# HTTP协议和tomcat服务器

**【学习目标】理解、了解、应用、记忆**

通过今天的学习，参训学员能够：（解释的时候说出二级目标的掌握程度）

1. **【应用】能够使用ServletContext读取Web工程中资源文件路径**
2. **【理解】能够阐述HttpServletResponse的概念**
3. **【应用】能够独立设置response的状态码**
4. **【应用】能够独立设置response的响应头**
5. **【理解】能够说出文件下载的2种方式**
6. **【应用】能够编写代码完成文件下载**

# ServletContext接口

场景：

在线人数：xxx万，吸引用户来注册。

当用户进入系统，应该当前在线人数加一——servlet处理

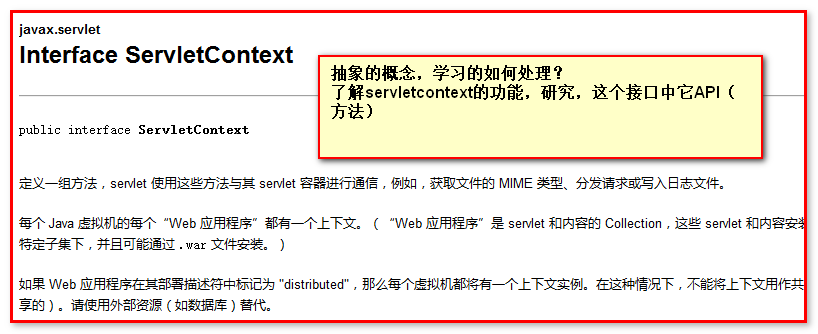
当用户退出系统，应该当前在线人数减一——servlet处理

需求：

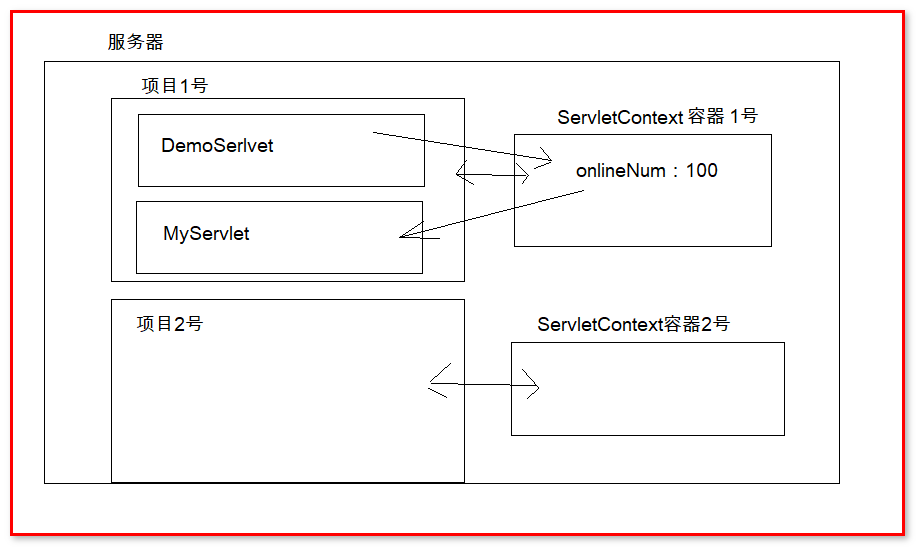
有在线人数的数据，必须，可以让两个servlet都可以获取使用。

希望：有这么一个容器（保存数据），这个容器可以被多个servlet同时使用。

这个就是我们ServletContext容器：



ServletContext示意图：他是一个存储数据的容器



总结：

1 容器可以存储数据

2 操作容器set get remove

3 可以被多个Servlet使用共享。

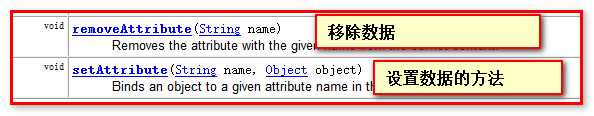
当tomcat启动时，会为每个web应用创建一个唯一的ServletContext对象代表当前Web应用.该对象不仅封装了当前web应用的所有信息，而且实现了多个servlet的数据共享.

在每个项目中可以有多个Servlet程序，每个Servlet程序都是独立的。Servlet中的配置信息可以使用ServletConfig获取，而当前这个项目的配置信息，就必须使用描述这个项目的ServletContext对象获取。

## ServletContext容器的存取删操作（重点）

由于一个web应用程序中的所有Servlet共享同一个ServletContxt对象,因此ServletContext中的属性可以被Web应用程序中的所有servlet访问。在ServletContext中分别定义了用于 增加、删除、获取ServletContext属性的4个方法：

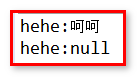




代码演示：

|  |
| --- |
| **package** cn.igeek.web;  **import** java.io.IOException;  **import** java.lang.reflect.Field;  **import** java.util.Map;  **import** java.util.Set;  **import** javax.servlet.ServletContext;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** MyServlet **extends** HttpServlet {  **public** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  ServletContext servletContext = getServletContext();  servletContext.setAttribute("hehe", "呵呵");  String hehe = (String) servletContext.getAttribute("hehe");  System.***out***.println("hehe:"+hehe);  servletContext.removeAttribute("hehe");  String hehe2 = (String) servletContext.getAttribute("hehe");  System.***out***.println("hehe:"+hehe2);  }  **public** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  doGet(request, response);  }  } |

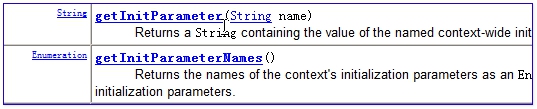
效果：



## 读取全局（整个项目）配置参数（读取项目在web.xml中的配置信息）

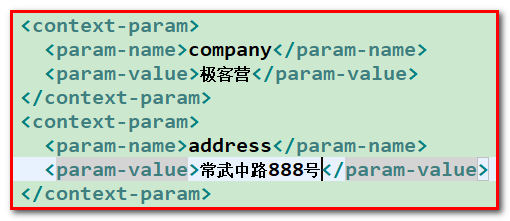
我们在web.xml文件中是可以配置项目的初始化信息:

项目的配置信息需要在当前这个web.xml的全局位置配置：

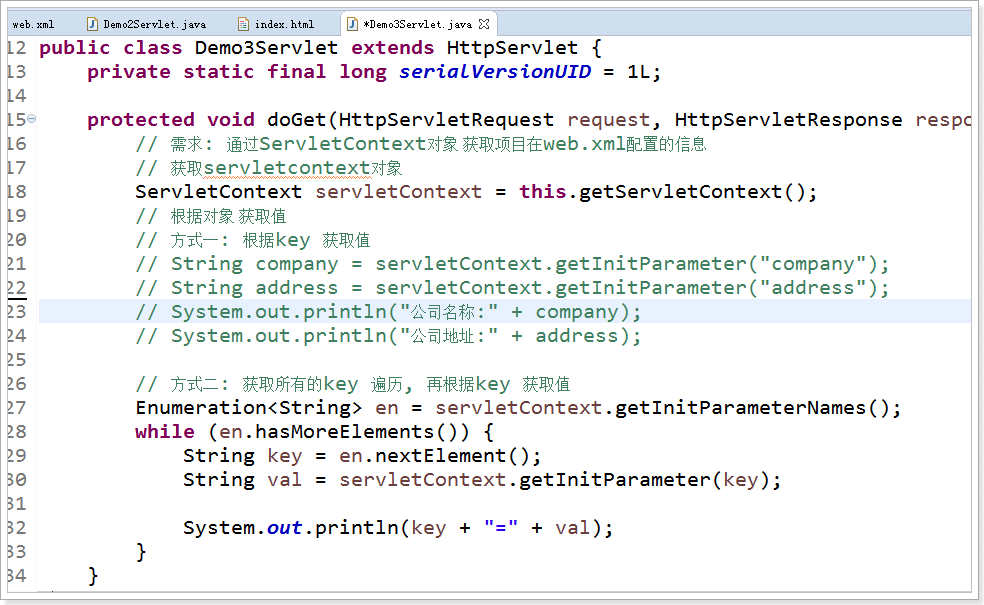


需求: 设置项目全局参数 公司=极客营 地址=常武中路888号

1 在web.xml中配置全局（项目）参数信息



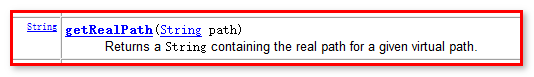
2 根据key获取value



## ServletContext读取Web工程中资源文件路径

要读取资源，必须先找到这个文件：

第一个要介绍的方法是：



|  |
| --- |
| String realPath = context.getRealPath("/WEB-INF/classes/c3p0-config.xml");  System.***out***.println(realPath); |

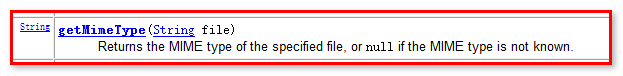
效果：

|  |
| --- |
| C:\apache-tomcat-7.0.81\webapps\aa\WEB-INF\classes\c3p0-config.xml |

使用IO读取展示到控制台：

|  |
| --- |
| BufferedReader reader = **new** BufferedReader(**new** FileReader(**new** File(realPath)));  String line = **null**;  **while**((line = reader.readLine()) != **null**){  System.***out***.println(line);  } |

解析获取到的资源的文件类型：



|  |
| --- |
| ServletContext context = getServletContext();    String realPath = context.getRealPath("/WEB-INF/classes/c3p0-config.xml");  System.***out***.println(context.getMimeType(realPath));    String realPath2 = context.getRealPath("/1.html");  System.***out***.println(context.getMimeType(realPath2)); |

效果：

|  |
| --- |
| application/xml  text/html |

# response介绍（服务器 ——》客户端浏览器）

在之前学习servlet的时候，我们已经可以处理请求和响应，针对请求和响应，我们servlet方法中提供了request对象和response对象，方便我们使用。

今天，我们主要来学习和使用response对象：



ServletResponse定义了基本的响应数据的方法，但是不面对任何协议。



## response的状态码和响应头设置

设置状态码:

IMG_258

int sc 是状态码（200,302,404,500），不要设置不存在的响应状态码

默认是200;

设置响应头信息:

IMG_259

name : 表示响应头 name的值

value : 响应头对应的值

## 常用的响应头演示：

### Refresh: 3;url=http://www.jd.com （页面自动刷新）

 3 表示是的时间，3秒

url 地址

到3秒后，页面刷新url

测试代码：

|  |
| --- |
| response.setHeader("Refresh", "5;url=http://www.jd.com"); |

### Location: http://www.it315.org/index.jsp   //  通常告知浏览器 马上向该地址发送请求 通常 和 302 一起使用!!

（重点） 结合302完成重定向 操作 Location重定向后地址

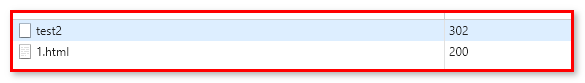
代码演示一：

|  |
| --- |
| response.setStatus(302);  response.setHeader("Location", " http://localhost:9090/day02/1.html "); |

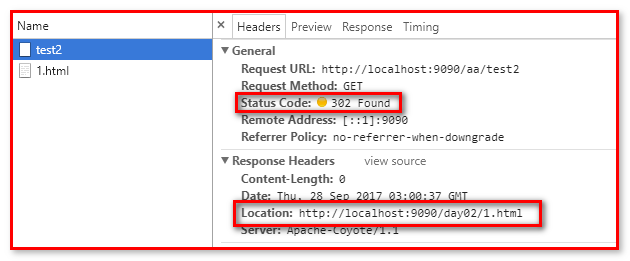
代码演示二：

|  |
| --- |
| response.sendRedirect("http://localhost:9090/day02/1.html "); |

重定向的细节：

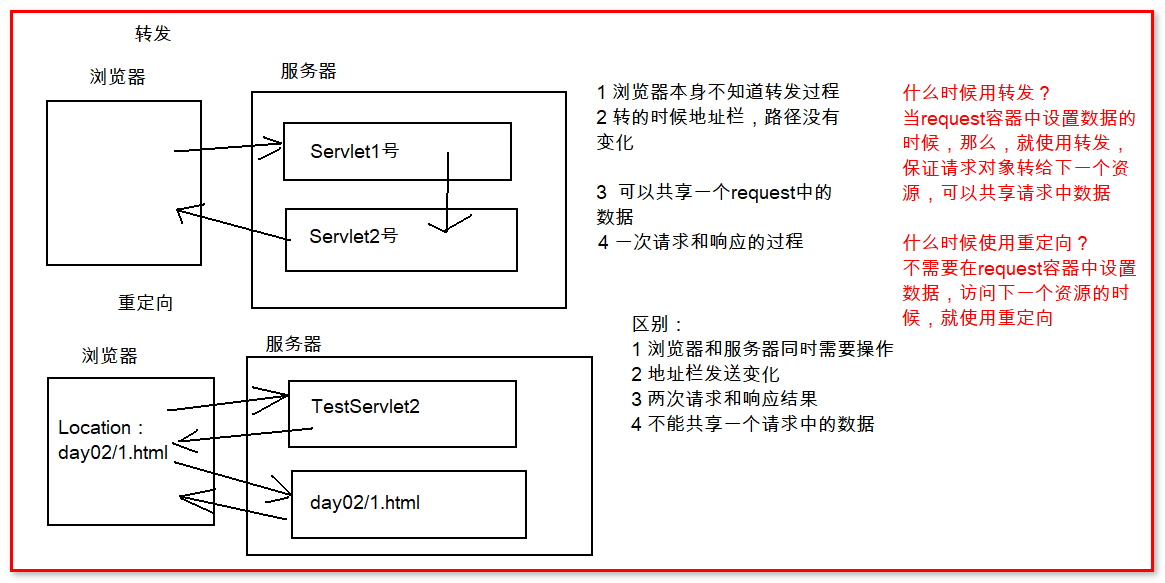


浏览器发送了两次请求，第一次请求中：



指定了让浏览器再次发送请求的地址

请求转发和重定向的区别:



重定向和转发的区别：

1 转发操作发生在服务器内部，重定向是浏览器执行操作

2 转发地址栏不变，重定向，地址栏变化

3 转发在一次请求中完成，重定向是两次请求

4 转发可以在一次请求中共享数据，重定向不行。

### Content-Type: text/html; charset=utf-8   告诉浏览器以什么样的方式打开资源，和使用编码集

代码演示：

|  |
| --- |
| response.setContentType("text/html;charset=utf-8");  response.getWriter().write("<h1>哈哈哈<h1>"); |

效果：



如果设置普通文本形式：text/plain

|  |
| --- |
| response.setContentType("text/plain;charset=utf-8");  response.getWriter().write("<h1>哈哈哈<h1>"); |

效果：



## 设置响应体

应用场景: 浏览器显示的内容, 下载文件等.

response 以流的形式,将内容输出给浏览器

response.getWriter() 获取响应体的字符输出流，输出页面的时候使用

response.getOutputStream() 获取响应体的字节输出流，输出文件（图片，avi等。。。。）的时候使用

通过这两个流可以向浏览器输出 响应体内容.

注意问题:

1 文件数据传输: getOutputStream    手动生成响应内容时 使用 getWriter

2 getOutputStream 和 getWriter 相互排斥,不能同时使用,否则会冲突.

3 tomcat会自动调用response输出流的 close方法和flush方法, 因此不需要我们手动关闭流.

### 需求: 向浏览器输出一个百度的超链接 的html页面

public class Response5Servlet extends HttpServlet {

private static final long serialVersionUID = 1L;

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

// 需求: 向浏览器输出一个百度的超链接 的html页面

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

response.getWriter().println("<html>");

response.getWriter().println("<head><title>百度超链接</title></head>");

response.getWriter().println("<body>");

response.getWriter().println("<a href='http://www.baidu.com'>百度</a>");

response.getWriter().println("</body>");

response.getWriter().println("</html>");

}

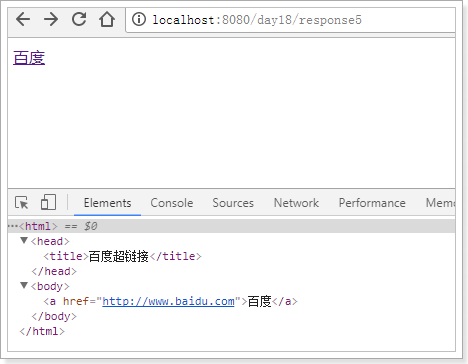
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

}

页面效果:



## 下载介绍

### 下载方式介绍

文件的下载有2种方式可以完成：

1、超链接形式：

<a href=”资源的路径” >资源名称</a>

问题：

1 中文文件名乱码

2 图片和文本直接解析而不是下载

3 只有rar压缩文件，才直接下载

2、使用Servlet通知浏览器下载：

1 修改请求路径，将要下载的文件名称，发送给服务器的servlet，让servlet进行处理

2 加载服务器上的文件资源

3 提示浏览器，以下载的方式，获取服务器资源

4 使用IO的方式，将文件数据输出到浏览器（response.getOutputStream();）

### 服务器端写程序下载（重点：必须掌握）

package cn.igeek.web;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.net.URLEncoder;

import javax.servlet.ServletException;

import javax.servlet.ServletOutputStream;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import sun.misc.BASE64Encoder;

public class DownServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// 1）浏览器发送请求给服务器（用户要下载的文件名）

String fileName = request.getParameter("fileName");

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

// 2）服务器加载用户要下载的文件数据。

String realPath = this.getServletContext().getRealPath("/upload");

File file = new File(realPath,fileName);

// 3）通知浏览器以下载的方式请求资源

//a.Content-Type 设置文件媒体格式

response.setContentType(this.getServletContext().getMimeType(fileName));

//处理中文文件名乱码

String header = request.getHeader("User-Agent");

if(header.contains("Firefox")){

//表示当前是火狐

BASE64Encoder base64Encoder = new BASE64Encoder();

fileName = "=?utf-8?B?" + base64Encoder.encode(fileName.getBytes("utf-8")) + "?=";

}else{

//谷歌

fileName = URLEncoder.*encode*(fileName,"utf-8");

}

//b.Content-Disposition 设置要被下载的文件名

response.setHeader("Content-Disposition", "attachment;filename=" + fileName);

// 4）使用IO技术，将数据发送（使用response对象发送数据）

//a.获取输入流

FileInputStream in = new FileInputStream(file);

//b.获取输出流

ServletOutputStream out = response.getOutputStream();

//缓冲区

byte[] buf = new byte[8192];

int len = 0;

while(( len = in.read(buf))!=-1){

out.write(buf, 0, len);

}

in.close();

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

响应消息头设置：

Content-Type 设置文件媒体格式

response.setContentType(getServletContext().getMimeType(filename));

Content-Disposition 设置要被下载的文件名

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

谷歌浏览器（和主流其他浏览器）中文乱码处理：

fileName = URLEncoder.encode(fileName,"utf-8");

火狐下载文件名乱码处理：（先判断是否是火狐浏览器）

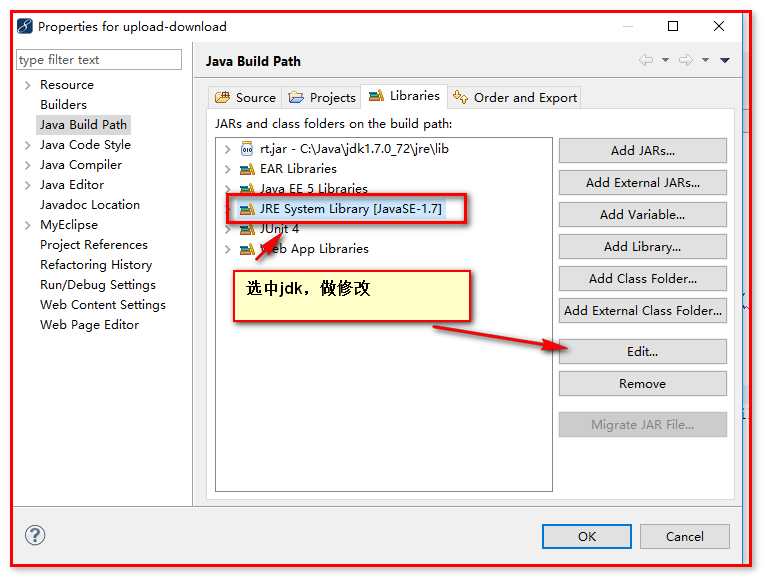
BASE64Encoder base64Encoder = new BASE64Encoder();

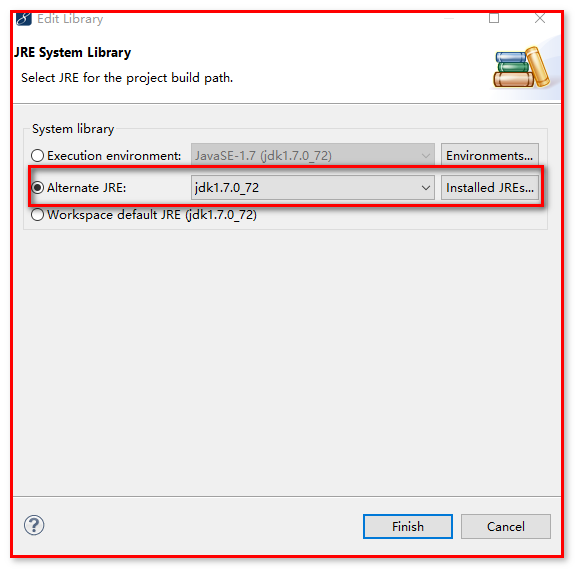
filename = "=?utf-8?B?" + base64Encoder.encode(filename.getBytes("utf-8")) + "?=";

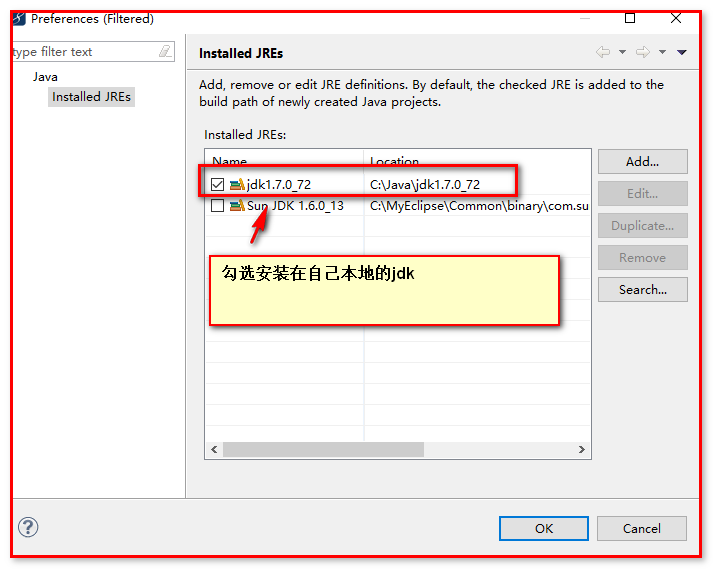
注意：如果MyEclipse10.0无法使用BASE64Encoder这个类（无法导入这个包），按一下方式解决。

解决方法：按照如下方法设置Eclipse导入%JAVA\_HOME%\jre\lib目录下的rt.jar包即可，Project->Properties,选择Java Build Path设置项，再选择Libraries标签，Add External Jars添加%JAVA\_HOME%\jre\lib\rt.jar就可以使用

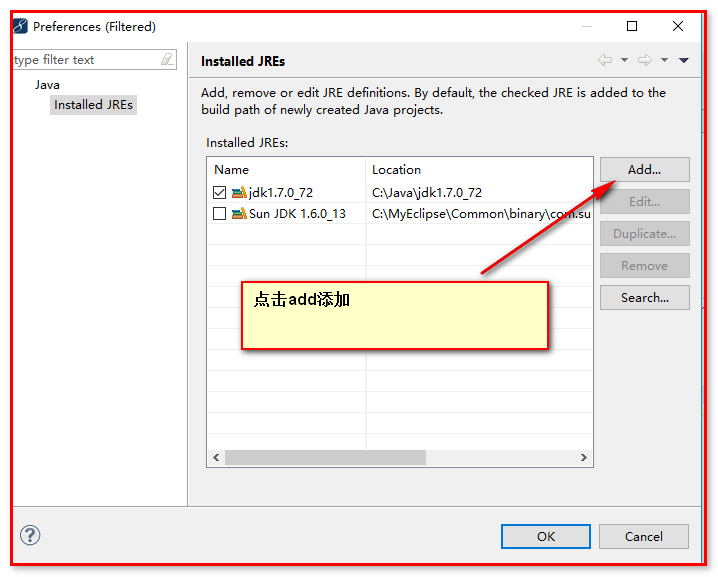
如果还是不行，可以再修改项目的JDK，将JDK版本改成本地安装的JDK版本

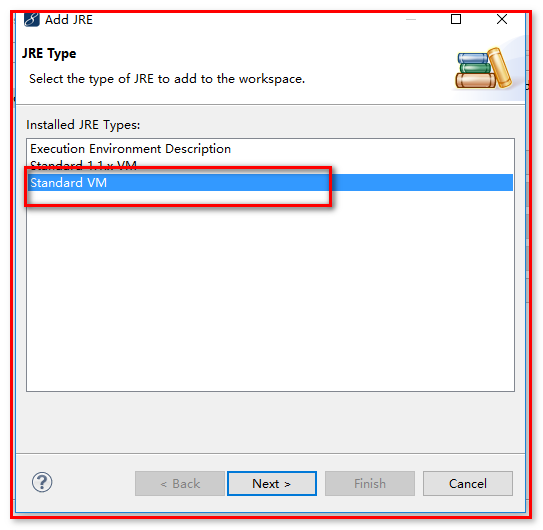


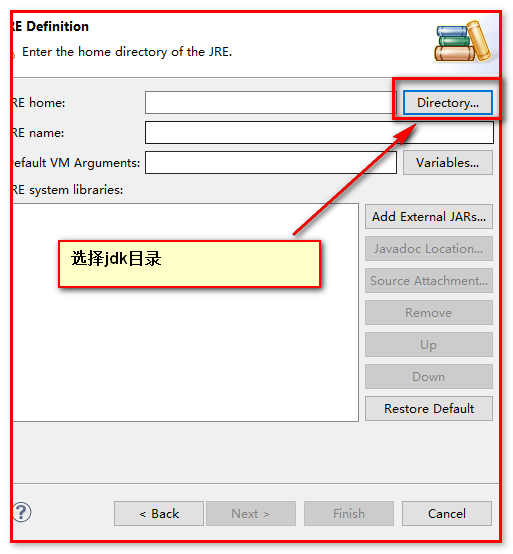


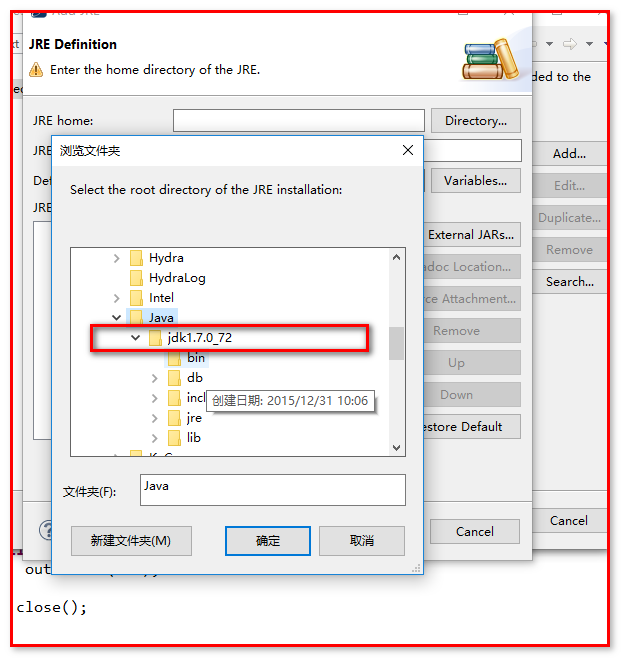


如果没有自己的JDK：



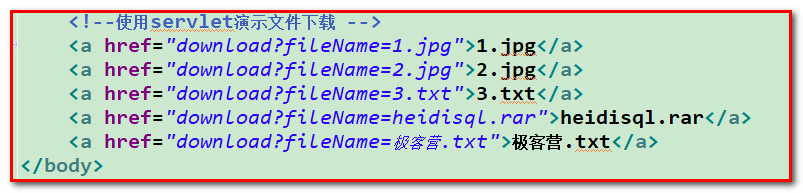






小结：

1. 先修改页面——发送要下载的文件的文件名



1. 定位要被下载的文件的位置（获取文件名的时候，处理中文乱码问题）
2. 设置文件的媒体格式
3. 解决中文文件名问题(先要判断浏览器，根据不同的浏览器，不同编码处理)
4. 设置要被下载的文件名
5. 使用IO技术将数据发送给浏览器（response获取的out输出流，不用手动关闭，自己使用的输入流，需要手动关闭）

附录：学员问题解答

问题1 ：ServletContext的getContext(),getContextPath(),getRealPath() 以及HttpServletRequst的getContextPath(),getRequestURI() getRequestURL() getServletPath() 这些方法获得路径有什么区别和联系

答： ServletContext

getContext() ：获取指定路径的项目，它的ServletContext对象，但是，处于安全考虑，如果获取其他的项目ServletContext都会拿到null，所以方法不常用

getContextPath()：获取当前项目根路径

getRealPath()：获取当前项目中，指定资源的路径

HttpServletRequst：

getContextPath()：获取当前项目根路径（同上）

getRequestURI()： 获取请求地址，不包含协议，地址，端口

getRequestURL()： 获取请求地址，包含协议，地址，端口

问题2 ：ServletContext 与Servlet的在服务器中存在形式被称为什么？ 使用ServletContext会出现并发问题么？如何解决

答：1 对象

ServletContext tomcat在底层的实现类是org.apache.catalina.core.ApplicationContextFacade，这是一个容器存取数据的

Servlet 是处理请求和响应的对象，由我们自己实现

2 ServletContext对象在存取数据的时候使用ConcurrentHashMap，线程安全。