今日内容

1. HTTP协议：响应消息
2. Response对象
3. ServletContext对象

HTTP协议：

请求消息：客户端发送给服务器端的数据

- 数据格式：
    1. 请求行
    2. 请求头
    3. 请求空行
    4. 请求体

响应消息：服务器端发送给客户端的数据

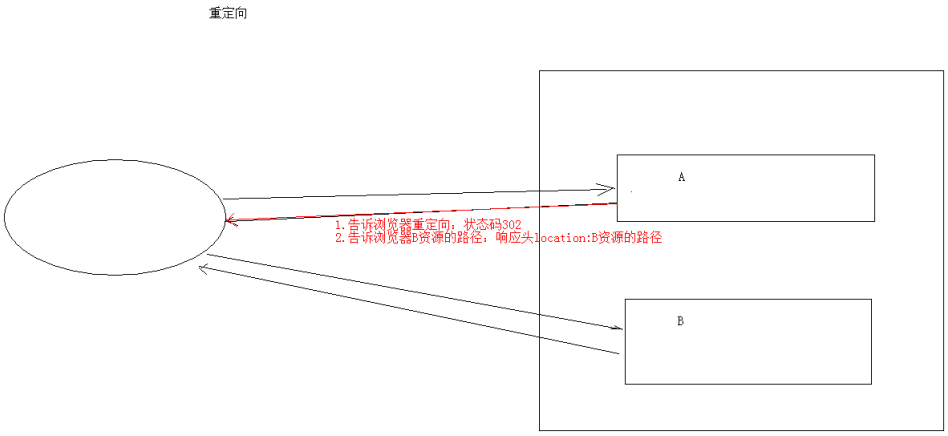
- 数据格式：

响应行

1. 组成：协议/版本 响应状态码 状态码描述
2. 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。
    1. 状态码都是3位数字
    2. 分类：

    （Http状态码）

        1. 1xx：服务器就收客户端消息，但没有接受完成，等待一段时间后，发送1xx多状态码。客户端给服务器发消息，服务器收了一半认为客户端还没发完，就等着，等了一会之后服务器就给客户端发消息，问还有没有发完
        2. 2xx：成功。代表：200
        3. 3xx：重定向。代表：302(重定向)，304(访问缓存)



4. 4xx：客户端错误。
   - 代表：
   - 404（请求路径没有对应的资源）
   - 405：请求方式和访问的方法不同(比如用get方式请求，代码却写到doPost里面)
5. 5xx：服务器端错误。代表：500(服务器内部出现异常)

响应头：

1. 格式：头名称： 值
2. 常见的响应头：
   1. 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
   2. Content-disposition：服务器告诉客户端以什么格式打开响应体数据
      - 值：
        - in-line:默认值,在当前页面内打开
        - attachment;filename=xxx：以附件形式打开响应体。文件下载

响应空行

响应体:传输的数据

- 响应字符串格式

```
HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF-8
Content-Length: 101
Date: Wed, 06 Jun 2018 07:08:42 GMT

<html>
  <head>
    <title>$Title$</title>
  </head>
  <body>
  hello , response
  </body>
</html>
```

Response对象

- 功能：设置响应消息

设置响应行

1. 格式：HTTP/1.1 200 ok
2. 设置状态码：setStatus(int sc)

设置响应头：setHeader(String name, String value)

设置响应体：

- 使用步骤：
    1. 获取输出流
        - 字符输出流：PrintWriter getWriter()
        - 字节输出流：ServletOutputStream getOutputStream()

使用输出流，将数据输出到客户端浏览器

案例一：重定向

- 设置状态码为302
- 设置响应头location

```
package cn.web.servlet;

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;


/*
* 重定向
* */
@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {


        System.out.println("demo1.....");

//          访问这个界面，会自动跳转到/responseDemo2
```

```
        /*设置状态码*/
        response.setStatus(302);
        /*设置响应头location*/
        response.setHeader("location","/ResponseTest/responseDemo2");
    }

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

```
package cn.web.servlet;

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;


/*
* 重定向
* */
@WebServlet("/responseDemo2")
public class ResposeDemo2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

//        访问/responseDemo1，会自动跳转到这里
        System.out.println("demo2.....");
    }

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

案例：简单方式的重定向

- 因为重定向的方式就是那一个套路，设置状态码-->设置响应头location，这个套路固定，所以有一个方法直接可以替代这个方法：response.sendRedirect(虚拟路径+资源)

```
package cn.web.servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```java
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;


/*
* 重定向
* */
@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {


        System.out.println("demo1.....");

        /*简单重定向*/
        response.sendRedirect("/ResponseTest/responseDemo2");

    }

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

```java
package cn.web.servlet;

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;


/*
* 重定向
* */
@WebServlet("/responseDemo2")
public class ResposeDemo2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

//          简单重定向
        System.out.println("demo2.....");
    }
```

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

- 重定向的特点和转发的特点对比(forward 和 redirect 区别)

重定向的特点:**redirect**	转发的特点：**forward**
地址栏发生变化	转发地址栏路径不变
重定向可以访问其他站点(服务器)的资源	转发只能访问当前服务器下的资源
重定向是两次请求。不能使用request对象来共享数据	转发是一次请求，可以使用request对象来共享数据

路径写法：

路径分类

1. 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 ./可以省略
        - ../:后退一级目录

绝对路径：通过绝对路径可以确定唯一资源

- 如：http://localhost/day15/responseDemo2，而这样写太麻烦了，因为前面这些都可以省略，所以我们直接从/day15/responseDemo2开始写，作为绝对路径，省略了协议，(IP)域名，端口号
- 以/开头的路径
- 规则：判断定义的路径是给谁用的？判断请求将来从哪儿发出
    - 给客户端浏览器使用：需要加虚拟目录(项目的访问路径)
        - 建议虚拟目录动态获取：request.getContextPath()，否则手写很可能有问题
        - 比如重定向的时候，请求是从客户端发出来的主动跳转的界面

```
package cn.web.servlet;
```

```java
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;


/*
* 重定向
* */
@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {


        System.out.println("demo1.....");

        /*简单重定向+动态获取虚拟目录*/

 response.sendRedirect(request.getContextPath()+"/responseDemo2");

    }

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

- 给服务器使用：不需要加虚拟目录
  - 转发路径，比如转发，那就是从服务器的

服务器输出字符数据到浏览器

- 步骤：
  1. 获取字符输出流：response.getWriter()
  2. 输出数据：response.getWriter().write()/response.getWriter().print()
- 注意：
  - 乱码问题：
    1. 浏览器使用的字符集默认是系统字符集，那么在中国就是gbk(gb2312)
    2. PrintWriter pw = response.getWriter();

       获取的流的默认编码是ISO-8859-1，而这个是不支持中文的
    3. 那么就好办了，设置该流的默认编码为gbk就可以了，但是注意，要在获取流之前就要设置流的编码，否则就没意义了:response.setCharacterEncoding()

4. 但是还有个问题，就是以后假如有别的浏览器不是gbk，那么就又乱码了，所以告诉浏览器响应体使用的编码：response.setHeader("content-type","text/html;charset=utf-8");

```
package cn.web.servlet;

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;
import java.io.PrintWriter;


/*
* 重定向
* */
@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

        /*设置流的编码为GBK*/
        response.setCharacterEncoding("utf-8");

        /*告诉浏览器本次的编码，要使用这个编码来解码*/
        response.setHeader("content-type","text/html;charset=utf-8");

        PrintWriter writer = response.getWriter();

        writer.write("<h1>你好</h1>");
        writer.print("<h1>Response</h1>");

    }

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

5. 但是还是有一些麻烦，所以有一个简单的形式来设置编码：

```
package cn.web.servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```java
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;


/*
* 重定向
* */
@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

        /*简单的方法来设置编码*/
        response.setContentType("text/html;charset=utf-8");

        PrintWriter writer = response.getWriter();

        writer.write("<h1>你好</h1>");
        writer.print("<h1>Response</h1>");

    }

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

6. 注意是要在获取流之前设置编码

服务器输出字节数据到浏览器

- 步骤:
    1. 获取字节输出流

       response.getOutputStream()
    2. 输出数据

       response.getOutputStream().write()

1. 验证码
    1. 本质：图片
    2. 目的：防止恶意表单注册

案例二：验证码

```java
package cn.web.servlet;
```

```
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;
import java.awt.*;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.Random;

@WebServlet("/checkCodeServlet")
public class CheckCodeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        int width   =100,
            height  =50;

        /*创建对象（验证码图片对象），在内存中
        * BufferedImage(x1,x2,x3);
        * x1：图片宽度
        * x2：图片高度
        * x3：图片的颜色类型，一般选择RGB三元色的图片类型
        * 图片默认为黑色
        * */
        BufferedImage bufferedImage = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);

        /*美化图片*/
            /*填充背景色，这里是填充*/
        Graphics graphics = bufferedImage.getGraphics();     //画笔对象
        graphics.setColor(Color.PINK);       //设置画笔颜色
        graphics.fillRect(0,0,width,height);     //填充,从(0,0)开始画，画width宽，画
height高
            /*画边框，这里是画*/
        graphics.setColor(Color.BLUE);
        graphics.drawRect(0,0,width-1,height-1);     //减去一，因为图画的边框本身就有
1px，如果不减去1，那么画边框就会画到外边去
            /*写验证码*/
        String str =
"ABCDEFGHIGKLMNOPQXYZUVWabcdefghigklmnopqxyzuvwxyz0123456789";

        Random random = new Random();

        for (int i = 1; i <= 4; i++) {
            int index = random.nextInt(str.length());//获取字符索引
            char c = str.charAt(index); //获取字符
```

```
            graphics.drawString(c+"",width/5*i,height/2);//把验证码写到位置为(x,y)
的地方
        }
            /*画干扰线*/
        graphics.setColor(Color.GRAY);

        for (int i = 0; i < 10; i++) {
            int a = random.nextInt(width);
            int b = random.nextInt(width);
            int c = random.nextInt(width);
            int d = random.nextInt(width);
            graphics.drawLine(a,b,c,d);
            graphics.drawLine(b,a,c,d);
            graphics.drawLine(b,c,a,d);
            graphics.drawLine(b,c,d,a);
        }

        /*将图片输出到任何流中显示
        * ImageIO.write(x1,x2,x3);
        * x1：图片
        * x2：后缀名
        * x3：流
        * */
        ImageIO.write(bufferedImage,"jpg",response.getOutputStream());

    }

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

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

    <img class="change" src="/ResponseTest/checkCodeServlet">
    <a class="change" href="">看不清..</a>


    <script>

        /*
        * 功能：点击一次就换一张图片
        * */
```

```
        function change() {
            var change = document.getElementsByClassName("change");
            change.onclick = function (ev) {
                /*因为服务器读取图片的时候会优先读取服务器的缓存，所以应该在这里加一个没有什么用的参数?xxx，但是?xxx每次读取的时候也不应该一样，所以要加时间戳*/
                var date = new Date();
                image.src = "/ResponseTest/checkCodeServlet?"+date;
            }
        }

    </script>

</body>
</html>
```

评价：丑极了

所以当在开发的时候显示验证码的时候，一般从网上找些好看的代码改改就成了，能看懂就行

ServletContext对象：

概念

代表整个web应用，可以和程序的容器(服务器)来通信

获取：

1. 通过request对象获取 request.getServletContext();
2. 通过HttpServlet获取 this.getServletContext();

```
package cn.web.servlet;

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;
import java.io.PrintWriter;


@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        /*
        * ServletContext对象获取
        * */
```

```
        /*方法一*/
        ServletContext servletContext = request.getServletContext();
        /*方法二*/
        ServletContext servletContext2 = this.getServletContext();
    }

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

功能

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

  如：text/html image/jpeg
- 获取：String getMimeType(String file) ，其实是根据后缀名来获取的

```
package cn.web.servlet;

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;
import java.io.PrintWriter;


@WebServlet("/responseDemo1")
public class ResponseDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        /*
        * 获取MIME TYPE
        * */
        ServletContext servletContext = this.getServletContext();

        /*现在先假装获取到了这个文件名称...*/
        String file = "1.jpg";

        /*获取MIME类型*/
        String mimeType = servletContext.getMimeType(file);
```

```
        System.out.println(mimeType);    //mage/jpeg
    }

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

域对象：共享数据

1. setAttribute(String name,Object value)
2. getAttribute(String name)
3. removeAttribute(String name)

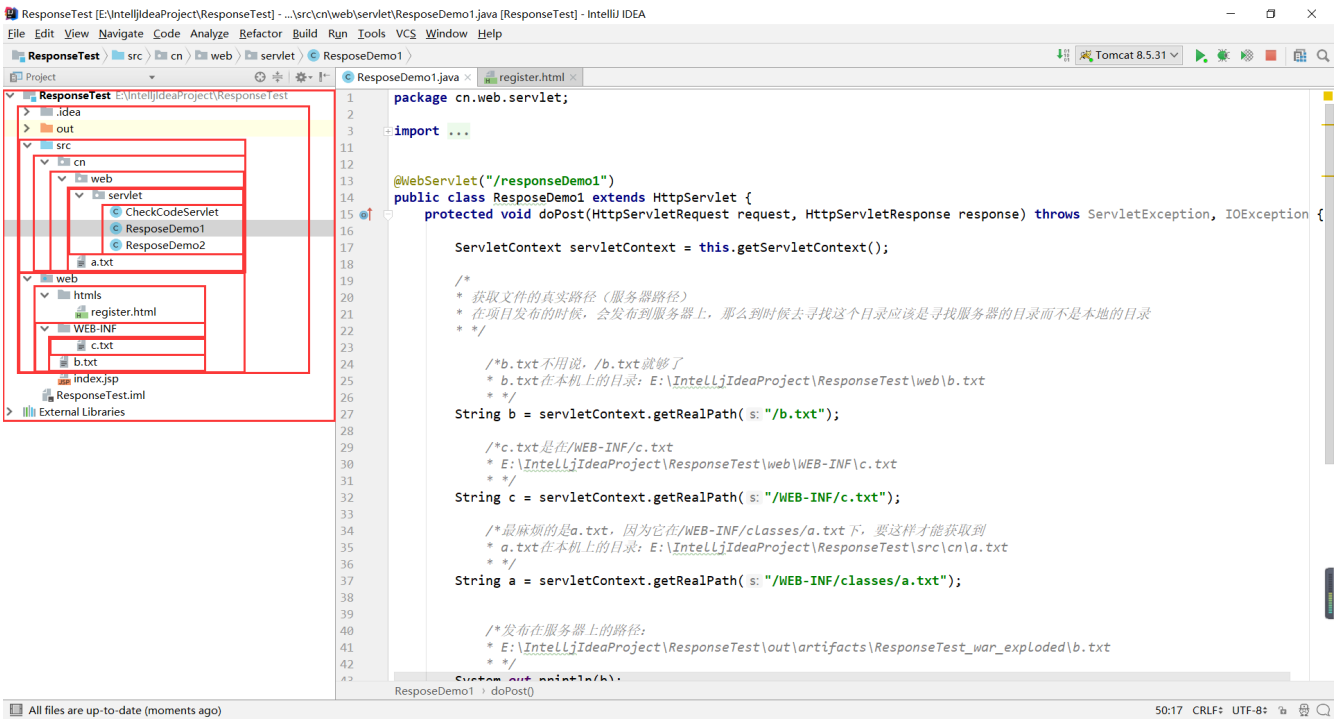
- ServletContext对象范围：所有用户所有请求的数据都可以，这个就不专门做代码演示了

获取文件的真实路径（是服务器上的路径）

1. 方法：String getRealPath(String path)

```
package cn.web.servlet;

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;
import java.io.PrintWriter;


@WebServlet("/responseDemo1")
public class ResposeDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        ServletContext servletContext = this.getServletContext();

        /*
        * 获取文件的真实路径（服务器路径）
        * 在项目发布的时候，会发布到服务器上，那么到时候去寻找这个目录应该是寻找服务器的目录而
不是本地的目录
        * */

            /*b.txt不用说，/b.txt就够了
            * b.txt在本机上的目录：E:\IntelljIdeaProject\ResponseTest\web\b.txt
            * */
        String b = servletContext.getRealPath("/b.txt");
```

```
            /*c.txt是在/WEB-INF/c.txt
            * E:\IntelljIdeaProject\ResponseTest\web\WEB-INF\c.txt
            * */
        String c = servletContext.getRealPath("/WEB-INF/c.txt");

            /*最麻烦的是a.txt，因为它在/WEB-INF/classes/a.txt下，要这样才能获取到
            * a.txt在本机上的目录：E:\IntelljIdeaProject\ResponseTest\src\cn\a.txt
            * */
        String a = servletContext.getRealPath("/WEB-INF/classes/a.txt");


            /*发布在服务器上的路径：
            *
E:\IntelljIdeaProject\ResponseTest\out\artifacts\ResponseTest_war_exploded\b.txt
            * */
        System.out.println(b);
            /*发布在服务器上的路径：
            *
E:\IntelljIdeaProject\ResponseTest\out\artifacts\ResponseTest_war_exploded\WEB-
INF\c.txt
            * */
        System.out.println(c);
            /*发布在服务器上的路径：
            *
E:\IntelljIdeaProject\ResponseTest\out\artifacts\ResponseTest_war_exploded\WEB-
INF\classes\a.txt
            * */
        System.out.println(a);
    }

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



案例：

- 文件下载需求：
    1. 页面显示超链接
    2. 点击超链接后弹出下载提示框
    3. 完成图片文件下载
- 分析：
    1. 超链接指向的资源如果能够被浏览器解析，则在浏览器中展示，如果不能解析，则弹出下载提示框。不满足需求
    2. 任何资源都必须弹出下载提示框
    3. 使用响应头设置资源的打开方式：
        - content-disposition:attachment;filename=xxx
- 步骤：
    1. 定义页面，编辑超链接href属性，指向Servlet，传递资源名称filename
    2. 定义Servlet
        1. 获取文件名称
        2. 使用字节输入流加载文件进内存
        3. 指定response的响应头： content-disposition:attachment;filename=xxx
        4. 将数据写出到response输出流
- 问题：
    - 中文文件问题
        - 解决思路：
            1. 获取客户端使用的浏览器版本信息

2. 根据不同的版本信息，设置filename的编码方式不同

```
package cn.web.utils;

import sun.misc.BASE64Encoder;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;


public class DownLoadUtils {

    public static String getFileName(String agent, String filename) throws
UnsupportedEncodingException {
        if (agent.contains("MSIE")) {
            // IE浏览器
            filename = URLEncoder.encode(filename, "utf-8");
            filename = filename.replace("+", " ");
        } else if (agent.contains("Firefox")) {
            // 火狐浏览器
            BASE64Encoder base64Encoder = new BASE64Encoder();
            filename = "=?utf-8?B?" +
base64Encoder.encode(filename.getBytes("utf-8")) + "?=";
        } else {
            // 其它浏览器
            filename = URLEncoder.encode(filename, "utf-8");
        }
        return filename;
    }
}
```

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

<a id="0_jpg" href="/ResponseTest/checkCodeServlet?resource=0.jpg">图片0</a>
<a id="1_jpg" href="/ResponseTest/checkCodeServlet?resource=1.jpg">图片1</a>

</body>
</html>
```

```
package cn.web.servlet;


import javax.imageio.ImageIO;
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.awt.*;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.LinkedList;
import java.util.Random;

@WebServlet("/checkCodeServlet")
public class CheckCodeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {



        /*获取参数值*/
        String resource = request.getParameter("resource");

        /*获取context*/
        ServletContext context = this.getServletContext();

        /*获取文件的真实路径*/
        String realPath = context.getRealPath( "/resource/" + resource);

        /*创建文件字节输入流读取文件进入内存*/
        FileInputStream fileInputStream = new FileInputStream(new
File(realPath));

        byte[] bytes = new byte[1024*8];
        int value;

        /*设置response响应头*/
            /*设置响应头类型*/
        String mimeType = context.getMimeType(resource);
        response.setHeader("content-type",mimeType);
            /*设置打开方式*/
        response.setHeader("content-
disposition","attachment;filename="+resource);


        ServletOutputStream outputStream = response.getOutputStream();
```

```
            while ((value=fileInputStream.read(bytes))!=-1){
                outputStream.write(bytes,0,value);
            }

        }

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