3 Response

一、学习目标

1. 掌握 HTTP 协议之响应部分
2. 熟练掌握 Response 的相关 API （重定向、路径、输出流）
3. 掌握 ServletContext 的常用 API
4. 独立完成文件下载案例

二、学习内容

1. HTTP 协议之响应消息

1.1. HTTP 响应消息概述

> 学习目标：了解 HTTP 响应消息的概述
>
> 掌握程度：了解

响应消息：服务器端发送给客户端的数据

响应消息数据格式：

- 响应行：协议/版本 响应状态码 响应状态码描述
- 响应头：设置内容怎么展示的，头名称：值
- 响应空行：空 —— 行
- 响应体：传输的数据，页面展示信息的内容

> **课堂提问**
>
> 简述 HTTP 响应消息的消息格式

1.2. HTTP 响应消息：响应行 & 状态码

> 学习目标：了解常用状态码及其含义

> 掌握程度：了解
>
> 要点提示：常用状态码表示的含义

响应行：协议/版本 响应状态码 状态码描述

响应状态码：服务器告诉客户端浏览器本次请求和响应的一个状态

- 状态码都是 3 位数字
- 分类：
  - 1xx：服务器接收客户端消息，但没有接收完成，等待一段时间发送 1xx 状态码
  - 2xx：成功，代表：200
  - 3xx：重定向，代表：302（重定向），304（访问缓存）
  - 4xx：客户端错误
    - 404 请求路径没有对应的资源
    - 405 请求方式没有对应的 doXX 方法
  - 5xx：服务器错误，如 500（服务器内部出现异常）

HTTP 响应状态码 – HTTP | MDN

HTTP 响应状态码用来表明特定 HTTP 请求是否成功完成。 响应被归为以下五大类：

MDN Web Docs

> **课堂提问**
>
> 服务器内部错误的状态码常以什么开头?

1.3. HTTP 响应消息：响应头

> 学习目标：掌握记忆常用响应头及其含义
>
> 掌握程度：记忆
>
> 要点提示：常用响应消息及其含义

MIME 类型（IANA 媒体类型） – HTTP | MDN

媒体类型（也通常称为多用途互联网邮件扩展或 MIME 类型）是一种标准，用来表示文档、文件或一…

MDN Web Docs

格式：头名称：值

常见响应头：

a. Content-Type：text/html;charset=UTF-8
    i. Content-Type：服务器告诉客户端，本次响应体数据格式，以及编码格式
    ii. text：表示文本内容
    iii. html：表示 html 格式
    iv. charset：编码格式，浏览器会根据反馈的内容改变当前页面的字符集
b. Content – Length：字节个数
c. Date：日期

2. Response 对象 (一)

2.1. Response 功能介绍

> 学习目标：掌握 Response 对象的作用和使用
>
> 掌握程度：掌握
>
> 要点提示：Response 对象的功能

Response 对象是一个非常重要的对象，主要负责响应数据给浏览器

功能：设置响应消息

a. 设置响应行：
    i. 格式：HTTP/1.1 200 ok
    ii. 设置状态码：setStatus（int sc）
b. 设置响应头：setHeader（String name，String value）
c. 设置响应体：
    i. 获取输出流
    ii. 字符输出流：PrintWriter getWriter()
    iii. 字节输出流：ServletOutputStream getOutputStream()
    iv. 使用输出流，将数据输出到客户端浏览器

> **课堂提问**
>
> Response 对象的作用

2.2. Response 案例：重定向代码实现

> 学习目标：理解并独立实现重定向案例
>
> 掌握程度：掌握
>
> 重定向案例

重定向：资源跳转的方式

重定向代码实现

```java
@WebServlet("/responseDemo1")
public class ResponseDemo1 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
         System.out.println("ResponseDemo1........");
        //访问/responseDemo1，会自动跳转到/responseDemo2资源
        //1. 设置状态码为302
        response.setStatus(302);
        //2.设置响应头location
        response.setHeader("location", "/responseDemo2");
        //简单的重定向方法
        response.sendRedirect("/responseDemo2");
        // 重定向可以跨域访问
        //response.sendRedirect("https://www.baidu.cn");
    }
}

```

```java
@WebServlet("/responseDemo2")
public class ResponseDemo2 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("responseDemo2被访问...");
    }
}
```

> **课堂练习**
>
> 完成重定向案例

2.3. 重定向特点

> 学习目标：理解并掌握重定向的特点
>
> 掌握程度：理解记忆
>
> 提示要点：重定向的特点

redirect 重定向的特点：

a. 地址栏发生变化

b. 重定向可以访问其他站点（服务器）的资源

c. 重定向是两次请求，不能使用 request 对象来共享数据

forword 转发 的特点：

a. 转发地址栏路径不变

b. 转发只能访问当前服务器下的资源

```java
@WebServlet("/responseDemo3")
public class ResponseDemo3 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //转发
        request.getRequestDispatcher("/responseDemo2").forward(request, response);
    }
}
```

> **课堂提问**
>
> 重定向的特点?

3. Response对象（二）

3.1. Response 案例：相对路径

> 学习目标：掌握跳转路径之相对路径
>
> 掌握程度：掌握
>
> 提示要点：相对路径

相对路径：通过相对路径不可以确定唯一资源，如：./index.html

- 不以 / 开头，以 . 开头路径
- 规则：找到当前资源和目标资源之间的相对位置关系
  - ./：当前目录
  - ../：后退一级目录

> **课堂提问**
>
> 什么是相对路径?

3.2. Response 案例：绝对路径

> 学习目标：理解掌握路径跳转 —— 绝对路径
>
> 掌握程度：掌握
>
> 提示要点：绝对路径

绝对路径：通过绝对路径可以确定唯一资源

a. http://localhost:8080/responseDemo2
b. /responseDemo2
c. 以 / 开头的路径

规则：判断定义的路径是给谁用的？判断请求将从哪儿发出来

- 给客户端浏览器使用：需要加虚拟目录（项目的访问路径）
  - 超链接
  - 重定向
  - 建议虚拟目录动态获取：request.getContextPath()
- 给服务器使用：不需要夹虚拟目录
  - forward 转发

> 课堂提问
>
> 什么是绝对路径？转发和重定向份别怎么设置路径?

3.3. Response 案例：输出字符数据

> 学习目标：了解使用 Response 对象输出字符数据
>
> 掌握程度：了解

> 要点提示：Response 如何输出字符数据

步骤

1. 获取字符输出流
2. 输出数据

注意乱码问题：

- PrintWriter pw = response.getWriter() 获取的流的默认编码是 ISO-8859-1
- 设置该流的默认编码，告诉浏览器响应体的编码，**【要在获取流之前设置】**
  - response.setContentType（"text/html;charset=utf-8"）

```java
@WebServlet("/responseDemo4")
public class ResponseDemo4 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //获取流对象之前，设置流的默认编码：ISO-8859-1 设置为：GBK
        // response.setCharacterEncoding("utf-8");
        //告诉浏览器，服务器发送的消息体数据的编码，建议浏览器使用该编码解码
        //response.setHeader("content-type","text/html;charset=utf-8");
        //简单形式设置编码
        response.setContentType("text/html;charset=utf-8");
        //1.获取字符输出流
        PrintWriter pw = response.getWriter();
        //2.输出数据
        //pw.write("<h1>hello response</h1>");
        pw.write("你好 response");
    }
}
```

> **课堂提问**
>
> Response 如何实现字符数据输出?

3.4. Response 案例：输出字节数据

> 学习目标：理解掌握使用 Response 对象输出字节数据
>
> 掌握程度：应用
>
> 提示要点：Response 对象输出字节数据

步骤：

1. 获取字节输出流
2. 输出数据

```java
@WebServlet("/responseDemo5")
public class ResponseDemo5 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        //1.获取字节输出流
        ServletOutputStream sos = response.getOutputStream();
        //2.输出数据
        sos.write("你好".getBytes(StandardCharsets.UTF_8));
    }
}
```

> **课堂提问**
>
> Response 如何 实现字节数据输出?

4. Response对象（三）

4.1. Response 案例：验证码

> 学习目标：了解验证码案例需求分析
>
> 掌握程度：了解
>
> 提示要点：验证码需求分析

验证码需求分析

1. 创建一个在内存中的验证码图片对象
2. 通过该图片获取画笔对象
    a. 设置画笔颜色
    b. 绘制填充矩形
    c. 生成四位随机字符串并绘制到画布
    d. 绘制干扰线

3. 输出验证码图片

Java

```java
@WebServlet("/verifyCodeServlet")
public class VerifyCodeServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int width = 160;
        int height = 45;

        //1.创建验证码图片对象
        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

        //2.美化图片
        //2.1 填充背景色
        //画笔对象
        Graphics g = image.getGraphics();
        //设置画笔颜色
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, width, height);
        String str = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghigklmnopqrstuvwxyz0123456789";
        //生成随机角标
        Random ran = new Random();

        for (int i = 1; i <= 5; i++) {
            int index = ran.nextInt(str.length());
            //获取随机字符
            char ch = str.charAt(index);
            // 设置字体颜色
            Color color = new Color(random.nextInt(256), random.nextInt(256), random.nextInt(256));
            g.setColor(color);
            //2.2写验证码
            g.drawString(ch + "", width / 5 * i, height / 3);
        }

        //2.3 画干扰线
        g.setColor(Color.GREEN);
        //随机生成坐标点
        for (int i = 0; i < 10; i++) {
            int x1 = ran.nextInt(width);
            int x2 = ran.nextInt(width);

            int y1 = ran.nextInt(height);
```

```
41              int y2 = ran.nextInt(height);
42              g.drawLine(x1, y1, x2, y2);
43          }
44
45          //3.将图片输出到页面展示
46          ImageIO.write(image, "jpg", response.getOutputStream());
47      }
48
49      @Override
50      protected void doGet(HttpServletRequest request, HttpServletResponse r
    esponse) throws ServletException, IOException {
51          this.doPost(request, response);
52      }
53  }
```

> **课堂提问**
>
> 验证码案例的实现步骤?

4.2. Response 案例：切换验证码

> 学习目标：独立实现验证码切换功能
>
> 掌握程度：掌握
>
> 提示要点：验证码切换功能

分析：点击超链接或者图片，需要换一张

1. 给超链接和图片绑定点击事件
2. 重新设置图片的 src 属性值

```HTML
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>注册页面（验证码）</title>
    <script>
        /*
         分析：点击超链接或者图片，需要换一张
         1.给超链接和图片绑定单击事件
         2.重新设置图片的src属性值
         */
        window.onload = function () {
            //1.获取图片对象
            const img = document.getElementById("checkCode");
            //2.绑定单击事件
            img.onclick = function () {
                //加时间戳避免缓存
                const date = new Date().getTime();
                img.src = "/verifyCodeServlet?" + date;
            }
        }
    </script>
</head>
<body>
  <img id="checkCode" src="/verifyCodeServlet" alt="img"/>
  <a id="change" href="">看不清换一张？</a>
</body>
</html>
```

> **课堂提问**
>
> 描述验证码切换思路

5. ServletContext 对象

5.1. ServletContext 概述

> 学习目标：了解 ServletContext 对象
>
> 掌握程度：了解
>
> 提示要点：ServletContext 对象

ServletContext 代表整个 Web 应用，它可以和程序的容器（服务器）来通信

ServletContext 的作用：

1. 获取 MIME 类型：在互联网通信过程中定义的一种文件数据类型
    a. 格式：大类型/小类型 text/html、image/jpeg
    b. 获取：String getMimeType（String file）
2. 域对象：共享数据
3. 获取文件的真实（服务器）路径

> **课堂提问**
>
> ServletContext 的作用

5.2. ServletContext 获取

> 学习目标：掌握 ServletContext 对象的获取方式
>
> 掌握程度：掌握
>
> 提示要点：ServletContext 对象的获取方式

ServletContext 获取：

- 通过 request 对象获取 request.getServletConetxt()
- 通过 HttpServelt 获取 this.getServletContext()

```java
@WebServlet("/servletContextDemo1")
public class ServletContextDemo1 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        /*
         ServletContext对象获取:
           1. 通过request对象获取 request.getServletContext();
           2. 通过HttpServlet获取 this.getServletContext();
         */

        //1. 通过request对象获取
        ServletContext context1 = request.getServletContext();
        //2. 通过HttpServlet获取
        ServletContext context2 = this.getServletContext();
        System.out.println(context1);
        System.out.println(context2);

        //true
        System.out.println(context1 == context2);
    }
}
```

> **课堂提问**
>
> ServletContext 对象如何获取

5.3. ServletContext 功能：获取 MIME 类型

> 学习目标：掌握使用 ServletContext 获取 MIME 类型
>
> 掌握程度：掌握
>
> 提示要点：获取 MIME 类型

获取文件 MIME 类型

- MIME 类型：在互连网通信过程中定义的一种文件数据类型
  - 格式：大类型/小类型 text/html image/jpeg
  - 获取：String getMimeType（String file）

```java
@WebServlet("/servletContextDemo2")
public class ServletContextDemo2 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        /*
          ServletContext功能:
            1. 获取MIME类型:
             MIME类型:在互联网通信过程中定义的一种文件数据类型
             格式:  大类型/小类型   text/html      image/jpeg
             获取: String getMimeType(String file)
            2. 域对象: 共享数据
            3. 获取文件的真实(服务器)路径
         */

        //2. 通过HttpServlet获取
        ServletContext context = this.getServletContext();

        //3. 定义文件名称

        String filename = "a.jpg";

        //4.获取MIME类型
        String mimeType = context.getMimeType(filename);
        System.out.println(mimeType);
    }
}
```

> **课堂提问**
>
> 如何获取文件 MIME 类型

5.4. ServletContext 功能：域对象

> 学习目标：掌握使用 ServletContext 作为域对象
>
> 掌握程度：掌握
>
> 提示要点：ServletContext 作为域对象使用

ServletContext 作为域对象共享数据

- setAttribute(String name,Object name)

- getAttribute(String name)
- removeAttribute(String name)

ServletContext 对象范围：所有用户所有请求的数据

```java
@WebServlet("/servletContextDemo3")
public class ServletContextDemo3 extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // 通过HttpServlet获取
        ServletContext context = this.getServletContext();
        //设置数据
        context.setAttribute("msg", "hello world");
    }
}

@WebServlet("/servletContextDemo4")
public class ServletContextDemo4 extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // 通过HttpServlet获取
        ServletContext context = this.getServletContext();
        //获取数据
        Object msg = context.getAttribute("msg");
        System.out.println(msg);
    }
}
```

> **课堂提问**
>
> 常用的操作域对象的方法有哪些?

5.5. ServletContext 功能：获取文件服务器路径

> 学习目标：掌握使用 ServletContext 获取文件服务器路径
>
> 掌握程度：掌握
>
> 提示要点：ServletContext 获取文件服务器路径

ServletContext 获取文件服务器路径方法：

```java
String getRealPath(String path)
```

String b = context.getRealPath("/b.txt");　　//web目录下资源访问

String c = context.getRealPath("/WEB-INF/c.txt");　//WEB-INF目录下的资源访问

String a = context.getRealPath("/WEB-INF/classes/a.txt"); //resources目录下的资源访问

```java
@WebServlet("/servletContextDemo5")
public class ServletContextDemo5 extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // 通过 HttpServlet 对象获取 ServletContext 对象
        ServletContext context = this.getServletContext();

        // webapp目录下资源访问
        String b = context.getRealPath("/b.txt");
        System.out.println(b);

        // WEB-INF目录下的资源访问
        String c = context.getRealPath("/WEB-INF/c.txt");
        System.out.println(c);

        // resources 目录下的资源访问
        String a = context.getRealPath("/WEB-INF/classes/a.txt");
        System.out.println(a);

        // 读取文件内容
        File file = new File(bRealPath);
        InputStream in = new FileInputStream(file);
        int read = 0;
        ServletOutputStream out = resp.getOutputStream();
        while ((read = in.read()) != -1) {
            out.write(read);

        }
        out.flush();
        out.close();

    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        this.doPost(request, response);
    }
}
```

> **课堂提问**
>
> 如何使用 ServletContext 获取文件服务器路径?

6. 文件下载案例

6.1. 文件下载分析

> 学习目标：理解文件下载的基本步骤和需求分析
>
> 掌握程度：理解记忆
>
> 提示要点：需求分析

文件下载需求：

1. 页面显示超链接
2. 完成图片文件下载
3. 进阶：点击超链接后弹出下载提示框

> **课堂提问**
>
> 文件下载的步骤分析

6.2. 文件下载代码实现

> 学习目标：掌握文件下载的代码实现
>
> 掌握程度：掌握
>
> 提示要点：文件下载代码实现

```java
@WebServlet("/download")
public class DownloadServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // 获取请求参数，文件名称
        String filename = request.getParameter("filename");

        // 找到文件服务器路径
        ServletContext servletContext = this.getServletContext();
        String realPath = servletContext.getRealPath("/img/" + filename);

        // 用字节流关联
        FileInputStream fis = new FileInputStream(realPath);

        // 获取文件的 mime 类型
        String mimeType = servletContext.getMimeType(filename);

        // 设置响应头类型：content-type
        response.setHeader("content-type", mimeType);

        // 设置响应头打开方式:content-disposition
        response.setHeader("content-disposition", "attachment;filename=" + filename);

        // 将输入流的数据写出到输出流中
        ServletOutputStream sos = response.getOutputStream();
        byte[] buff = new byte[1024 * 8];
        int len;
        while ((len = fis.read(buff)) != -1) {
            sos.write(buff, 0, len);
        }
        fis.close();
    }
}
```

前端

```HTML
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>下载文件</title>
</head>
<body>
<a href="download?filename=img.png">壁纸</a>
<a href="download?filename=b.txt">文件</a>
</body>
</html>
```

> **课堂练习**
>
> 独立完成文件下载代码实现

三、应知应会

1. 概念理解

- 简述 HTTP 响应消息的消息格式?
- 服务器内部错误的状态码常以什么开头?
- 说出常用的响应头及其含义
- Response 对象的作用?
- 重定向的特点?
- 什么是相对路径?
- 什么是绝对路径? 转发和重定向分别怎么设置路径?
- Response 如何实现字符数据输出?
- Response 如何实现字节数据输出?
- 验证码案例的实现步骤?
- 描述验证码切换思路
- ServletContext 的作用?
- ServletContext 对象如何获取?

- 如何获取文件 MIME 类型?
- 常用的操作域对象的方法有哪些?
- 如何使用 ServletContext 获取文件服务器路径
- 文件下载的步骤分析路径
- 如何处理文件上传过程中的中文文件名称问题?

2. 动手练习

- 完成重定向案例
- 动手尝试实现验证码
- 动手实现文件下载

四、课后任务

1. 整理课堂笔记
2. 完成教学笔记所有代码练习