

response学习

学习目标

- 1.能够使用使用浏览器开发工具查看响应
- 2.能够理解向银行（状态行）的内容
- 3.能够理解常见的状态码
- 4.能够使用Response对象操作HTTP响应内容
- 5.能够处理响应乱码
- 6.能够完成文件下载案例
- 7.能够使用servletcontext域对象
- 8.能够说出servlet生命周期方法执行流程

第1章 http协议

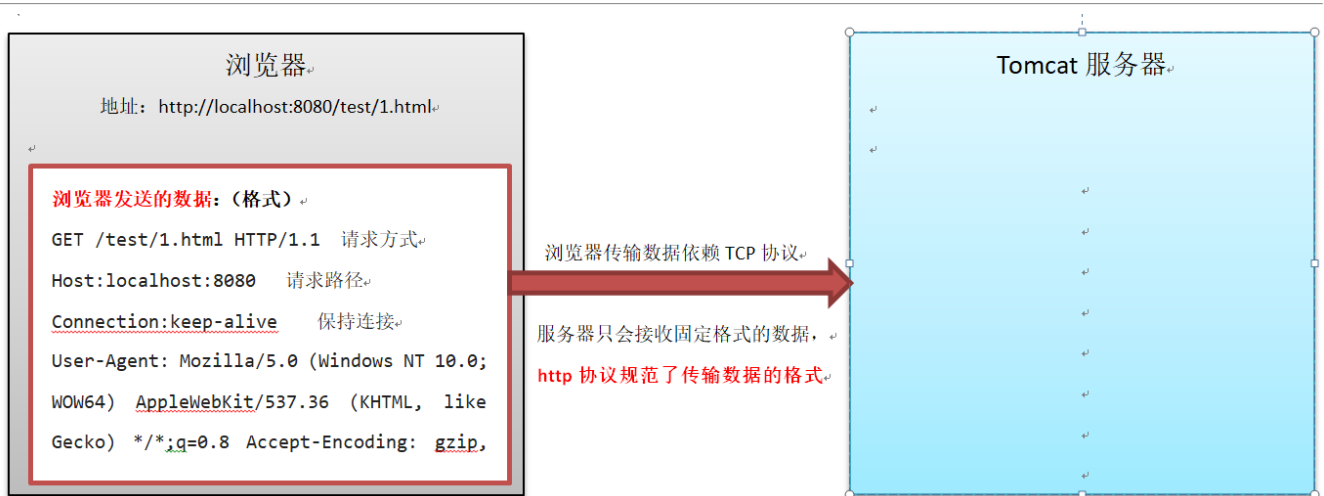
1.1 http协议的基本概念

1.1.1 http协议的简介

超文本传输协议（HTTP，HyperText Transfer Protocol）是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。1960年美国人Ted Nelson构思了一种通过计算机处理文本信息的方法，并称之为超文本（hypertext），这成为了HTTP超文本传输协议标准架构的发展根基。Ted Nelson组织协调万维网协会（World Wide Web Consortium）和互联网工程工作小组（Internet Engineering Task Force）共同合作研究，最终发布了一系列的RFC，其中著名的RFC 2616定义了HTTP 1.1。

1.1.2 http协议的作用

互联网的数据传输底层依赖tcp协议，http协议架构在tcp协议之上，**它规范了传输数据的内容和格式。**



1.1.3 HTTP协议的应用场景

刷屏软件

| | | |
|--|---|--|
| yy刷屏器 v1.0 绿色版 ★★★★★ 杂类工具 12.7M 下载 | Candy刷屏器 2016 免费版 ★★★★★ 杂类工具 904KB 下载 | 花锦万能刷屏器 v1.6 绿色版 ★★★★★ 杂类工具 475KB 下载 |
|--|---|--|

刷屏软件的原理其实通过观察12306查阅剩余票数需要发出的参数，然后自己编程拼接服务器所需要的参数发给服务器，然后接受服务器的数据。如果需要查看12306发出的请求参数格式，那么就必须学习http协议。

智能家居



SUPOR 终身保修

WIFI智能饭煲

智能云操控
4L大容量

到手价
¥289

极速物流 正品保证 全国联保

¥349.00 包邮 5人付款

SUPOR/苏泊尔 CFXB40FC8533-75智能
WIFI手机遥控家用自动新电饭煲

苏泊尔文耀专卖店 浙江 台州

智能家居在我们的生活中的地位越来越重了，手机是如何操作我们的家庭设备的呢？其实也是手机软件通过拼接参数，然后发送给电话煲的，这个过程也是需要使用到http协议的。不管是刷屏软件还是智能家居都需要知道服务器要何种格式的数据，那么我们怎么才能知道呢？我们可以查看浏览器与服务器的通讯过程。

1.2 浏览器查看浏览器与服务器通讯过程

接下来我们准备开始学习HTTP协议的具体内容，既然要学习内容，必然要先能看的具体的内容，这就需要借助查看http协议内容的工具了，工具一般有两个：

1.2.1 chrome(谷歌)——按键f12:

localhost:9090/aaa/1.html 输入访问路径，按下f12(注意不要按回车，发送请求)

Hello Tomcat

按下f12之后，打开谷歌开发者工具，选择network监控网络。然后，回到位置1，按下回车发送请求。

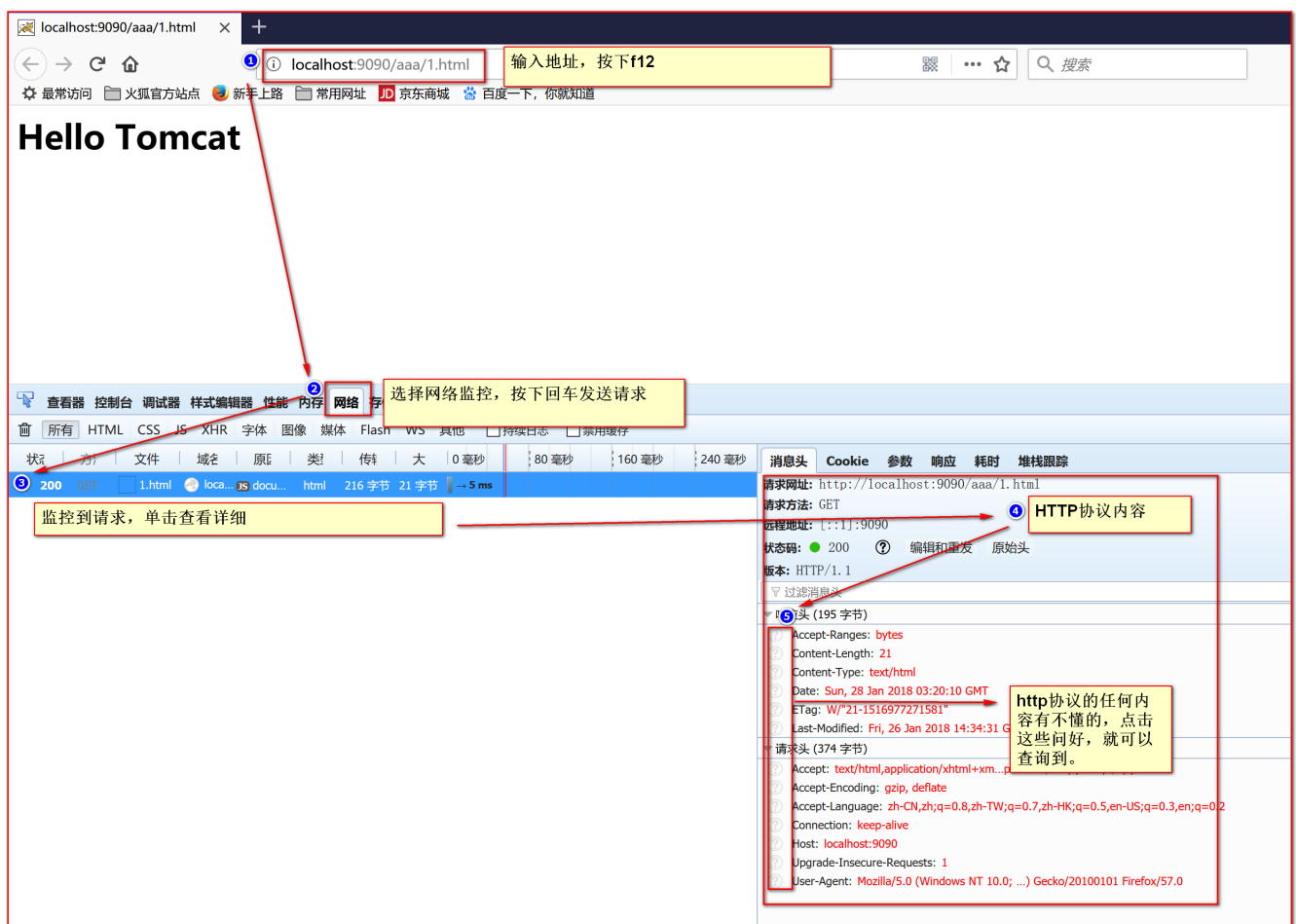
发送请求后，可以查看这一次请求和响应的内容。

这里是http协议的内容

谷歌浏览器，默认，对http协议做了修饰，为了看到http协议原来的模样，将4、5两处位置单击，单击后效果如图所示，就http协议内容原来的样子。

1 requests | 216 B transferred | Finish: 8 ms | DOMCo...

1.2.2 firefox(火狐)——按键f12:



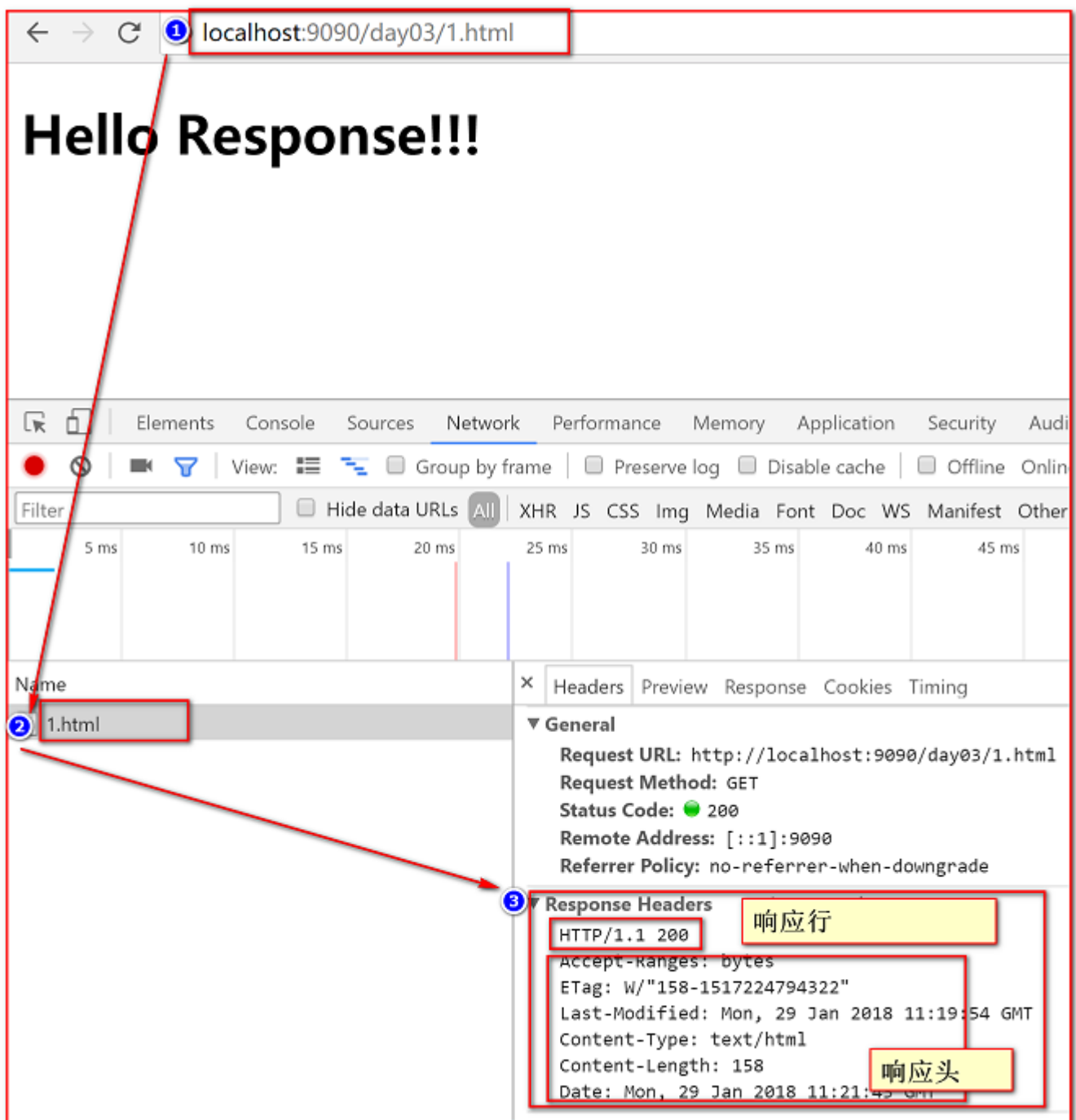
了解了如何使用工具查看http协议，那么接下来，我正式开始学习里面的内容：第1章 响应信息的组成部分
在前面我们学习过了http协议与请求相关的内容，那么接下来的任务就是要完成响应部分的学习。

1.1 http协议中的响应

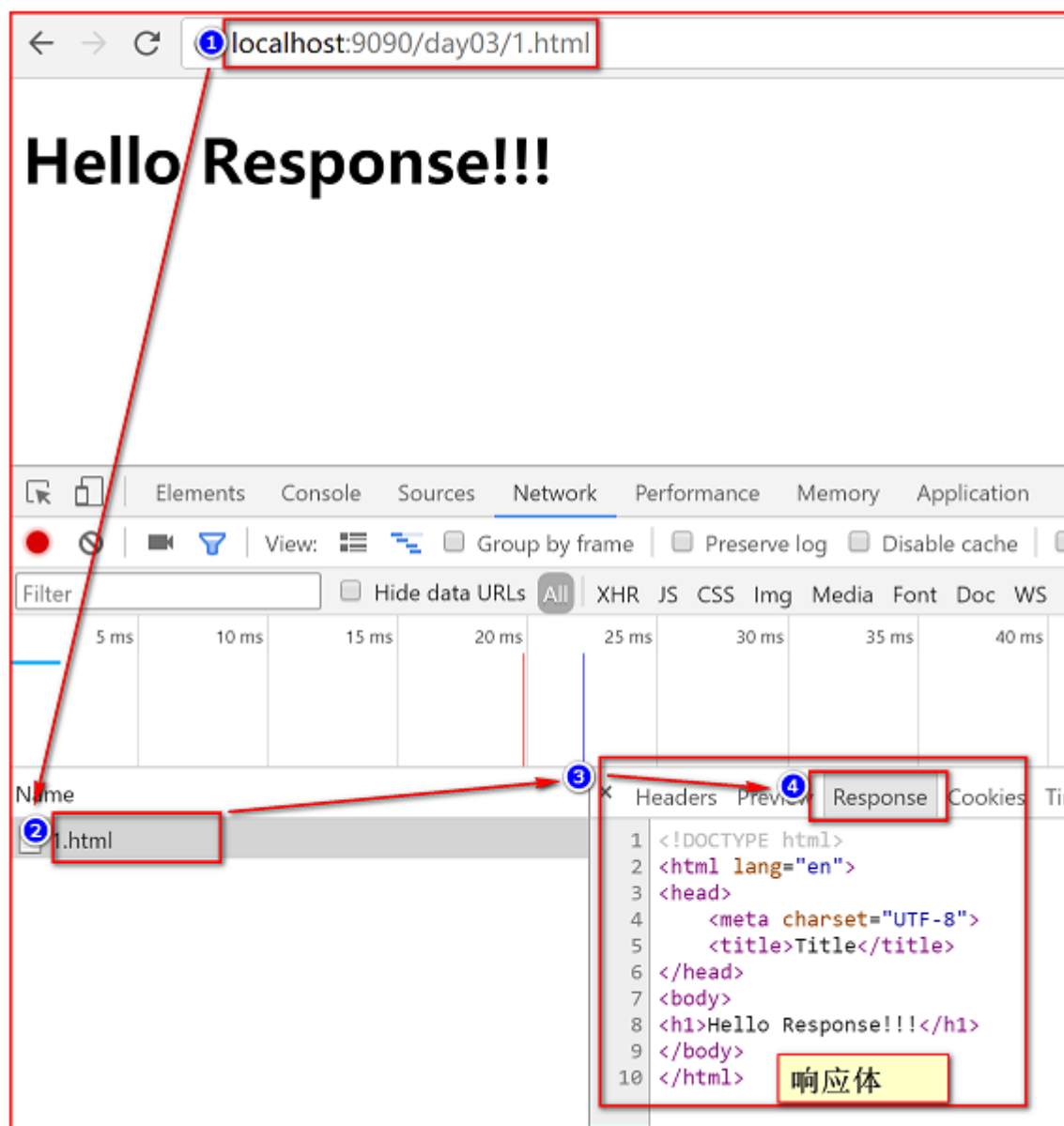
1.1.1 使用浏览器开发工具查看响应

http协议中的响应分成三个部分——响应行（状态行）、响应头、响应体。

我们先来看一下这三部分内容：



上图展示了响应行和响应头，那么还有响应体在哪里呢？下图展示了响应体如何查看：



注意：谷歌浏

览器并没有将响应的所有内容都放在一起，所以响应体的内容需要单独在response标签页查看。

由上图可知，那么既然分成三个部分，因此，我们的学习也是一步一步来，我们先从响应行（状态行）开始。

第2章 响应行（状态行）

2.1 状态行的组成

响应行是http响应内容的第一行。

响应行一般数据为：HTTP/1.1 200 (tomcat8.5) 或者 HTTP/1.1 200 OK (tomcat7)

响应行分为三个部分： HTTP/1.1: 协议版本

200: 响应状态码

OK: 对响应状态码的解释

2.2 常见的响应状态码：

1. **200 OK** 请求已成功，请求所希望的响应头或数据体将随此响应返回。出现此状态码是表示正常状态。
2. **302 Move temporarily** 重定向，请求的资源临时从不同的 URI 响应请求。
3. **304 Not Modified** 从缓存中读取数据，不从服务器重新获取数据。
4. **403 Forbidden** 服务器已经理解请求，但是拒绝执行它。
5. **404 Not Found** 请求失败，请求所希望得到的资源未被在服务器上发现。
6. **405 Method Not Allowed** 请求行中指定的请求方法不能被用于请求相应的资源。
7. **500 Internal Server Error** 服务器遇到了一个未曾预料的状态，导致了它无法完成对请求的处理。

以上就是我们常用的响应状态码，了解了http协议有关响应行的部分，那么我们接下来要学习，如何使用java对象操作响应。

第3章 response对象操作

3.1 response对象的概念

3.1.1 什么是response对象

我们要学习使用response对象，先要了解它，我们先来看下官方文档截图：

javax.servlet.http

Interface HttpServletResponse

All Superinterfaces:

[ServletResponse](#)

All Known Implementing Classes:

[HttpServletResponseWrapper](#)

```
public interface HttpServletResponse  
extends ServletResponse
```

Implements: [ServletResponse](#)

Implemented by: [HttpServletResponseWrapper](#)

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。

servlet 容器创建 [HttpServletResponse](#) 对象，并将该对象作为参数传递给 servlet 的 service 方法 (doGet、doPost，等等)。

总结：HttpServletResponse对象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。开发人员必须学会使用HttpServletResponse，才能给浏览器发送数据。

3.1.2 response常用API

3.1.2.1 API介绍

1. `void setStatus(int sc)` 设置此响应的状态代码
2. `void setHeader(String name, String value)` 用给定名称和值设置响应头

3.1.2.3 使用步骤

1. 创建ResponseServlet
2. 调用setHeader，设置消息头 ("Refresh"," 3;url=<http://www.jd.com>")
3. 调用setStatus，设置响应状态码

3.1.2.4 演示代码

```
package cn.itcast.web;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ResponseServlet",urlPatterns = "/response")
public class ResponseServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        //自动刷新操作，在3秒后跳转京东主页
        response.setHeader("Refresh"," 3;url=http://www.jd.com");
        //设置响应状态码
        response.setStatus(200);
    }
}
```

学会了简单消息头设置，是远远不够的，响应消息头的内容非常的多，接下来向大家介绍几个，需要掌握的常见的响应头。

3.2 常见的响应头介绍

1. location :

重定向操作：通常告知浏览器马上向该地址发送请求，通常和响应码302 一起使用

2. content-encoding :

设置当前数据的压缩格式，告知浏览器以何种压缩格式解压数据

3. content-disposition :

通知浏览器以何种方式获取数据（直接解析数据（网页，图片文本），或者以附件方式（下载文件））

4. content-type :

实体头部用于指示资源的MIME类型（MIME类型：用于提示当前文件的媒体类型，例如图片——（image/png）、音频——（audio/ogg）。它的作用与传统上Windows上的文件扩展名相同。该名称源于最初用于电子邮件的MIME标准。）

注意：我们content-type常用的设置一般都是——“text/html; charset=utf-8”，其中“text/html;”——设置浏览器以文件格式解析数据；“charset=utf-8”——响应数据的编码表。

响应头虽然学习了，但是简单的介绍肯定记不住，因此，我们准了几个案例，让大家来练习。

3.3 案例1：使用location响应头实现跳转

3.3.1 使用location响应头实现重定向跳转页面

3.3.1.1 案例需求

使用location响应头实现重定向跳转淘宝主页www.taobao.com

3.3.1.2 案例效果

浏览器访问RedirectServlet之后跳转淘宝主页

3.3.1.3 案例分析

1. 创建servlet
2. 使用response对象，发送location消息头和302响应码给浏览器

3.3.1.4 代码实现

```
package cn.itcast.web;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "RedirectServlet", urlPatterns = {"/redirect"})
```

```
public class RedirectServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        //设置重定向响应头
        // response.setHeader("location", "http://www.taobao.com");
        //设置状态码
        //response.setStatus(302);
        //=====上面代码的简化方式=====
        response.sendRedirect("http://www.taobao.com");
    }
}
```

3.4 案例2：使用Content-Type响应头向浏览器输出中文数据无乱码方案

3.4.1 案例需求

向页面输出中文数据没有乱码

3.4.2 案例效果

你好中文世界

3.4.3 案例分析

1. 创建servlet
2. 使用response对象，调用setContentType方法传入参数：“text/html;charset=utf-8”
3. 使用response对象，向页面输出“你好中文世界”

3.4.4 代码实现

```
package cn.itcast.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

```
@WebServlet(name = "EncodingServlet",urlPatterns = "/encoding")
public class EncodingServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        response.getWriter().write("你好中文世界");
    }
}
```

3.5 案例3：以附件形式下载文件

3.5.1 案例需求

完成文件下载功能。

3.5.2 案例效果

用户点击页面的链接，浏览器开始下载文件。

3.5.3 案例分析

1. 创建一个页面，展示所有要被下载文件的链接
2. 链接将要下载的文件名称，发送给服务器的servlet，让servlet进行处理
3. 服务器加载文件资源
4. 提示浏览器，以下载的方式，获取服务器资源
5. 使用IO的方式，将文件数据输出到浏览器（response.getOutputStream();）

3.5.4 代码实现

1. html页面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>

        function isIE(){
            //获取当前浏览器相关信息
```



```

        var explorer = window.navigator.userAgent.toLowerCase() ;
        //判断是否是ie浏览器
        if (explorer.indexOf("msie") >= 0 || explorer.indexOf("rv:11.0) like
gecko") >= 0) {
            return true;
        }else {
            false;
        }
    }
}
window.onload = function () {
    if(isIE()){
        //在是IE浏览器的情况下，对中文请求参数编码
        var str = document.getElementById("ww").href;
        var str = encodeURIComponent(str);
        document.getElementById("ww").href = str;
    }
};
</script>
</head>
<body>
<a href="/day03/download?fileName=1.jpg">1.jpg</a>
<a href="/day03/download?fileName=2.jpg">2.jpg</a>
<a href="/day03/download?fileName=3.txt">3.txt</a>
<a id="ww" href="/day03/download?fileName=传智播客.txt">传智播客.txt</a>
<a href="/day03/download?fileName=heidisql.rar">heidisql.rar</a>

</body>
</html>

```

2. servlet演示代码

```

package cn.itcast.web;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.Base64;

@WebServlet(name = "DownloadServlet",urlPatterns = "/download")
public class DownloadServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

```



```
doGet(request, response);
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    //1 获取要下载的文件名称
    String fileName = request.getParameter("fileName");
    System.out.println(fileName);
    //2 加载当前文件
    // 注意：需要动态的获取当前文件的目录位置（即使服务器所在目录发生变化，我也可以获取到准确位置）

    // 我们需要使用servletContext，获取资源路径
    ServletContext context = getServletContext();
    String realPath = context.getRealPath("/download");
    File file = new File(realPath, fileName);

    //3 提示浏览器，以下载的方式，获取服务器资源
    //响应消息头设置：
    //Content-Type 设置文件媒体格式  getMimeType: 1.txt 2.jpg 获取文件的后缀名
    response.setContentType(getServletContext().getMimeType(fileName));

    //4 处理中文文件名乱码问题
    // 获取浏览器类型，通过请求头中的User-Agent来判断
    String ua = request.getHeader("User-Agent");
    boolean IE_LT11 = ua.contains("MSIE"); // IE11以下版本
    boolean IE11 = ua.contains("rv:11.0) like Gecko"); // IE11
    boolean Edge = ua.contains("Edge"); // win10自带的Edge浏览器
    // 如果是微软的浏览器，直接进行UTF-8编码
    if (IE_LT11 || IE11 || Edge) {
        fileName = URLEncoder.encode(fileName, "UTF-8");
        // java的编码方式和浏览器有略微的不同：对于空格，java编码后的结果是加号，
        // 而浏览器的编码结果是%20，因此将+替换成%20，这样浏览器才能正确解析空格
        fileName = fileName.replace("+", "%20");
    }
    // 标准浏览器使用Base64编码
    else {
        Base64.Encoder encoder = Base64.getEncoder();
        fileName = encoder.encodeToString(fileName.getBytes("utf-8"));
        // =?utf-8?B?文件名? 是告诉浏览器以Base64进行解码
        fileName = "=?utf-8?B?" + fileName + "?=";
    }
    //Content-Disposition 设置要被下载的文件名
    response.setHeader("Content-Disposition", "attachment;filename=" +
fileName);
    //5 将指定文件使用IO技术，向浏览器输出
    FileInputStream in = new FileInputStream(file);
    ServletOutputStream out = response.getOutputStream();

    //6 标准IO代码
    byte[] buf = new byte[1024];
    int len = -1;
    while((len = in.read(buf)) != -1) {
        out.write(buf, 0, len);
    }
}
```

```
    }  
    in.close();  
}  
}
```

3.6 案例4：验证码

3.6.1 案例需求

在页面展示登录验证码

3.6.2 案例效果

用户点击验证码图片可以更换新的验证码

3.6.3 案例分析

1. 创建一个登录页面页面，展示验证码图片
2. 配置今天资料中提供的servlet输出验证码图片
3. 设置页面的点击事件，触发点击事件就重新获取验证码图片

3.6.4 代码实现

1. html页面

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>Title</title>  
</head>  
<body>  
<form action="/day03/login" method="post">  
    <table>  
        <tr>  
            <td>用户名</td>  
            <td><input type="text" name="username"/></td>  
        </tr>  
        <tr>  
            <td>密码</td>  
            <td><input type="password" name="password"/></td>  
        </tr>  
        <tr>  
            <td>验证码: </td>  
            <td><input type="password" name="checkcode"/></td>  
        </tr>  
        <tr>  
            <td></td>  
            <td></td>  
        </tr>  
    </table>  
</form>
```



```
<td></td>
</tr>
<tr>
<td></td>
<td><input type="submit" value="登录"/></td>
</tr>
</table>
</form>
</body>
</html>
```

2. 配置验证码servlet

```
package cn.itcast.web;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.Random;

import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(urlPatterns = "/checkcode")
public class CheckcodeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // 创建画布
        int width = 120;
        int height = 40;
        BufferedImage bufferedImage = new BufferedImage(width, height,
            BufferedImage.TYPE_INT_RGB);
        // 获得画笔
        Graphics g = bufferedImage.getGraphics();
        // 填充背景颜色
        g.setColor(Color.white);
        g.fillRect(0, 0, width, height);
        // 绘制边框
        g.setColor(Color.red);
        g.drawRect(0, 0, width - 1, height - 1);
        // 生成随机字符
        // 准备数据
        String data =
            "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890";
```




```
// 准备随机对象
Random r = new Random();
// 声明一个变量 保存验证码
String code = "";
// 书写4个随机字符
for (int i = 0; i < 4; i++) {
    // 设置字体
    g.setFont(new Font("宋体", Font.BOLD, 28));
    // 设置随机颜色
    g.setColor(new Color(r.nextInt(255), r.nextInt(255), r.nextInt(255)));

    String str = data.charAt(r.nextInt(data.length())) + "";
    g.drawString(str, 10 + i * 28, 30);

    // 将新的字符 保存到验证码中
    code = code + str;
}
// 绘制干扰线
for (int i = 0; i < 6; i++) {
    // 设置随机颜色
    g.setColor(new Color(r.nextInt(255), r.nextInt(255), r.nextInt(255)));

    g.drawLine(r.nextInt(width), r.nextInt(height), r.nextInt(width),
r.nextInt(height));
}

// 将验证码 打印到控制台
System.out.println(code);

// 将验证码放到session中
request.getSession().setAttribute("code_session", code);

// 将画布显示在浏览器中
ImageIO.write(bufferedImage, "jpg", response.getOutputStream());
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    doGet(request, response);
}

}
```

3. 设置页面点击事件的js代码



```
<script>

    window.onload = function () {
        var img = document.getElementById("img");
        img.onclick=function () {
            img.src="/day03/checkcode?r="+new Date().getTime();
        };
    };

</script>
```

第4章 servletContext

在之前的文件下载案例中，我们使用过了一个对象：servletContext。虽然简单的使用过，但是，想要了解这个对象，我们还需要进一步的学习servletContext。

4.1 ServletContext的概述

ServletContext是一个容器（域对象）可以存储键值对数据（String key,Object value），保存在ServletContext中的数据不仅可以提供给所有的servlet使用，而且可以在整个项目范围内使用（后期的过滤器、监听器也可以使用ServletContext）。

4.2 ServletContext作为域对象

ServletContext中定义了很多方法，在javaweb阶段我最常用的就是将ServletContext作为容器（域对象）使用，因此，接下来我们要学习这个容器（域对象的）API。

4.2.1 API介绍

GenericServlet:

1. ServletContext `getServletContext()`
获取ServletContext对象

ServletContext:

1. void `setAttribute(String name, Object object)`
往servletcontext容器存入数据，name为数据名称，object为数据的值
2. Object `getAttribute(String name)`
从ServletContext中获取数据，根据指定的数据名称
3. void `removeAttribute(String name)`
从ServletContext中移除数据，根据指定的数据名称

4.2.2 使用步骤

1. 创建ServletContextServlet1和ServletContextServlet2

2. ServletContextServlet1调用存方法
3. ServletContextServlet2调用取方法
4. ServletContextServlet2调用删方法
5. ServletContextServlet2调用取方法

4.2.4 演示代码

ServletContextServlet1:

```
package cn.itcast.web;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ServletContextServlet1", urlPatterns = "/context1")
public class ServletContextServlet1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        //获取容器
        ServletContext context = getServletContext();
        //存入数据
        context.setAttribute("addr", "广州");
    }
}
```

ServletContextServlet2:

```
package cn.itcast.web;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@WebServlet(name = "ServletContextServlet2", urlPatterns = "/context2")
public class ServletContextServlet2 extends HttpServlet {
```



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

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    //获取容器
    ServletContext context = getServletContext();
    //获取数据
    String addr = (String) context.getAttribute("addr");
    System.out.println("存入之后，获取数据：" + addr);
    //移除数据
    context.removeAttribute("addr");
    //重新获取数据
    String addr2 = (String) context.getAttribute("addr");
    System.out.println("移除之后，获取数据：" + addr2);
}
}
```

代码准备好之后，分别访问context1和context2，我们会得到一下结果：

存入之后，获取数据：广州
移除之后，获取数据：null

通过这个结果，我们发现两次请求虽然访问了不同的servlet，但是依然能通过ServletContext共享数据，而且即使是由一个同学先访问，然后由另一个同学访问也是同样的结果。

总结：保存在ServletContext中的数据是项目全局共享的数据。

4.3 ServletContext作用总结

ServletContext对象一共两个作用：

1. 可以读取资源在当前项目中的文件位置
2. 可以作为域对象在项目全局范围内提供共享数据