Request和Response

课前默写

```
1、写出Servlet获取请求参数
2、写出Servlet实现页面跳转的方式
3、写出Servlet的生命周期
4、写出Servlet的两种配置方式
5、写出ServletContext的常用方法
```

课程回顾

```
1、Servlet获取请求参数
2、Servlet实现页面跳转
```

今日内容

```
1、HTTPServletResponse的使用
2、HTTPServletRequest的使用
```

教学目标

```
1、掌握HTTPServletResponse的使用
2、熟悉HTTPServletRequest的使用
```

> 作为后台开发人员,我们大多时候都在接收处理用户请求,给予用户响应,为了方便操作,服务器软件将请求和响应封装成了Request和Response,我们今天就讲解两个对象的操作!

第七章 Response讲解

7.1 Response简介

定义辅助 servlet 将响应发送到客户端的对象。servlet 容器创建 ServletResponse 对象，并将它作为参数传递给 servlet 的 service 方法。 要发送 MIME 正文响应中的二进制数据，请使用 #getOutputStream 返回的 ServletOutputStream。要发送字符数据，请使用 #getWriter 返回的 PrintWriter 对象。要混合二进制数据和文本数据，例如要创建 multipart 响应，请使用 ServletOutputStream 并手动管理字符部分。 可使用 #setCharacterEncoding 和 #setContentType 显式指定 MIME 正文响应的 charset，或使用 #setLocale 方法隐式指定它。显式指定优先于隐式指定。如果未指定 charset，则将使用 ISO-8859-1。setCharacterEncoding、setContentType 或 setLocale 方法必须在调用 getWriter 之前，并且必须在提交采用要使用的字符编码的响应之前调用。

7.2 HttpServletResponse介绍

扩展 ServletResponse 接口以提供特定于 HTTP 的发送响应功能。例如，该接口拥有访问 HTTP 头和 cookie 的方法。 客户端向服务器发起的都是HTTP协议操作,所以我们大部分使用HttpServletResponse对象作为直接操作对象!

7.3 HttpServletResponse 常用API介绍

方法名称	作用
setStatus(int code)	设置响应状态码
setHeader(name,value)	设置响应信息头
setCharacterEncoding(String);	设置编码格式
setContentType(String)	设置返回数据mimetype
getWriter()	获取字符输出流
getOutputStream()	获取字节输出流

7.4 设置返回字符编码格式

- 方案1

```
response.setCharacterEncoding ()
设置tomcat编码格式
<html  meta charset=UTF-8 ><body>编写返回的文本内容</body>
设置浏览器解析文本内容格式
```

可以解决返回字符串乱码问题,但是需要将返回的字符串封装到html代码中.操作繁琐!

- 方案2

```
response.setCharacterEncoding ()
response.setHeader("Content-type","text/html;charset=UTF-8")
```

方案按相对简单,通过设置响应头告知浏览器解析字符串的编码格式!

- 方案3

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

利用setContentTyoe这种综合性的写法解决问题!此方法也是开发中常用的方法!方便!

7.5 Response练习

1. 用户下载服务器图片

```
public class DownLoadServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        /**
         * 别人访问我 我给你一张图片
         *
         * 1.获取图片的输入流
         * 2.response的字节输出流

         */
```

```
        //1.获取图片或者文件的输入流
        //想要或某个文件  你必须知道绝对路径
        //手段:  项目中的相对路径 去获取绝对路径
//      /吃鸡.jpg
//      /WEB-INF/classes/吃鸡.jpg
//      /WEB-INF/classes/com/itqf/servlet/吃鸡.jpg
        //ServletContext getRealPath 相对路径转成绝对路径
        String path = getServletContext().getRealPath("/吃鸡.jpg");

        System.out.println(path);
        //文件输入流
        FileInputStream stream = new FileInputStream(path);
        //xx/xx/x/x/x/x/x/吃鸡.jpg
        //获取文件的名字   /
        //File.separator  == /
        String filename = path.substring(path.lastIndexOf(File.separator)+1);

        //IE
        // filename =  URLEncoder.encode(filename,"UTF-8");
        //其他
        filename = new String(filename.getBytes("UTF-8"),"ISO-8859-1");
        //设置响应头
        //content-disposition", "attachment;filename="+filename
        //下载用的 attachment下载:  filename下载文件的名字
        response.setHeader("content-disposition", "attachment;filename="+filename);
        //根据文件名字的后缀名获取类型
        String mimetype = getServletContext().getMimeType(filename);
        response.setContentType(mimetype); //下载文件的类型
        ServletOutputStream outputStream = response.getOutputStream();
        byte [] buffer = new byte [8*1024];
        int len = 0;
        while((len=stream.read(buffer)) != -1){
            outputStream.write(buffer, 0, len);
        }
        stream.close();
    }
}
```

2. 页面中添加验证码
    1. 项目中添加第三方jar包!validatecode.jar
    2. 创建返回验证码的servlet!

```
public class CodeServlet extends HttpServlet {

    /**
         * The doGet method of the servlet. <br>
         * This method is called when a form has its tag value method equals to get
         * @param request the request send by the client to the server
         * @param response the response send by the server to the client
         * @throws ServletException if an error occurred
         * @throws IOException if an error occurred

         */
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //1.设置生成规则
        /**
         * width: 验证码的宽度  px 像素
         * height: px
         * codeCount:生成验证码有几个数
         * lineCount:有几根线
         */
        ValidateCode code = new ValidateCode(200, 50, 6, 20);
        //2.获取生成的验证码的字符串值
        System.out.println(code.getCode()); //获取正确值
        //3.响应写回验证图片
        code.write(response.getOutputStream());
    }
}
```

1. 页面中使用当前servlet

```
<!DOCTYPE html>
<html>
  <head>
    <title>login.html</title>
    <meta name="keywords" content="keyword1,keyword2,keyword3">
    <meta name="description" content="this is my page">
    <meta name="content-type" content="text/html; charset=UTF-8">
    <!--<link rel="stylesheet" type="text/css" href="./styles.css">-->
  </head>
  <body>
      <form action="/Day11_Response1/servlet/LoginServlet"  method="post">
          <input type="text" name="username" placeholder="请输入账号! " />
          <input type="submit" value="登录"/>
      </form>
      <img src="/Day11_Response1/servlet/CodeServlet" onclick="changeCodes()" alt="验证码"
title="验证码"/>
      <a href="javascript:changeCodes()" >看不清? </a>
      <script type="text/javascript">
        function changeCodes(){
          var img= document.getElementsByTagName("img")[0];
          img.src = "/Day11_Response1/servlet/CodeServlet?r="+Math.random();
        }
      </script>
  </body>
</html>
```

第八章 Request讲解

8.1 ServletRequst介绍

> 定义将客户端请求信息提供给某个 servlet 的对象。servlet 容器创建 `ServletRequest` 对象，并将该对象作为参数传递给该 servlet 的 `service` 方法。

8.2 HttpServletRequest介绍

HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，HTTP请求头中的所有信息都封装在这个对象中，开发人员通过这个对象的方法，可以获得客户这些信息。

小结: 同响应相同,客户端请求协议都是基于HTTP所以我们选用HttpServletRequest来操作用户发送过来的请求的数据!

8.3 HttpServletRequest常用API

```
//获取请求路径相关参数
getRequestURL方法返回客户端发出请求时的完整URL。
getRequestURI方法返回请求行中的资源名部分。
getQueryString 方法返回请求行中的参数部分。
getRemoteAddr方法返回发出请求的客户机的IP地址
getRemoteHost方法返回发出请求的客户机的完整主机名
getRemotePort方法返回客户机所使用的网络端口号
getLocalAddr方法返回WEB服务器的IP地址。
getLocalName方法返回WEB服务器的主机名
getMethod得到客户机请求方式
//获取请求头信息
getHead(name)方法
getHeaders(String name)方法
getHeaderNames方法
//获取请求正文参数
getParameter(name)方法
getParameterValues (String name) 方法
getParameterNames方法
getParameterMap方法  //做框架用，非常实用
getInputStream
```

8.4 获取请求数据练习

```
        //1.测试获取请求行数据的方法和请求头的方法
        //1.获取请求行的方法
         //获取请求方式
         String method = request.getMethod();
         //获取请求的url
         String url = request.getRequestURL()+"";
         //获取uri
         String uri = request.getRequestURI();
         //获取请求的参数  get
         String query = request.getQueryString();
         //获取请求人的id
         String ip = request.getRemoteAddr();
         //获取请求的主机名
         String host = request.getRemoteHost();
         System.out.println(method+" url:"+url+" uri:"+
                 uri+" query:"+query+" ip:"+ip+" host:"+host);
        //获取请求信息
        //全部输出
         //post fix
         String header = request.getHeader("user-agent");
         if (header.contains("firefox")) {
```

```
        }
           //获取所有请求的names
          Enumeration<String> headerNames = request.getHeaderNames();
          while (headerNames.hasMoreElements()) {
               String name = (String) headerNames.nextElement();
               //根据name获取keys
               Enumeration<String> headers = request.getHeaders(name);
               while (headers.hasMoreElements()) {
                    String key = (String) headers.nextElement();
                    System.out.println(name+"--->"+key);
               }
      }
```

8.5 封装请求参数

- 将数据封装到实体类上

  创建一个对应的实体类!

  > 实体类要变量命名和变量类型都有相应的要求,要求变量名跟提交参数的key相同,变量跟参数类型形同!

```
public class JavaBean {

    private String username;
    private String password;
    private String sex;
    private String [] hobby;
    //getter/setter
}
```

原生方式进行解析

```
        //表单的name值和value值
        //key  name值  value  values值
        Map<String, String[]> parameterMap = request.getParameterMap();

        JavaBean bean = new JavaBean();

        Set<Entry<String, String[]>> entrySet = parameterMap.entrySet();

        for (Entry<String, String[]> entry : entrySet) {
         //entry map中的一条
            //username password sex
            String key = entry.getKey();
            String[] value = entry.getValue();

            try {
             PropertyDescriptor descriptor =
                        new PropertyDescriptor(key, JavaBean.class);


             Method set = descriptor.getWriteMethod();
```

```
                /**
                 * 参数：哪个对象的set方法
                 * password username sex
                 */
                if (value.length == 1) {
                    set.invoke(bean, value[0]);
                }else{
                    set.invoke(bean, (Object)value);
                }


            } catch (IntrospectionException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalAccessException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalArgumentException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (InvocationTargetException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }
```

- 使用Apache BeanUtils进行快速映射

  1.导入beanutils对应jar包!

  2.映射

```
BeanUtils.populate(bean2, request.getParameterMap());
System.out.println(bean2);
```

8.6 Servlet如何处理HTTP协议

Web服务器收到客户端的http请求，会针对每一次请求，分别创建一个用于代表请求的HttpServletRequest对象、和代表响应的HttpServletResponse对象。

HttpServletRequest和HttpServletResponse对象即然代表请求和响应，那我们要获取客户端提交过来的数据，只需要找HttpServletRequest对象就行了。要向客户端输出数据，只需要找HttpServletResponse对象就行了

8.6.1 HttpServletRequest

HttpServletRequest对象代表客户端的请求，当客户端通过HTTP协议访问服务器时，HTTP请求头中的所有信息都封装在这个对象中，通过这个对象提供的方法，可以获得客户端请求的所有信息

获得客户端信息
    getRequestURL方法返回客户端发出请求时的完整URL。
    getRequestURI方法返回请求行中的资源名部分。
    getQueryString 方法返回请求行中的参数部分。
    getPathInfo方法返回请求URL中的额外路径信息。额外路径信息是请求URL中的位于Servlet的路径之后和查询参数之前的内容，它以“/”开头。
    getRemoteAddr方法返回发出请求的客户机的IP地址。
    getRemoteHost方法返回发出请求的客户机的完整主机名。
    getRemotePort方法返回客户机所使用的网络端口号。
    getLocalAddr方法返回WEB服务器的IP地址。
    getLocalName方法返回WEB服务器的主机名。

获得客户端请求头
    getHeader(string name)方法:获取指定名称的String值
    getHeaders(String name)方法:获取指定名称的Enumeration值
    getHeaderNames()方法：获取所有的请求消息头的名称

获得客户机端请求参数(客户端提交的数据)
getParameter(String)方法(常用)：获取指定参数的值
getParameterValues(String name)方法(常用)：获取指定参数的所有的值
getParameterNames()方法(不常用)：获取所有的请求参数
getParameterMap()方法(编写框架时常用)：获取所有的请求参数和对应的值

代码如下所示：

```
/**
 * Servlet implementation class RequestServlet
 * 演示request的各种常用方法
 */
@WebServlet("/RequestServlet")
public class RequestServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public RequestServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see 处理GET请求
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //编码格式
        //request.setCharacterEncoding("UTF-8");
        //获取参数方法
        System.out.println(request.getParameter("name"));

        System.out.println(Arrays.toString(request.getParameterValues("name")));
```

```
        System.out.println(request.getParameterMap());
        System.out.println(request.getParameterNames());
        //获取请求方法
        System.out.println("getMethod--->"+request.getMethod());
       //获取远程ip地址
        System.out.println(request.getRemoteAddr());
       //获取远程端口号
        System.out.println(request.getRemotePort());
        System.out.println(request.getRemoteHost());
        System.out.println(request.getRemoteUser());
        System.out.println("getRequestURL:"+request.getRequestURL());
        System.out.println("getContextPath:"+request.getContextPath());
        System.out.println("getServletPath--->"+request.getServletPath());
        System.out.println("getServerName--->"+request.getServerName());
       Enumeration<String> ets=request.getHeaderNames();
         while (ets.hasMoreElements()) {
            String key=ets.nextElement();
            System.out.println("请求消息头：key="+key+"---值："+request.getHeader(key));
        }
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        // TODO Auto-generated method stub
        //doGet(request, response);
    }
}
```

8.6.2 HttpServletResponse

HttpServletResponse对象代表服务器的响应。这个对象中封装了向客户端发送数据、发送响应头，发送响应状态码的方法。

向客户端(浏览器)发送响应头的相关方法：
addDateHeader：把给定的名字和日期值加入到响应的头部
addHeader：给定的名字和和数值加到响应的头部
addIntHeader：将给定的名字和整数值加到一个相应的头部
containsHeader：是否指定的响应消息头部已被设置过了
setDateHeader：设置带有给定的名字和数值的响应消息头
setHeader：设置带有给定的名字和数值的响应消息头
setIntHeader：设置带有给定的名字和整数值的响应消息头
setContentType：设置响应内容的格式
addCookie：新增Cookie

设置状态响应码：
setStatus：设置Http协议的状态响应码

负责向客户端(浏览器)发送数据的相关方法：
getOutputStream：获取输出字节流向客户端写出内容
getWriter：获取输出字符流向客户端写出内容

URL重定向：
encodeURL：可以对url进行重定向，也就是将JSESSIONID传递

代码如下所示：

```
/**
 * Servlet implementation class ResponseServlet
 * 演示response常用的方法
 */
@WebServlet("/ResponseServlet")
public class ResponseServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ResponseServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        //编码格式
        //request.setCharacterEncoding("");//设置请求内容的编码格式
        response.setCharacterEncoding("UTF-8");//设置响应数据的编码格式
        response.setStatus(404);//设置状态响应码
        //设置响应内容的类型和编码格式
        response.setContentType("text/html;charset=UTF-8");
        //response.getOutputStream();
        //获取打印字符流
        PrintWriter pw=response.getWriter();
        pw.println("困了");
        pw.println("站起来");
        pw.close();
        System.out.println("OK");

    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
```

```
    }
}
```

作业题

```
1.将用户登录验证 修改为连接数据版本
    登录成功 进入图书列表页面
2.将购物车功能 修改为连接数据库版本
    图书列表页面-->
        点击图书名称 -->
          图书详细信息页面-->
                点击添加进购物车-->将数据添加入购物车中
        点击查看购物车 -->查看购物车中的图书
                              -->点击清空购物车 执行清空
                        清空后显示提示信息"购物车已被清空"
                        跳转回图书列表页
```

面试题

```
1、request.getAttribute()和request.getParameter()的区别
```