# Java Web（1）

## Response（PROJECT: LearnResponse）

设置状态码和状态信息

1. 重定向

\*设置location

\*设置状态码302

1. 定时刷新

\*response.setHeader("Refresh","5;URL=/servletE");//5秒后自动跳转,前页可以添加提示信息

## Request（PROJECT: LearnRequest）

### 格式

请求行

请求头

空行

请求体（GET没体）

### 获取常用信息

1. 请求客户端IP

Request.getRemoteAddr()

1. 请求方式

getMethod()

结果为POST或者GET

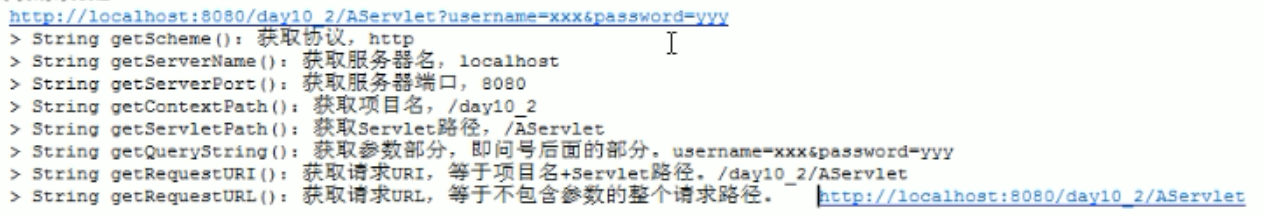
### 获取请求头

1. \*\*\*\*String getHeader(String Name),适用于单值头；
2. Int GetIntHeader(String name),适用于单值int类型请求头；
3. Enumeration<String> getHeaderNames()，使用于多值请求头。

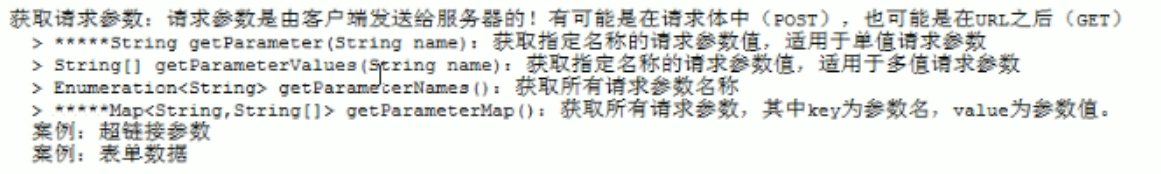
### 案例

1. 通过User-Agent识别用户浏览器类型；
2. 防盗链：如果请求不是通过本站的超链接发出的，发送错误代码404（获取头中的Referer字段，表示请求的来源）。

### 获取请求URL



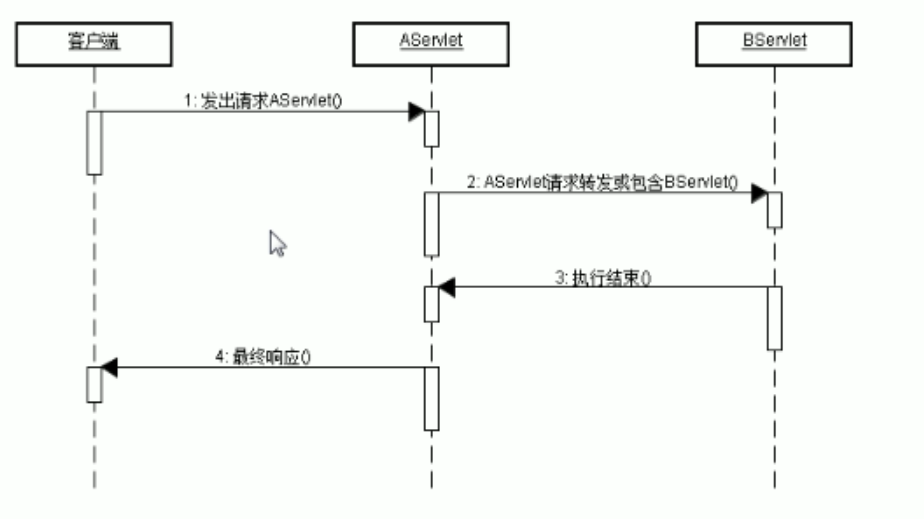
### 获取请求参数





### 请求转发和请求包含

1. 示意图：



1. Request域：

在多个request之间共享数据，在Aservlet中存数据，在Bservlet中使用。

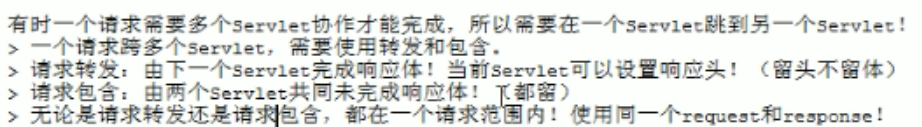
\*RequestDispatcher rd = request.get RequestDispatcher(“/servletD”);//获取RequestDispatcher对象，方法的参数是servlet的路径

请求转发：rd.forword(request,response);（重点）

请求包含：rd.include(request,response);

\*注意：request和response参数总是成对出现。

\*重定向是两个请求，而请求转发或请求包含仅一个请求。可以通过观察地址栏发现。



\*方法

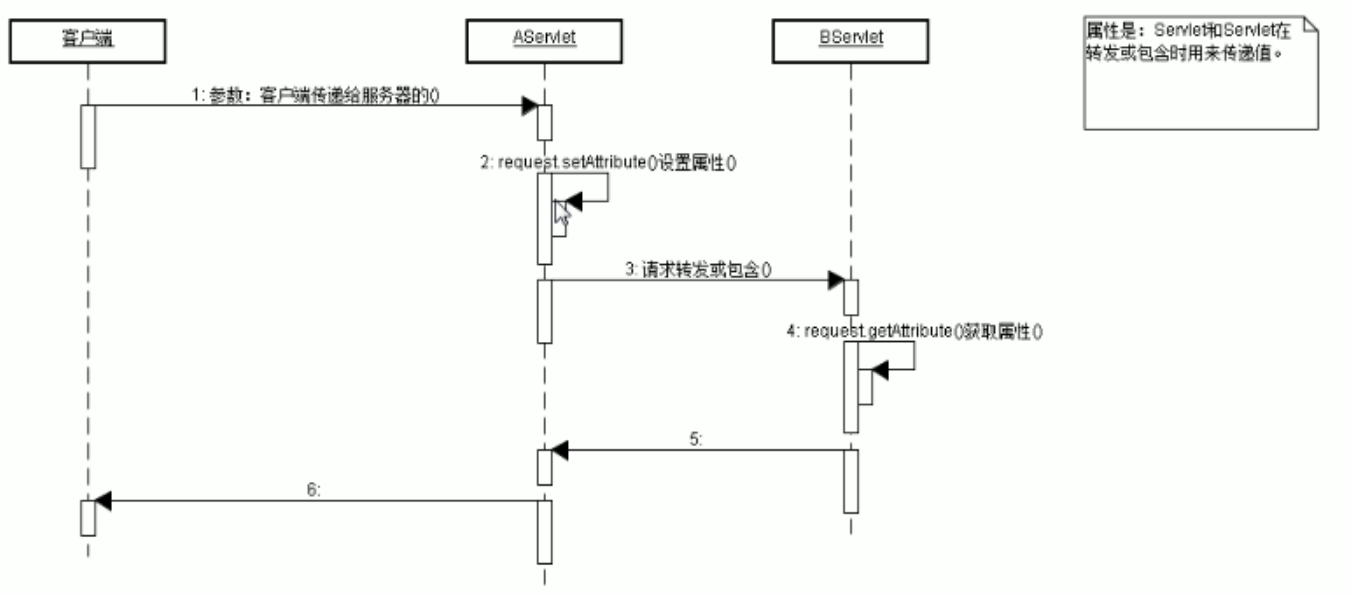
\*\*void setAttribute(String name,Object value);

\*\*Object getAttribute(String name);

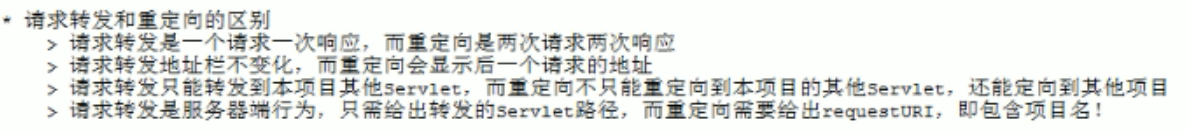
\*\*void removeAttribute(String name);

\*\*\*Servlet中三大域对象：request,session,application都有以上三种方法。

\*Request域示意图：



\*请求转发和重定向之间的区别：

\*\*请求转发和重定向效率是转发高！因为是一个请求。

<>若需要地址栏的发生变化，那么必须使用重定向；

<>若需要在下一个servlet中获取request域中的数据，必须要使用转发。

## 编码

### 常见的字符编码

\*iso-8859-1（不支持中文）

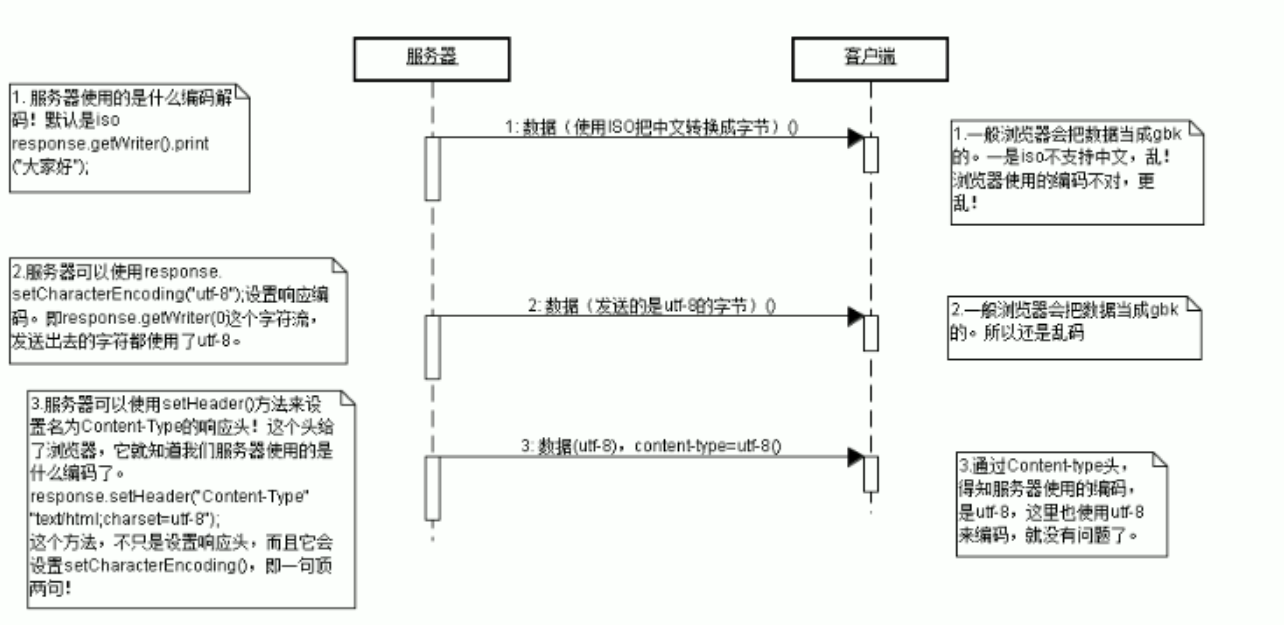
\*gbk（系统默认编码，中国的国标码）

\*utf-8（万国码，支持全世界的编码，一般使用此编码）

## 服务器端的设置

*/\*\*  
 \* 设置编码  
 \* 1.先设置服务器使用的编码  
 \* 2.再告诉浏览器应该使用什么编码  
 \* \*/  
//先设置服务器使用的编码*response.setCharacterEncoding(**"utf-8"**);  
*//再告诉浏览器应该使用什么编码，此方法调用第一个方法*response.setContentType(**"text/html;charset=UTF-8"**);

## 服务器端-客户端编码示意图



\*总结：

在使用getWriter()方法之前，先调用下面的方法：

Response.setContentType(“text/html;charset=utf-8”);就不会乱码。

### 请求编码

\*POST

在request.getParameter()之前调用方法：

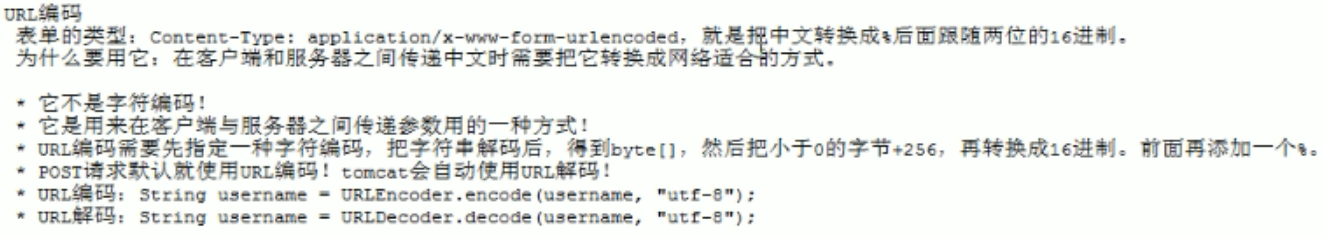
request.setCharacterEncoding(**"utf-8"**);

\*GET

在Tomcat的server.xml中添加配置：



### URL编码



\*代码示例

**public class** testURLEncoding {  
 @Test  
 **public void** fun() **throws** UnsupportedEncodingException {  
 String s = **"乔晓斌"**;*//%E5%BC%A0%E4%B8%89  
// byte[] bytes = s.getBytes("utf-8");  
// System.out.println(Arrays.toString(bytes));  
 /\*\*\*  
 \*得到URL编码  
 \*/* String urlencoding = URLEncoder.*encode*(s,**"UTF-8"**);  
 System.***out***.println(urlencoding);  
 */\*\*  
 \* URL解码  
 \* \*/* String urldecoding = URLDecoder.*decode*(urlencoding,**"UTF-8"**);  
 System.***out***.println(urldecoding);  
  
  
 }  
}

\*注意：GET请求时，需要把链接中的中文参数，使用URL来编码，通过在JSP中添加JAVA代码来实现。

## 路径



### 示例代码

**import** org.apache.commons.io.IOUtils;  
**import** org.junit.jupiter.api.Test;  
  
**import** java.io.InputStream;  
  
*/\*\*  
 \* 演示ClassLoader和Class获取类资源路径  
 \* \*/***public class** testClassLoader {  
 */\*  
 \* 使用类加载器  
 \* \*/* @Test  
 **public void** fun1() **throws** Exception{  
 */\*  
 \* ClassLoader获取资源时，不能以"/"开头  
 \* \*/  
 //得到类加载器* ClassLoader classLoader = testURLEncoding.**class**.getClassLoader();  
 *//让类加载器去类路径下查找资源* InputStream resourceAsStream = classLoader.getResourceAsStream(**"123.txt"**);  
 System.***out***.println(IOUtils.*toString*(resourceAsStream));  
 }  
  
 */\*  
 \* 与类加载器的效果相同（加"/"）  
 \* \*/* @Test  
 **public void** fun2() **throws** Exception{  
 Class c = testClassLoader.**class**;  
 InputStream resourceAsStream = c.getResourceAsStream(**"/123.txt"**);  
 System.***out***.println(IOUtils.*toString*(resourceAsStream));  
  
 }  
  
 */\*  
 \* 到当前Class对应的.class文件所在目录去加载（不加"/"）  
 \* \*/* @Test  
 **public void** fun3() **throws** Exception{  
 Class c = testClassLoader.**class**;  
 InputStream resourceAsStream = c.getResourceAsStream(**"123.txt"**);  
 System.***out***.println(IOUtils.*toString*(resourceAsStream));  
  
 }  
  
  
}