**验证码图片生成步骤**

1. 创建BufferedImage对象。
2. 获取BufferedImage的画笔，即调用getGraphics()方法获取Graphics对象。
3. 调用Graphics对象的setColor()方法和fillRect()方法设置图片背景颜色。
4. 调用Graphics对象的setColor()方法和drawLine()方法设置图片干扰线。
5. 调用BufferedImaged对象的setRGB()方法设置图片的噪点。
6. 调用Graphics对象的setColor()方法、setFont()方法和drawString()方法设置图片验证码。

因为验证码的图片的宽度和高度要根据网站的风格来确定的，所以字体的大小需要根据图片的宽度和高度来确定，用到了小小的技巧。

package util;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.util.Random;

import javax.imageio.ImageIO;

public class Verification {

private static final String ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890";

/\*\*

\* 生成一个宽为width， 高为height， 验证码为code的图片

\* @param width 图片的宽

\* @param height 图片的高

\* @param code 验证码字符串

\* @return 返回图片验证码

\*/

public static BufferedImage getImage(int width, int height, String code){

return getImage(width, height, code, 20);

}

/\*\*

\* 生成一个宽为width， 高为height， 验证码为code的图片，图片中干扰线的条数为lineCnt

\* @param width 图片的宽

\* @param height 图片的高

\* @param code 验证码字符串

\* @param lineCnt 干扰线的条数，建议为10条左右，可根据结果适当调整

\* @return 返回图片验证码

\*/

public static BufferedImage getImage(int width, int height, String code, int lineCnt){

return createImage(width, height, code, lineCnt, 0.01);

}

/\*\*

\* 生成一个宽为width， 高为height， 验证码为code的图片，图片中干扰线的条数为lineCnt

\* 噪声比为noiseRate，即图片中噪音像素点的百分比

\* @param width 图片的宽

\* @param height 图片的高

\* @param code 验证码字符串

\* @param lineCnt 干扰线的条数，建议为10条左右，可根据结果适当调整

\* @param noiseRate 图片中噪音像素点占总像素的百分比

\* @return 返回图片验证码

\*/

public static BufferedImage getImage(int width, int height, String code, int lineCnt, double noiseRate){

return createImage(width, height, code, lineCnt, noiseRate);

}

/\*\*

\*

\* 生成一个宽为width， 高为height， 验证码为code的图片，图片中干扰线的条数为lineCnt

\* 噪声比为noiseRate，即图片中噪音像素点的百分比

\* @param width 图片的宽

\* @param height 图片的高

\* @param code 验证码字符串

\* @param lineCnt 干扰线的条数，建议为10条左右，可根据结果适当调整

\* @param noiseRate 图片中噪音像素点占总像素的百分比

\* @return 返回图片验证码

\*/

private static BufferedImage createImage(int width, int height, String code, int lineCnt, double noiseRate){

int fontWidth = ((int)(width \* 0.8)) / code.length();

int fontHeight = (int)(height \* 0.7);

//为了在任意的width和height下都能生成良好的验证码，

//字体的大小为fontWdith何fontHeight中的小者，

int fontSize = Math.min(fontWidth, fontHeight);

//drawString时要用到

int paddingX = (int) (width \* 0.1);

int paddingY = height - (height - fontSize) / 2;

//创建图像

BufferedImage buffimg = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

//获得画笔

Graphics g = buffimg.getGraphics();

//设置画笔的颜色

g.setColor(getRandColor(200, 255));

//然后填充一个矩形，即设置背景色

g.fillRect(0, 0, width, height);

// 设置干扰线

for (int i = 0; i < lineCnt; i++) {

//随机获取干扰线的起点和终点

int xs = (int)(Math.random() \* width);

int ys = (int)(Math.random() \* height);

int xe = (int)(Math.random() \* width);

int ye = (int)(Math.random() \* height);

g.setColor(getRandColor(1, 255));

g.drawLine(xs, ys, xe, ye);

}

// 添加噪点

int area = (int) (noiseRate \* width \* height);

for(int i=0; i<area; ++i){

int x = (int)(Math.random() \* width);

int y = (int)(Math.random() \* height);

buffimg.setRGB(x, y, (int)(Math.random() \* 255));

}

//设置字体

Font font = new Font("Ravie", Font.PLAIN, fontSize);

g.setFont(font);

for(int i=0; i<code.length(); ++i){

String ch = code.substring(i, i+1);

g.setColor(getRandColor(1, 199));

g.drawString(ch, paddingX + fontWidth \* i, paddingY);

}

return buffimg;

}

/\*\*

\* 获取随机的颜色，r,g,b的取值在L到R之间

\* @param L 左区间

\* @param R 右区间

\* @return 返回随机颜色值

\*/

private static Color getRandColor(int L, int R){

if(L > 255)

L = 255;

if(R > 255)

R = 255;

if(L < 0)

L = 0;

if(R < 0)

R = 0;

int interval = R - L;

int r = L + (int)(Math.random() \* interval);

int g = L + (int)(Math.random() \* interval);

int b = L + (int)(Math.random() \* interval);

return new Color(r, g, b);

}

/\*\*

\* 随机生成若干个由大小写字母和数字组成的字符串

\* @param len 随机生成len个字符

\* @return 返回随机生成的若干个由大小写字母和数字组成的字符串

\*/

public static String getRandCode(int len){

String code = "";

for(int i=0; i<len; ++i){

int index = (int)(Math.random() \* ALPHABET.length());

code = code + ALPHABET.charAt(index);

}

return code;

}

/\*\*

\* 将图片转为byte数组

\* @param image 图片

\* @return 返回byte数组

\* @throws IOException

\*/

public static byte[] getByteArray(BufferedImage image) throws IOException{

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ImageIO.write(image, "png", baos);

return baos.toByteArray();

//ByteArrayOutputStream 不需要close

}

}

**使用验证码图片**

在verificationCode.java这个servlet中调用上面的类生成验证码图片，然后将图片返回给客户端。

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

HttpSession session = request.getSession();

//随机生成字符串，并写入session

String code = Verification.getRandCode(4);

session.setAttribute("verification", code);

BufferedImage image = util.Verification.getImage(100,30, code, 5);

response.setContentType("image/png");

OutputStream out = response.getOutputStream();

out.write(util.Verification.getByteArray(image));

out.flush();

out.close();

}

在index.jsp中设置验证码，用户点击验证码时，调用js代码请求服务器得到新的验证码。因为上面的那个生成验证码的servlet会被浏览器缓存，所以js代码中需要给该servlet一个随机的参数，这样浏览器就会向服务器发请求得到新的验证码，而不是去缓存中读取。

<%@page import="util.Verification"%>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Insert title here</title>

<script type="text/javascript">

function refreshcode(){

document.getElementById("verification").src= "/verificationCode/verificationCode?hehe="+Math.random();

}

</script>

</head>

<body>

<form action="<%=request.getContextPath()+"/checkVerification" %>" method="post">

验证码:<input type="text" name="submitVerification">

<img id="verification" alt="" title="看不清点击刷新验证码" src="<%=request.getContextPath()+"/verificationCode" %>"

onclick="refreshcode()"><br>

<input type="submit" name="submit" value="提交">

</form>

</body>

</html>

最后是在checkVerification.java这个servlet中判断用户输入的验证码是否正确，为了方便用户，验证码一般都设置成大小写不敏感，所以要先转化为小写字母再比对。

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

HttpSession session = request.getSession();

String verification = (String)session.getAttribute("verification");

String submitVerification = request.getParameter("submitVerification");

PrintWriter out = response.getWriter();

if(verification!=null && submitVerification!=null){

if(verification.toLowerCase().equals(submitVerification.toLowerCase())){

out.println("yes!!!");

}

else{

out.println("no!!!");

}

}

else{

out.println("no!!!");

}

session.removeAttribute("verification");//防止用户重复提交表单

}

/\*\*

\* @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);

}