

## 课程介绍:

通过前面的学习我们能够创建并运行一个简单的 *Servlet* 程序，完成浏览器和服务器的简单交互，但是我们编写的 *Servlet* 代码是没有对浏览器的请求进行处理的。现在我们开始学习如何使用 *Servlet* 进行浏览器请求的处理。

## *Request* 对象:

问题:

浏览器发起请求到服务器,会遵循 *HTTP* 协议将请求数据发送给服务器。

那么服务器接受到请求的数据该怎么存储呢?不但要存,而且要保证完成性。

解决:

使用对象进行存储,服务器每接受一个请求,就创建一个对象专门的存储此次请求的请求数据。

实现:

*request* 对象

解释:

服务器接收到浏览器的请求后,会创建一个 *Request* 对象,对象中

存储了此次请求相关的请求数据。服务器在调用 *Servlet* 时会将创建的

*Request* 对象作为实参传递给 *Servlet* 的方法,比如: *service* 方法。

使用:

- 获取请求头数据
- 获取请求行数据
- 获取用户数据

## Response 对象:

问题:

在使用 *Request* 对象获取了请求数据并进行处理后, 处理的结果如何显示到浏览器中呢?

解决:

使用 *Response* 对象

解释:

服务器在调用指定的 *Servlet* 进行请求处理的时候, 会给 *Servlet* 的方法传递两个实参 *request* 和 *response*。其中 *request* 中封存了请求相关的请求数据, 而 *response* 则是用来进行响应的一个对象。

使用:

设置响应头

设置响应编码格式

设置响应实体

## 请求乱码问题解决:

使用 *String* 进行重新编码:

```
uname=new String(uname.getBytes("iso8859-1"),"utf-8");
```

*Get* 方式请求:

在 *service* 方法中使用: `req.setCharacterEncoding("utf-8");`

在 *tomcat* 服务器目录下的 *conf* 文件下找到 *server.xml* 文件, 打开进

行如下配置:

```
<Connector port="8080" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443" useBodyEncodingForURI="true"/>
```

*Post* 方式请求:

在 *service* 方法中使用: `req.setCharacterEncoding("utf-8");`

## 流程总结:

*Servlet* 的使用流程:

- 设置请求编码格式
- 设置响应编码格式
- 获取请求信息
- 处理请求信息
- 响应处理结果

数据流转流程:

浏览器----->服务器----->数据库

浏览器<-----服务器<-----数据库

## 请求转发：

问题：

服务器在接收到浏览器的请求后，仅仅使用一个 *Servlet* 进行请求处理，会造成不同的 *Servlet* 逻辑代码冗余，*Servlet* 的职责不明确。

解决：

使用请求转发。

特点：

一次请求

地址栏信息不改变。

## Request 对象作用域

问题：使用请求转发后，不同的 *Servlet* 之间怎么进行数据的共享呢？或者说数据怎么从一个 *Servlet* 流转给另外一个 *Servlet* 呢？

解决：使用 *request* 对象的作用域

使用：

```
request.setAttribute(object name,Object value);
```

```
request.getAttribute(Object obj)
```

作用：解决了一次请求内的不同 *Servlet* 的数据(请求数据+其他数据)共享问题。

作用域：基于请求转发，一次请求中的所有 Servlet 共享。

注意：

使用 Request 对象进行数据流转，数据只在一次请求内有效。

特点：

服务器创建

每次请求都会创建

生命周期一次请求

## 重定向

问题：

如果当前的请求，Servlet 无法进行处理怎么办？

如果使用请求转发，造成表单数据重复提交怎么办？

解决：

使用重定向

使用：

`response.sendRedirect("路径")`。

本地路径为：uri

网络路径为：定向资源的 URL 信息

特点：

两次请求

浏览器地址栏信息改变

避免表单重复提交

