07-Servlet2

讲师：王振国

今日任务

1.HttpServletRequest 类

a)HttpServletRequest 类有什么作用。

每次只要有请求进入 Tomcat 服务器，Tomcat 服务器就会把请求过来的 HTTP 协议信息解析好封装到 Request 对象中。然后传递到 service 方法（doGet 和 doPost）中给我们使用。我们可以通过 HttpServletRequest 对象，获取到所有请求的信息。

b)HttpServletRequest 类的常用方法

i.	getRequestURI()	获取请求的资源路径 仅仅是资源地址
ii.	getRequestURL()	获取请求的统一资源定位符（绝对路径）包括服务器ip的所有地址
iii.	getRemoteHost()	获取客户端的 ip 地址
iv.	getHeader()	获取请求头
v.	getParameter()	获取请求的参数 前端发送了参数，通过该方法获取参数
vi.	getParameterValues()	获取请求的参数（多个值的时候使用）
vii.	getMethod()	获取请求的方式 GET 或 POST
viii.	setAttribute(key, value);	设置域数据
ix.	getAttribute(key);	获取域数据
x.	getRequestDispatcher()	获取请求转发对象

常用 API 示例代码：

```
public class RequestAPIServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//        i.getRequestURI()                获取请求的资源路径
        System.out.println("URI => " + req.getRequestURI());
//        ii.getRequestURL()               获取请求的统一资源定位符（绝对路径）
        System.out.println("URL => " + req.getRequestURL());
```

```
//        iii.getRemoteHost()          获取客户端的 ip 地址
        /**                                                    注意问题：
         * 在 IDEA 中，使用 localhost 访问时，得到的客户端 ip 地址是 ===>>> 127.0.0.1<br/>
         * 在 IDEA 中，使用 127.0.0.1 访问时，得到的客户端 ip 地址是 ===>>> 127.0.0.1<br/>
         * 在 IDEA 中，使用 真实 ip 访问时，得到的客户端 ip 地址是 ===>>> 真实的客户端 ip 地址<br/>
         */
        System.out.println("客户端 ip 地址 => " + req.getRemoteHost());
//        iv.getHeader()                  获取请求头
        System.out.println("请求头 User-Agent ==>> " + req.getHeader("User-Agent"));
//        vii.getMethod()             获取请求的方式 GET 或 POST
        System.out.println( "请求的方式 ==>> " + req.getMethod() );
    }
}
```

c)如何获取请求参数

表单：　　通过HttpServletRequest直接获取请求的参数

```
<body>
    <form action="http://localhost:8080/07_servlet/parameterServlet" method="get">
        用户名：<input type="text" name="username"><br/>
        密码：<input type="password" name="password"><br/>
        兴趣爱好：<input type="checkbox" name="hobby" value="cpp">C++
        <input type="checkbox" name="hobby" value="java">Java
        <input type="checkbox" name="hobby" value="js">JavaScript<br/>
        <input type="submit">
    </form>
</body>
```

Java 代码：

```
public class ParameterServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        // 获取请求参数
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        String[] hobby = req.getParameterValues("hobby");

        System.out.println("用户名：" + username);
        System.out.println("密码：" + password);
        System.out.println("兴趣爱好：" + Arrays.asList(hobby));
```

```
    }
}
```

doGet 请求的中文乱码解决：

```
// 获取请求参数
String username = req.getParameter("username");

//1 先以 iso8859-1 进行编码
//2 再以 utf-8 进行解码
username = new String(username.getBytes("iso-8859-1"), "UTF-8");
```

d)POST 请求的中文乱码解决

设置请求体的字符集为utf-8，需要在获取请求参数之前调用

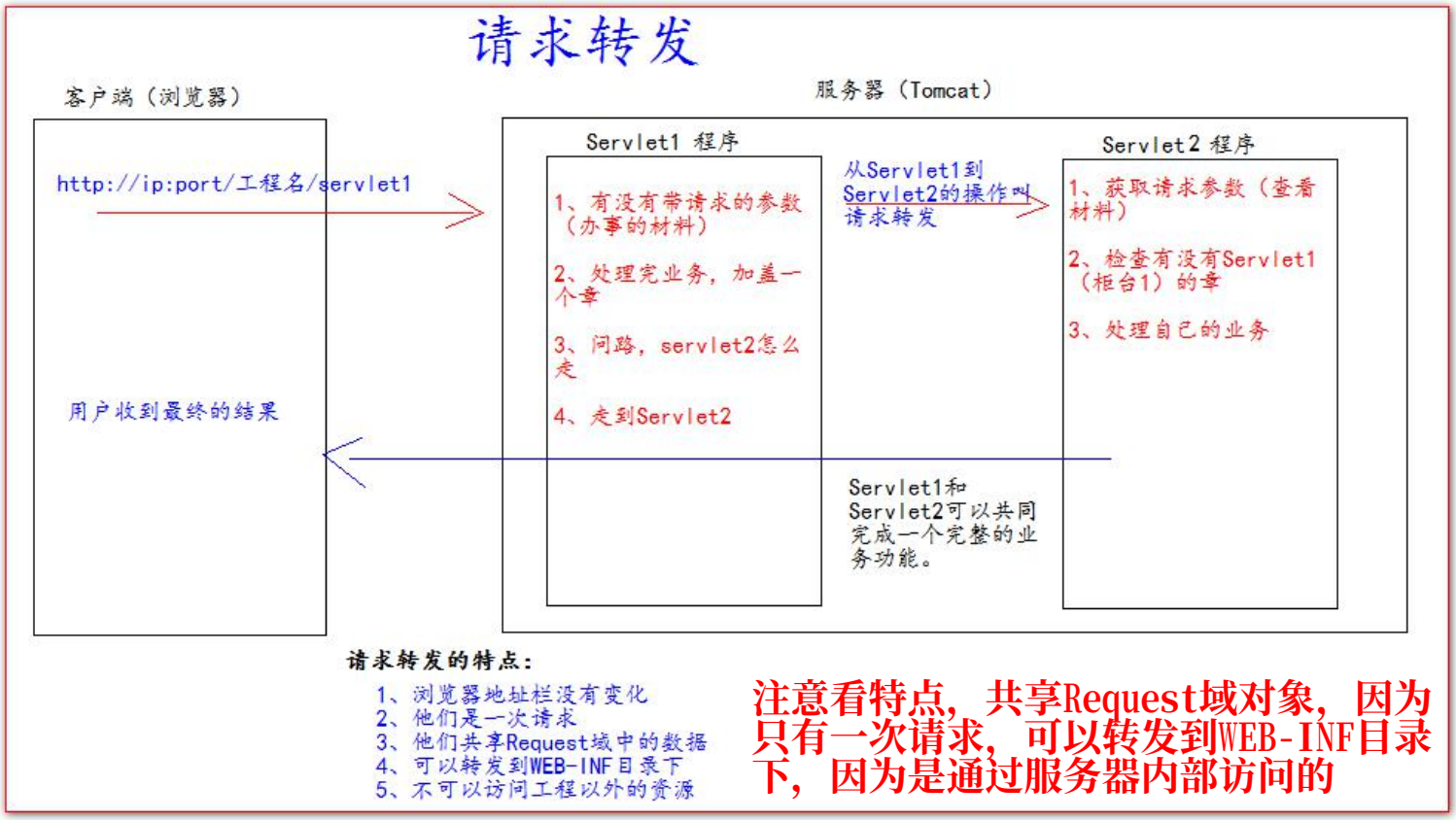
```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
    // 设置请求体的字符集为 UTF-8，从而解决 post 请求的中文乱码问题
    req.setCharacterEncoding("UTF-8");
    System.out.println("-------------doPost------------");
    // 获取请求参数
    String username = req.getParameter("username");
    String password = req.getParameter("password");
    String[] hobby = req.getParameterValues("hobby");

    System.out.println("用户名：" + username);
    System.out.println("密码：" + password);
    System.out.println("兴趣爱好：" + Arrays.asList(hobby));
}
```

e)请求的转发

什么是请求的转发?

请求转发是指，服务器收到请求后，从一次资源跳转到另一个资源的操作叫请求转发。



Servlet1 代码：

```
public class Servlet1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {

        // 获取请求的参数（办事的材料）查看
        String username = req.getParameter("username");
        System.out.println("在 Servlet1（柜台 1）中查看参数（材料）：" + username);

        // 给材料 盖一个章，并传递到Servlet2（柜台 2）去查看
        req.setAttribute("key1","柜台 1 的章");

        // 问路：Servlet2（柜台 2）怎么走
        /**
         * 请求转发必须要以斜杠打头，/ 斜杠表示地址为：http://ip:port/工程名/ ，映射到 IDEA 代码的 web 目录
<br/>
         *
         */
        RequestDispatcher requestDispatcher = req.getRequestDispatcher("/servlet2");
//        RequestDispatcher requestDispatcher = req.getRequestDispatcher("http://www.baidu.com");

        // 走向 Sevlet2（柜台 2）
        requestDispatcher.forward(req,resp);
```

```
    }
}
```

Servlet2 代码：

```
public class Servlet2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        // 获取请求的参数（办事的材料）查看
        String username = req.getParameter("username");
        System.out.println("在 Servlet2（柜台 2）中查看参数（材料）：" + username);

        // 查看 柜台 1 是否有盖章
        Object key1 = req.getAttribute("key1");
        System.out.println("柜台 1 是否有章：" + key1);

        // 处理自己的业务
        System.out.println("Servlet2 处理自己的业务 ");
    }
}
```

f) base 标签的作用

当我们点击a标签进行跳转的时候，浏览器地址栏中的地址是：http://localhost:8080/07_servlet/a/b/c.html

跳转回去的a标签路径是：../../index.html

所有相对路径在工作时候都会参照当前浏览器地址栏中的地址来进行跳转。

那么参照后得到的地址是：
http://localhost:8080/07_servlet/index.html

正确的跳转路径

base标签可以设置当前页面中所有相对路径工作时，参照哪个路径来进行跳转。

当我们使用请求转发来进行跳转的时候，浏览器地址栏中的地址是
http://localhost:8080/07_servlet/forwardC

跳转回去的a标签路径是：../../index.html

所有相对路径在工作时候都会参照当前浏览器地址栏中的地址来进行跳转。

那么参照后得到的地址是：
http://localhost:8080/index.html

错误的路径

注意相对路径会根据当前浏览器地址栏中的地址进行跳转。因此在使用相对地址时要注意这个问题。

可以使用base标签固定参考的地址。

base标签中的地址可以省略资源名，直接用目录

```
<!DOCTYPE html>
<html lang="zh_CN">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <!--base 标签设置页面相对路径工作时参照的地址
            href 属性就是参数的地址值
    -->
    <base href="http://localhost:8080/07_servlet/a/b/">
</head>
<body>
    这是 a 下的 b 下的 c.html 页面<br/>
    <a href="../../index.html">跳回首页</a><br/>
</body>
</html>
```

g)Web 中的相对路径和绝对路径

在 javaWeb 中，路径分为相对路径和绝对路径两种：

相对路径是：

.	表示当前目录
..	表示上一级目录
资源名	表示当前目录/资源名

绝对路径：

http://ip:port/工程路径/资源路径

在实际开发中，路径都使用绝对路径，而不简单的使用相对路径。

1、绝对路径

2、base+相对

h)web 中 / 斜杠的不同意义

在 web 中 / 斜杠 是一种绝对路径。

/ 斜杠 如果被浏览器解析，得到的地址是：http://ip:port/

```
<a href="/">斜杠</a>
```

被浏览器解析和被服务器解析得到的地址是不同的。

/ 斜杠 如果被服务器解析，得到的地址是：http://ip:port/工程路径

1、<url-pattern>/servlet1</url-pattern>

2、servletContext.getRealPath("/");

3、request.getRequestDispatcher("/");

特殊情况： response.sendRediect("/"); 把斜杠发送给浏览器解析。得到 http://ip:port/

2.HttpServletResponse 类

a)HttpServletResponse 类的作用

HttpServletResponse 类和 HttpServletRequest 类一样。每次请求进来，Tomcat 服务器都会创建一个 Response 对象传递给 Servlet 程序去使用。HttpServletRequest 表示请求过来的信息，HttpServletResponse 表示所有响应的信息，

我们如果需要设置返回给客户端的信息，都可以通过 HttpServletResponse 对象来进行设置

b)两个输出流的说明。

字节流	getOutputStream();	常用于下载（传递二进制数据）
字符流	getWriter();	常用于回传字符串（常用）

两个流同时只能使用一个。
使用了字节流，就不能再使用字符流，反之亦然，否则就会报错。

c)如何往客户端回传数据

要求 ： 往客户端回传 字符串 数据。

```
public class ResponseIOServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
//        要求 ： 往客户端回传 字符串 数据。
        PrintWriter writer = resp.getWriter();
        writer.write("response's content!!!");
    }
}
```

d)响应的乱码解决

解决响应中文乱码方案一（不推荐使用）：

```
        // 设置服务器字符集为UTF-8
        resp.setCharacterEncoding("UTF-8");
        // 通过响应头，设置浏览器也使用UTF-8字符集
        resp.setHeader("Content-Type", "text/html; charset=UTF-8");
```

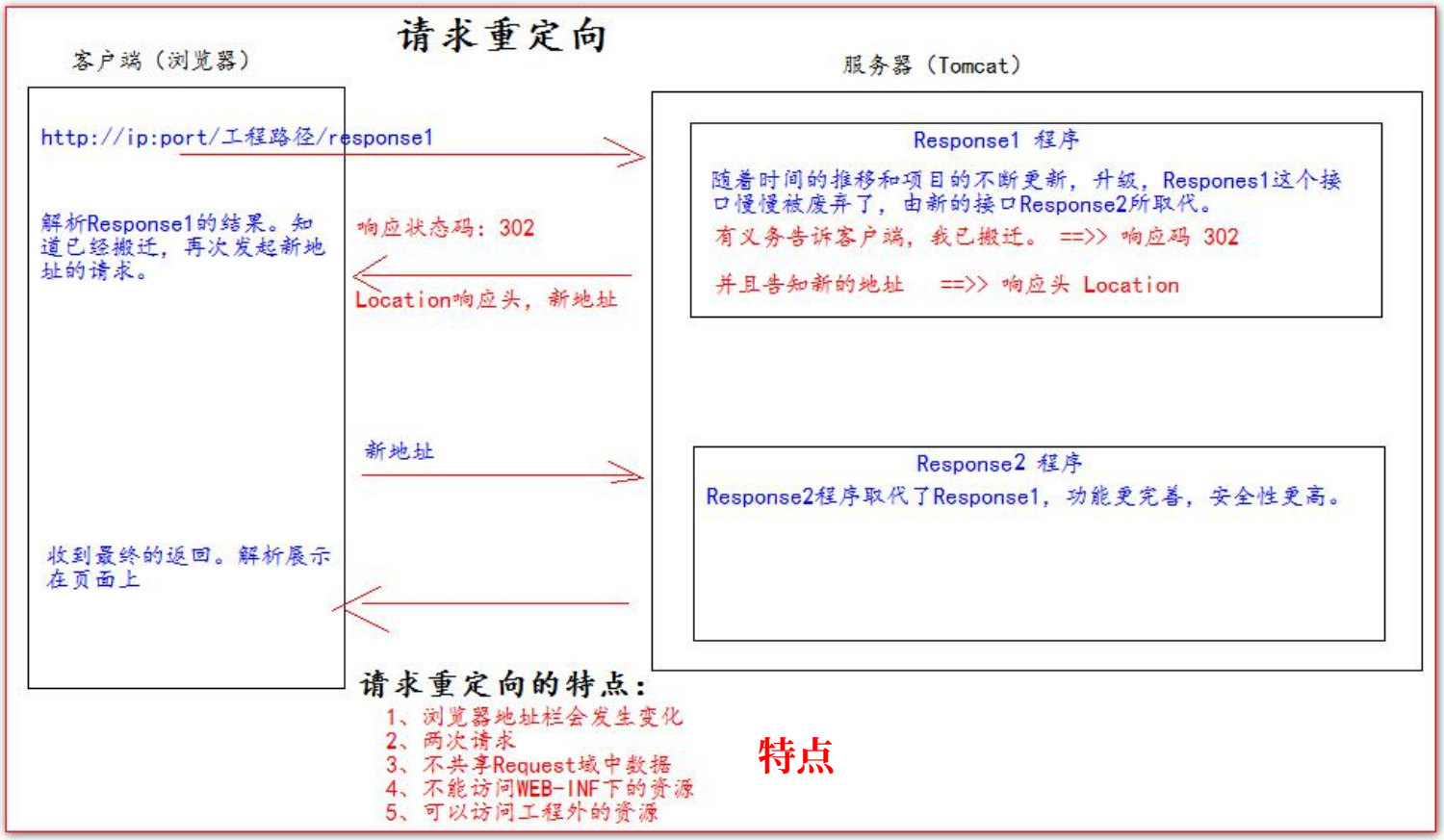
解决响应中文乱码方案二（推荐）：

```
// 它会同时设置服务器和客户端都使用 UTF-8 字符集，还设置了响应头
// 此方法一定要在获取流对象之前调用才有效
resp.setContentType("text/html; charset=UTF-8");
```

要求在获取流之前使用

e)请求重定向

请求重定向，是指客户端给服务器发请求，然后服务器告诉客户端说。我给你一些地址。你去新地址访问。叫请求重定向（因为之前的地址可能已经被废弃）。



请求重定向的第一种方案：

```
// 设置响应状态码 302 ，表示重定向，（已搬迁）
resp.setStatus(302);
// 设置响应头，说明 新的地址在哪里
resp.setHeader("Location", "http://localhost:8080");
```

两种方式

请求重定向的第二种方案（推荐使用）：

```
resp.sendRedirect("http://localhost:8080");
```