JSP生成静态HTML页面的几种方法

2008年04月12日 星期六 09:26

|  |
| --- |
| 一、从数据库中取相应数据并替换掉模板中的对应标签，下面是一个简单的示例    1.buildhtml.jsp  <%@ page contentType="text/html; charset=gb2312" import="java.util.\*,java.io.\*"%>  <%  try{  String title="This is Title";  String content="This is Content Area";  String editer="LaoMao";  String filePath = "";  filePath = request.getRealPath("/")+"test/template.htm";  //out.print(filePath+"<br>");  String templateContent="";  FileInputStream fileinputstream = new FileInputStream(filePath);//读取模块文件  int lenght = fileinputstream.available();  byte bytes[] = new byte[lenght];  fileinputstream.read(bytes);  fileinputstream.close();  templateContent = new String(bytes);  //out.print(templateContent);  templateContent=templateContent.replaceAll("###title###",title);  templateContent=templateContent.replaceAll("###content###",content);  templateContent=templateContent.replaceAll("###author###",editer);//替换掉模块中相应的地方  //out.print(templateContent);  // 根据时间得文件名  Calendar calendar = Calendar.getInstance();  String fileame = String.valueOf(calendar.getTimeInMillis()) +".html";  fileame = request.getRealPath("/")+fileame;//生成的html文件保存路径  FileOutputStream fileoutputstream = new FileOutputStream(fileame);//建立文件输出流  byte tag\_bytes[] = templateContent.getBytes();  fileoutputstream.write(tag\_bytes);  fileoutputstream.close();  }  catch(Exception e){  out.print(e.toString());  }  %>   2. template.htm  <html>  <head>  <title>###title###</title>  <meta http-equiv="Content-Type" content="text/html; charset=gb2312">  <LINK href="../css.css" rel=stylesheet type=text/css>  </head>   <body>  <table width="500" border="0" align="center" cellpadding="0" cellspacing="2">     <tr>        <td align="center">###title###</td>     </tr>     <tr>        <td align="center">author：###author###&nbsp;&nbsp;</td>     </tr>     <tr>       <td>###content###</td>     </tr>  </table>  </body>  </html>    =======================================================   二、从动态页的URL获取相应页面内容并写入到文件   /\*  \* Created on 2006-3-4  \* To change the template for this generated file go to  \* Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments  \*/  package com.easydone.cn.tools.utils;   import java.io.BufferedReader;  import java.io.File;  import java.io.FileOutputStream;  import java.io.InputStream;  import java.io.InputStreamReader;  import java.io.PrintWriter;  import java.net.HttpURLConnection;  import java.net.URL;  import java.util.Date;   /\*\*  \* @author Administrator  \* To change the template for this generated type comment go to  \* Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments  \*/  public class MakeHtml {           private static long star = 0;           private static long end = 0;           private static long ttime = 0;            //返回html代码           public static String getHtmlCode(String httpUrl){           Date before = new Date();           star = before.getTime();           String htmlCode = "";           try {           InputStream   in;           URL url = new java.net.URL(httpUrl);           HttpURLConnection connection = (HttpURLConnection)url.openConnection();           connection = (HttpURLConnection) url.openConnection();           connection.setRequestProperty("User-Agent","Mozilla/4.0");           connection.connect();           in = connection.getInputStream();           java.io.BufferedReader breader = new BufferedReader(new InputStreamReader(in , "GBK"));           String currentLine;             while((currentLine=breader.readLine())!=null){           htmlCode+=currentLine;             }           } catch (Exception e) {           e.printStackTrace();           }finally{           Date after = new Date();           end = after.getTime();           ttime = end-star ;           System.out.println("执行时间:"+ttime +"秒");           }           return htmlCode;           }           //存储文件           public static synchronized void writeHtml(String filePath,String info,String flag) {            PrintWriter pw = null;           try {           File writeFile = new File(filePath);           boolean isExit = writeFile.exists();           if (isExit != true) {           writeFile.createNewFile();           } else {           if (!flag.equals("NO")) {           writeFile.delete();           writeFile.createNewFile();           }            }           pw = new PrintWriter(new FileOutputStream(filePath, true));           pw.println(info);           pw.close();           } catch (Exception ex) {           System.out.println(ex.getMessage());           }finally{           pw.close();           }           }            public static void main(String[] args) {           String url = "http://www.easydone.cn/index.htm";           writeHtml("c:/demo.htm",getHtmlCode(url),"NO");           }  }   三、利用Filter和定制Response，把服务器返回的JSP响应输出到我们自己的Response中，就可以将响应快速写入Html文件，然后再发送给客户。   import java.io.\*;  import javax.servlet.\*;  import javax.servlet.http.\*;  import java.util.Calendar;   public class CacheFilter implements Filter {     ServletContext sc;     FilterConfig fc;     long cacheTimeout = Long.MAX\_VALUE;      public void doFilter(ServletRequest req,                          ServletResponse res,                          FilterChain chain)         throws IOException, ServletException {       HttpServletRequest request =           (HttpServletRequest) req;       HttpServletResponse response =           (HttpServletResponse) res;        // check if was a resource that shouldn't be cached.       String r = sc.getRealPath("");       String path =            fc.getInitParameter(request.getRequestURI());       if (path!= null && path.equals("nocache")) {         chain.doFilter(request, response);         return;       }       path = r+path;        String id = request.getRequestURI() +            request.getQueryString();       File tempDir = (File)sc.getAttribute(         "javax.servlet.context.tempdir");        // get possible cache       String temp = tempDir.getAbsolutePath();       File file = new File(temp+id);        // get current resource       if (path == null) {         path = sc.getRealPath(request.getRequestURI());       }       File current = new File(path);        try {         long now =           Calendar.getInstance().getTimeInMillis();         //set timestamp check         if (!file.exists() || (file.exists() &&             current.lastModified() > file.lastModified()) ||             cacheTimeout < now - file.lastModified()) {           String name = file.getAbsolutePath();           name =               name.substring(0,name.lastIndexOf("/"));           new File(name).mkdirs();           ByteArrayOutputStream baos =               new ByteArrayOutputStream();           CacheResponseWrapper wrappedResponse =             new CacheResponseWrapper(response, baos);           chain.doFilter(req, wrappedResponse);            FileOutputStream fos = new FileOutputStream(file);           fos.write(baos.toByteArray());           fos.flush();           fos.close();         }       } catch (ServletException e) {         if (!file.exists()) {           throw new ServletException(e);         }       }       catch (IOException e) {         if (!file.exists()) {           throw e;         }       }        FileInputStream fis = new FileInputStream(file);       String mt = sc.getMimeType(request.getRequestURI());       response.setContentType(mt);       ServletOutputStream sos = res.getOutputStream();       for (int i = fis.read(); i!= -1; i = fis.read()) {         sos.write((byte)i);       }     }      public void init(FilterConfig filterConfig) {       this.fc = filterConfig;       String ct =           fc.getInitParameter("cacheTimeout");       if (ct != null) {         cacheTimeout = 60\*1000\*Long.parseLong(ct);       }       this.sc = filterConfig.getServletContext();     }      public void destroy() {       this.sc = null;       this.fc = null;     }  }   参考文章：   使用Filter实现静态HTML缓冲(一种折中方法)   缓冲是Web应用中必须考虑的一个提高性能的重要手段。对于基于JSP/Servlet技术的站点，常用的缓冲有持久层的数据库连接池缓冲，内存中的值对象缓冲，JSP页面缓冲，以及各种各样的缓冲框架等等，无不是为了提高系统的吞吐量。   然而对于大型站点来说，将JSP页面转换为静态Html也许是最高效的方法，特别适合于数据不经常变化但是页面访问量特别大的站点，如新闻等，通过把JSP动态页面预先转换为静态Html页面，当用户请求此页面时，系统自动导向到对应的Html页面，从而避免解析JSP请求，调用后台逻辑以及访问数据库等操作所带来的巨大开销。   如何将一个已有的JSP站点的动态JSP页面转化为静态Html呢？我们希望在不用更改现有Servlet，JSP的前提下让系统自动将这些JSP转换为Html页。幸运的是，Filter为我们提供了一种实现方案。   Filter是Servlet 2.2规范中最激动人心的特性。Filter能过滤特定URL如/admin/\*并进行必要的预处理，如修改Request和Response，从而实现定制的输入输出。更强大的是，Filter本身是一个责任链模式，它能一个接一个地传递下去，从而将实现不同功能的Filter串起来，并且可以动态组合。   要自动生成静态页面，用Filter截获jsp请求并先进行预处理，自动生成Html，是个不错的主意。一个很容易想到的方法是在Filter截获Request后，导向一个Servlet，在这个Servlet中向本机发送一个http请求，然后将响应写入一个文件：      URLConnection urlConn = URLConnection.open(http://localhost/req);    注意要避免递归。   另一个方法是不模拟http，而是定制Response，把服务器返回的JSP响应输出到我们自己的Response中，就可以将响应快速写入Html文件，然后再发送给客户。而且，由于没有http模拟请求，直接读取服务器响应速度非常快。   截获Response的关键便是实现一个WrappedResponse，让服务器将响应写入我们的WrappedResponse中。这类似于一个代理模式，Servlet 2.x已经提供了一个WrappedResponse类，我们只需要复写其中的一些关键方法即可。   WrappedResponse实现了Response接口，它需要一个Response作为构造函数的参数，事实上这正是代理模式的应用：WrappedResponse充当了代理角色，它会将JSP/Servlet容器的某些方法调用进行预处理，我们需要实现自己的方法。   综上：用Filter实现HTML缓冲的步骤是：   1. 用Filter截获请求，如/a.jsp?id=123，映射到对应的html文件名为/html/a.jsp$id=123.htm。  2. 查找是否有/html/a.jsp$id=123.htm，如果有，直接forward到此html，结束。  3. 如果没有，实现一个WrappedResponse，然后调用filterChain(request, wrappedResponse)。  4. 将返回的WrappedResponse写入文件/html/a.jsp$id=123.htm，然后返回响应给用户。  5. 下一次用户发送相同的请求时，到第2步就结束了。    使用这个方法的好处是不用更改现有的Servlet，JSP页，限制是，JSP页面结果不能与Session相关，需要登陆或用户定制的页面不能用这种方法缓冲。 |