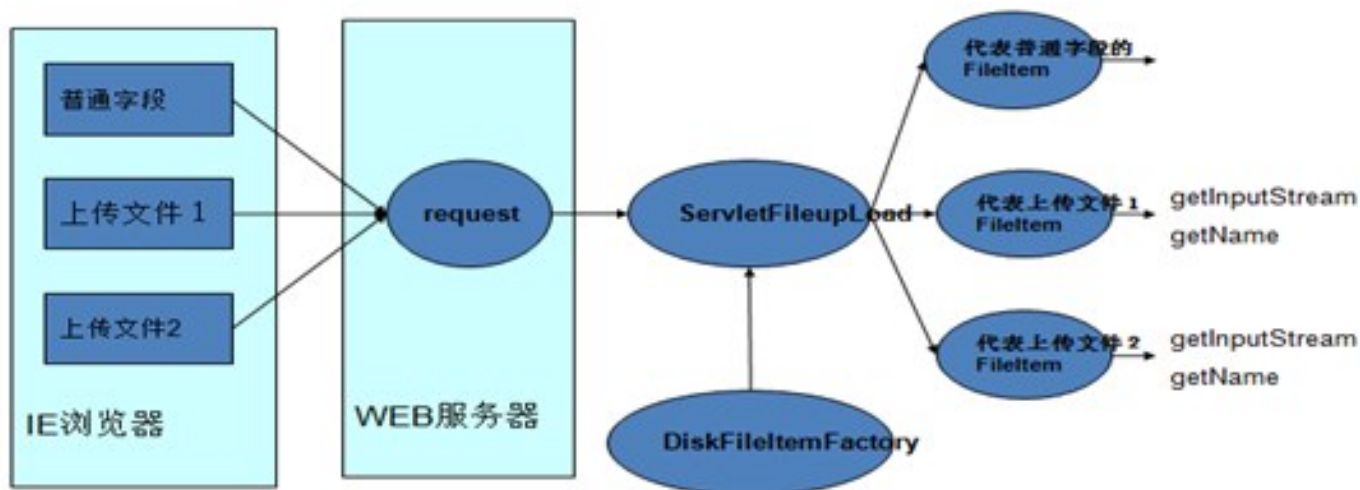
```

String filename = request.getParameter("filename");
filename = new String(filename.getBytes("iso-8859-1"), "UTF-8");
//处理业务
//根据文件名得到文件流
InputStream is = this.getServletContext().getResourceAsStream("/WEB-INF/classes/"+filename);
filename = URLEncoder.encode(filename, "UTF-8"); //A5%E6%F2
//filename = new String(filename.getBytes("UTF-8"), "iso-8859-1"); //火狐可用
//告知浏览器要下载文件
response.setHeader("Content-disposition", "attachment;filename="+filename);
//告知浏览器下载文件的格式
String mimeType = this.getServletContext().getMimeType(filename);
response.setHeader("Content-Type", mimeType);
//得到输出流
ServletOutputStream out = response.getOutputStream();
//响应数据
int len = 0;
byte[] b = new byte[1024];
while((len=is.read(b))!=-1){
    out.write(b, 0, len);
}
out.close();
is.close();

```

上传：

上传原理图



实现的步骤：

1、在项目的WebRoot目录下创建一个新的目录,存放客户上传源文件。

2、HTML里定义一个上传文件的标签 ★注意

```
<form action="servlet/Demo10" method="post"
enctype="multipart/form-data">
```

选择上传文件： <input type="file" name="f">

<input type="submit" value="上传">

```
</form>
```

3、将依赖jar包导入

commons-io-1.3.2.jar **commons-fileupload-1.2.1.jar**

4、处理文件上传类的代码实现：

4.1、创建磁盘文件项工厂

```
DiskFileItemFactory factory = new DiskFileItemFactory();
```

4.2、创建核心上传类，传入文件项工厂对象

```
ServletFileUpload fileUpload = new ServletFileUpload(factory);
```

4.3、判断客户端给出的请求格式是否正确

```
boolean flag = fileUpload.isMultipartContent(request);
```

4.4、如果格式正确，使用核心上传类解析request请求对象，返回的List集合。

```
List<FileItem> list = fileUpload.parseRequest(request);
```

4.5、遍历list集合，得到每个FileItem

```
for (FileItem item : list)
```

4.6、对FileItem判断是普通输入项还是文件上传项（解决如果是客户端直接输入其它数据而不是上传文件）

```
if (item.isFormField()) {
```

如果是普通输入项得到值true；如果是文件上传项得到值false

普通输入项：其实就是客户端随便输入的数据而不是文件项

getFieldName(): 得到普通输入项name的属性的值

getString("UTF-8")：得到普通输入项里面输入的值

```
}
```


4.7、否则就是文件上传项 创建流进行文件上传

文件上传项：

得到通过表单提交过了的文件的输入流，`getInputStream()`

创建输出流，把文件的输入流写到服务器的一个文件中

```
InputStream is = item.getInputStream(); 文件输入流
```

4.8、处理上传到服务器的文件名问题一定要考虑重名问题

`String path = getServletContext().getRealPath("upload");` //保存文件到服务器的目录

`String fileName = UUID.randomUUID().toString();` // 利用工具获得一个不重复的文件名

```
String temp = path + "\\" + fileName;
```

```
FileOutputStream fos = new FileOutputStream(temp); //将文件名给输出流
```

4.9、流操作

```
int len;
```

```
byte[] b = new byte[1024];
```

```
while ((len = is.read(b)) != -1) {
```

```
    fos.write(b, 0, len);
```

```
}
```

```
is.close();
```

```
fos.close();
```

最核心代码截图

```

String result = "";
DiskFileItemFactory factory = new DiskFileItemFactory();
ServletFileUpload fileUpload = new ServletFileUpload(factory);
boolean flag = fileUpload.isMultipartContent(request); // 判断请求格式是否正确
if (flag) {
    try {
        List<FileItem> list = fileUpload.parseRequest(request);
        for (FileItem item : list) {
            if (item.isFormField()) { // 如果是普通字段，就不做处理
            } else { // item.getName(); 说明是流，就进行流操作
                InputStream is = item.getInputStream();
                // 保存文件一定要考虑重名问题
                String path = getServletContext().getRealPath("upload");
                String fileName = UUID.randomUUID().toString(); // 利用工具获得一个不重复的文件名
                String temp = path + "\\ " + fileName;
                FileOutputStream fos = new FileOutputStream(temp);
                int len;
                byte[] b = new byte[1024];
                while ((len = is.read(b)) != -1) {
                    fos.write(b, 0, len);
                }
                is.close();
                fos.close();
                result = "上传成功";
            }
        }
    }
}

```



具体详见附件

转发和重定向

转发和重定向的区别：

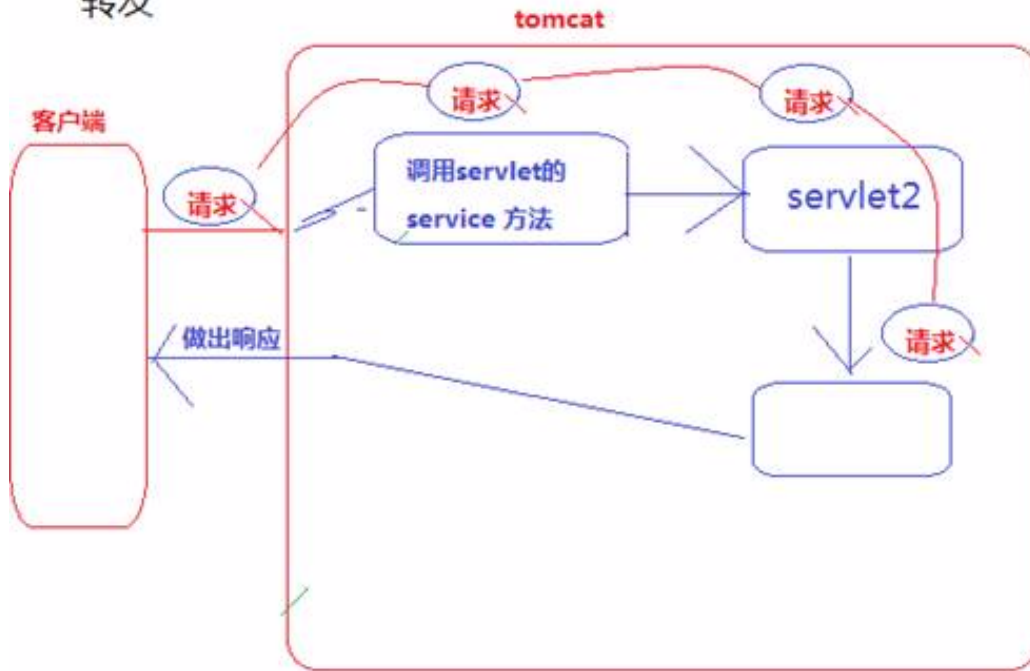
*** 转发：1次请求，服务器行为，地址栏不变，请求域中的数据不丢失**

*** 重定向：2次请求，浏览器行为，地址栏改变，请求域中的数据丢失**

★ 转发：

由于用户名就在请求中，所以我们需要将请一路转发下去

转发



转发请求，本来客户请求是由处理请求的Demo1 servlet类处理，但是Demo1做了其它处理并没有向客户响应，而由Demo2 servlet类向客户端做出响应，当客户端请求时由Demo1将请求的数据交给Demo2去处理，然后响应给客户端。

`getRequestDispatcher("Demo2")` 该方法的作用是转发给哪个类处理或者转发到哪个页面处理

```
request.getRequestDispatcher("Demo2").forward(request, response);
```

★重定向：将响应重新定向给其它servlet类去处理

`response.sendRedirect("/AutoServlet/servlet/Demo");` 该方法的参数是 目标重定向servlet类的全路径 即：/项目名/重定向类的映射路径

详细代码见附件