跟前面一样，建一个项目文件夹ServletPrintToClient，打开在其中建WEB-INF，再打开建classes、lib两个文件目录加一个web.xml文件

任意新建PrintToClient.java

```
import javax.servlet.Servlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.IO.PrintWriter;//标准字符输出流

public class PrintToClientt implements Servlet{

    public void init(ServletConfig config) throws ServletException{

    }

    //五个方法中重点是该方法
    public void service(ServletRequest request,ServletResponse response) throws
IOException,ServletException{

        //有时候响应到浏览器的内容中文会出现乱码，那么用响应对象设置编码方式
        /*
            ServletResponse接口有这样一个方法：
            void setContentType(String type)
          Sets the content type of the response being sent to the client, if the
response has not been committed yet.
            该方法设置内容类型和字符编码方式，不能写错，且要在获取标准流之前设置
        */
        response.setContentType("text/html;charset=HTF-8");

        //想将信息输出到浏览器,就需要使用标准输出流将程序的输出位置更改
        /*
            ServletResponse接口有这样一个方法：
        PrintWriter getWriter()
          Returns a PrintWriter object that can send character text to the
client.
            该方法返回一个将信息输出到客户端的PrintWriter类的标准字符输出流对象
        */
        PrintWriter out = response.getWriter();
        //将HTML字符串输出到浏览器上，Browser解释执行HTML语言
        //这里响应HTML代码到浏览器
        //下面不需要用println()方法，没有必要，用的话只是将HTML字符串换行输出到浏览器，浏览
器看源码
        //的时候HTML源代码是换行的，这样还会占内存，影响效率。
        //但是你想要将里面夹带的文字什么的信息换行，有其他方法
         out.print("<html>");
         out.print("<head>");
         out.print("<title>welcome servlet</title>");
        //具体信息，h1标记
         out.print("<h1 align=\"center\">welcome study servlet</h1>");//该标题下一
信息是自动换行的
        //该输出信息与标题不在同一行
```

```
            out.print("hello");
            out.print("world");
           //换行添加的标签<br>
            out.print("<br>");
            out.print("大家好！");
            out.print("</head>");
            out.print("</html>");

    }

    public void destroy(){

    }

    //本次的项目程序不复杂，这里的方法用到的不多，直接返回null
    public String getServletInfo(){
        return  null;
    }

    //本次的项目程序不复杂，这里的方法用到的不多，直接返回null
    public ServletConfig getServletConfig(){
        return null;
    }

}
```

写好编译，将.class文件放进classes目录中

编写web.xml配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                      http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0">

    <!--这是一个合法的web.xml文件-->
    <!--一个webapp只有一个web.xml文件-->
    <!--web.xml文件主要配置请求路径和实现Servlet的类名之间的绑定关系-->
    <!--web.xml文件在Tomcat服务器启动阶段被解析-->
    <!--web.xml文件解析失败，就会导致服务器启动失败-->
    <!--web.xml文件的标签不能随便编写，Tomcat服务器早就知道该文件中需要编写的标签，该标签也
是SUN制定的规范-->
    <!--直接记住下面的个标签写法和含义-->
    <servlet>
        <!--下面这个servlet-name标签位置的名字随便起,但要与下面同一个标签名一致-->
        <servlet-name>helloServlet</servlet-name>
        <!--下面这个servlet-class标签位置写的是项目中的类名-->
        <servlet-class>PrintToClient</servlet-class>
    </servlet>
    <servlet-mapping>
        <!--下面这个servlet-name标签位置的名字随便起,但要与上面同一个标签名一致-->
        <servlet-name>helloServlet</servlet-name>
        <!--下面这个url-pattern标签位置写的是请求路径，随意编写，但要以/开头，不用添加项目的
名称-->
        <!--该请求路径是虚拟的路径，只是代表一个资源的名称，就是上面类的代号-->
```

```
        <url-pattern>/zzw</url-pattern>
        <!--url-pattern标签还可以编写多个，但都得以/开头且不需要添加项目名-->
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

保存，然后部署项目，启动Tomcat服务器，浏览器输入http://localhost:8080/ServletPrintToClient/zzw