[利用freemarker 静态化网页](http://blog.csdn.net/ajun_studio/article/details/6932185)

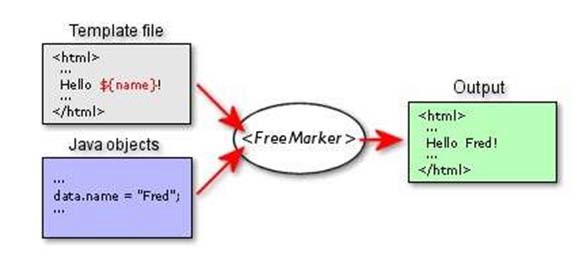
分类： [Java](http://blog.csdn.net/ajun_studio/article/category/824782) [设计](http://blog.csdn.net/ajun_studio/article/category/833284)2011-11-03 16:09 34736人阅读 [评论](http://blog.csdn.net/ajun_studio/article/details/6932185/#comments)(38) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/ajun_studio/article/details/6932185/#report)

[freemarker](http://www.csdn.net/tag/freemarker)[string](http://www.csdn.net/tag/string)[html](http://www.csdn.net/tag/html)[class](http://www.csdn.net/tag/class)[servlet](http://www.csdn.net/tag/servlet)[file](http://www.csdn.net/tag/file)

1、介绍-FreeMarker是什么

     模板引擎：一种基于模板的、用来生成输出文本的通用工具

     基于Java的开发包和类库



2、介绍-FreeMarker能做什么

     MVC框架中的View层组件

     Html页面静态化

    代码生成工具

    CMS模板引擎

  页面栏目动态定制

3、介绍-为什么要用FreeMarker

      程序逻辑(Java 程序)和页面设计(FreeMarker模板)分离

      分层清晰，利于分工合作

      主流Web框架良好的集成（struts2,springmvc）

      简单易学、功能强大

      免费开源

4、FreeMarker优点

     FreeMarker不依赖于Servlet，网络或Web 环境

     FreeMarker一开始就是为MVC设计的，它仅仅专注于展示

     你可以从任意位置加载模板；从类路径下，从数据库中等

     易于定义特设的宏和函数

5、上面简单介绍一下Freemarker，下面主要是利用Freemarker实习网页静态化的功能。

通过上面的介绍知道Freemarker是一种基于模板的、用来生成输出文本的通用工具，所以我们必须要定制符合自己业务的模板出来，然后生成的我们得html页面

Freemarker是通过freemarker.template.Configuration这个对象对模板进行加载的（它也处理创建和缓存预解析模板的工作），然后我们通过getTemplate方法获得你想要的模板，有一点要记住freemarker.template.Configuration在你整个应用必须保证唯一实例。

5.1、在Configuration 中可以使用下面的方法来方便建立三种模板加载

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **void** setDirectoryForTemplateLoading(File dir);

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **void** setClassForTemplateLoading(Class cl, String prefix);

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **void** setServletContextForTemplateLoading(Object servletContext, String path);

上述的第一种方法在磁盘的文件系统上设置了一个明确的目录，它确定了从哪里加载模板。不要说可能，File 参数肯定是一个存在的目录。否则，将会抛出异常。  
第二种调用方法使用了一个Class 类型的参数和一个前缀。这是让你来指定什么时候通过相同的机制来加载模板，不过是用Java 的ClassLoader 来加载类。这就意味着传  
入的Class 参数会被用来调用Class.getResource()方法来找到模板。参数prefix是给模板的名称来加前缀的。在实际运行的环境中，类加载机制是首选用来加载模板的方法，因为通常情况下，从类路径下加载文件的这种机制，要比从文件系统的特定目录位置加载安全而且简单。在最终的应用程序中，所有代码都使用.jar 文件打包也是不错的，这样用户就可以直接执行包含所有资源的.jar 文件了。  
第三种调用方式需要Web 应用的上下文和一个基路径作为参数，这个基路径是Web 应用根路径（WEB-INF 目录的上级目录）的相对路径。那么加载器将会从Web 应用目录开  
始加载模板。尽管加载方法对没有打包的.war 文件起作用， 因为它使用了ServletContext.getResource()方法来访问模板，注意这里我们指的是“目录”。如果忽略了第二个参数（或使用了””），那么就可以混合存储静态文件（.html，.jpg 等）和.ftl 文件，只是.ftl 文件可以被送到客户端执行。当然必须在WEB-INF/web.xml中配置一个Servlet 来处理URI 格式为\*.ftl 的用户请求，否则客户端无法获取到模板，因此你将会看到Web 服务器给出的秘密提示内容。在站点中不能使用空路径，这将成为一个问题，你应该在WEB-INF 目录下的某个位置存储模板文件，这样模板源文件就不会偶然  
void setDirectoryForTemplateLoading(File dir);  
void setClassForTemplateLoading(Class cl, String prefix);  
void setServletContextForTemplateLoading(Object  
servletContext, String path);  
地被执行到，这种机制对servlet 应用程序来加载模板来说，是非常好用的方式，而且模板可以自动更新而不需重启Web 应用程序，但是对于类加载机制，这样就行不通了。

5.2、从多个位置加载模板

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **import** freemarker.cache.\*; // 模板加载器在这个包下
2. ...
3. FileTemplateLoader ftl1 = **new** FileTemplateLoader(**new** File("/tmp/templates"));
4. FileTemplateLoader ftl2 = **new** FileTemplateLoader(**new** File("/usr/data/templates"));
5. ClassTemplateLoader ctl = **new** ClassTemplateLoader(getClass(),"");
6. TemplateLoader[] loaders = **new** TemplateLoader[] { ftl1, ftl2,ctl };
7. MultiTemplateLoader mtl = **new** MultiTemplateLoader(loaders);
8. cfg.setTemplateLoader(mtl);

现在，FreeMarker 将会尝试从/tmp/templates 目录加载模板，如果在这个目录下没有发现请求的模板，它就会继续尝试从/usr/data/templates 目录下加载，如果还是没有发现请求的模板，那么它就会使用类加载器来加载模板。

5.3、封装freemarker用于创建模板和加载模板

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.utils;
3. **import** java.io.IOException;
4. **import** java.io.Writer;
5. **import** java.util.Locale;
6. **import** java.util.Map;
8. **import** javax.servlet.ServletContext;
10. **import** freemarker.template.Configuration;
11. **import** freemarker.template.DefaultObjectWrapper;
12. **import** freemarker.template.Template;
13. **import** freemarker.template.TemplateException;
15. /\*\*
16. \* @author ajun
17. \* @http://blog.csdn.net/ajun\_studio
18. \*\*/
19. **public** **class** FreeMarkertUtil {
21. **private** **static**  Configuration config = **new** Configuration();
23. /\*\*
24. \* @param templateName 模板名字
25. \* @param root 模板根 用于在模板内输出结果集
26. \* @param out 输出对象 具体输出到哪里
27. \*/
28. **public** **static** **void** processTemplate(String templateName, Map<?,?> root, Writer out){
29. **try**{
30. //获得模板
31. Template template=config.getTemplate(templateName,"utf-8");
32. //生成文件（这里是我们是生成html）
33. template.process(root, out);
34. out.flush();
35. } **catch** (IOException e) {
36. e.printStackTrace();
37. } **catch** (TemplateException e) {
38. e.printStackTrace();
39. }**finally**{
40. **try** {
41. out.close();
42. out=**null**;
43. } **catch** (IOException e) {
44. e.printStackTrace();
45. }
46. }
47. }
48. /\*\*
49. \* 初始化模板配置
50. \* @param servletContext javax.servlet.ServletContext
51. \* @param templateDir 模板位置
52. \*/
53. **public** **static** **void** initConfig(ServletContext servletContext,String templateDir){
54. config.setLocale(Locale.CHINA);
55. config.setDefaultEncoding("utf-8");
56. config.setEncoding(Locale.CHINA, "utf-8");
57. config.setServletContextForTemplateLoading(servletContext, templateDir);
58. config.setObjectWrapper(**new** DefaultObjectWrapper());
59. }
60. }

5.4、例子介绍

会用freemarker.jar自己google下载吧。

这个例子中我们会Freemarker生成一个html文件 包括html的头部和尾部，已经body，这三个部分会分别对应三个模板文件，如下：

在模板内要想输出结果集 可以用类似于EL表达式输出${}

header.ftl

**[plain]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. companyName==>${h.companyName}<br/>
2. address==>${h.address}<br/>

footer.ftl

**[plain]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. des==>${f.des}<br/>
3. <a href="http://localhost/htmlpage/UpdateFooter.do"> 更新Footer </a>

body.ftl，这个模板include以上两个模板文件

**[plain]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2. <html>
3. <head>
4. <title>用户列表</title>
6. <meta http-equiv="pragma" content="no-cache">
7. <meta http-equiv="cache-control" content="no-cache">
8. <meta http-equiv="expires" content="0">
9. <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
10. <meta http-equiv="description" content="This is my page">
11. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
12. <!--
13. <link rel="stylesheet" type="text/css" href="styles.css">
14. -->
16. </head>
18. <body>
19. <#include "header.ftl" parse=true encoding="utf-8">
20. <hr/>
21. <a href="#">用户列表</a><br/>
22. <table border="1">
23. <tr>
24. <td>用户名</td>
25. <td>年龄</td>
26. <td>生日</td>
27. <td>id</td>
28. <td>操作</td>
29. </tr>
30. <#list users as user>
31. <tr>
32. <td>${user.name}</td>
33. <td>${user.age}</td>
34. <td>
35. ${user.birthday?string("yyyy-MM-dd HH:mm:ss")}
36. </td>
37. <td>${user.id}</td>
38. <td><a href="http://localhost/htmlpage/DelUser.do?id=${user.id}">删除</a></td>
39. </tr>
40. </#list>
42. </table>
43. <hr/>
44. <#include "footer.ftl" parse=true encoding="utf-8">
45. </body>
46. </html>

这三个模板对应的三个实体类

Footer.java

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.bean;
3. /\*\*
4. \* @author ajun
5. \* @http://blog.csdn.net/ajun\_studio
6. \*\*/
7. **public** **class** Footer {
9. **private** String des;
11. **public** String getDes() {
12. **return** des;
13. }
15. **public** **void** setDes(String des) {
16. **this**.des = des;
17. }

20. }

Header.java

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.bean;
2. /\*\*
3. \* @author ajun
4. \* @http://blog.csdn.net/ajun\_studio
5. \*\*/
6. **public** **class** Header {
8. **private** String companyName;
10. **private** String address;
12. **public** String getCompanyName() {
13. **return** companyName;
14. }
16. **public** **void** setCompanyName(String companyName) {
17. **this**.companyName = companyName;
18. }
20. **public** String getAddress() {
21. **return** address;
22. }
24. **public** **void** setAddress(String address) {
25. **this**.address = address;
26. }


30. }

User.java

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.bean;
3. **import** java.util.Date;
5. **public** **class** User {
7. **private** Integer id;
9. **private** String name ;
11. **private** **int** age;
13. **private** Date birthday;
15. **public** String getName() {
16. **return** name;
17. }
19. **public** **void** setName(String name) {
20. **this**.name = name;
21. }
23. **public** **int** getAge() {
24. **return** age;
25. }
27. **public** **void** setAge(**int** age) {
28. **this**.age = age;
29. }
31. **public** Date getBirthday() {
32. **return** birthday;
33. }
35. **public** **void** setBirthday(Date birthday) {
36. **this**.birthday = birthday;
37. }

40. **public** Integer getId() {
41. **return** id;
42. }
44. **public** **void** setId(Integer id) {
45. **this**.id = id;
46. }
48. **public** User(Integer id,String name, **int** age, Date birthday) {
49. **super**();
50. **this**.name = name;
51. **this**.age = age;
52. **this**.birthday = birthday;
53. **this**.id = id;
54. }
56. **public** User() {
57. **super**();
58. }

61. }

下面模板一些业务逻辑操作，对这三个实体类

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.service;
3. **import** com.ajun.template.bean.Footer;
4. /\*\*
5. \* @author ajun
6. \* @http://blog.csdn.net/ajun\_studio
7. \*\*/
8. **public** **class** FooterService {
10. **private** **static** Footer f = **new** Footer();
11. **static**{
12. f.setDes("北京-廊坊-好公司呢！！！！哇哈哈！！！");
13. }
15. **public** **static** **void** update(String des){
16. f.setDes(des);
17. }
19. **public** **static** Footer gerFooter(){
20. **return** f;
21. }
22. }

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.service;
3. **import** com.ajun.template.bean.Header;
4. /\*\*
5. \* @author ajun
6. \* @http://blog.csdn.net/ajun\_studio
7. \*\*/
8. **public** **class** HeaderService {
10. **private** **static** Header h = **new** Header();
12. **static**{
13. h.setAddress("北京朝阳CBD");
14. h.setCompanyName("上海唐秀！！！");
15. }
17. **public** **static** **void** update(String address,String companyName){
18. h.setAddress(address);
19. h.setCompanyName(companyName);
20. }
22. **public** **static** Header getHeader(){
23. **return** h;
24. }
25. }

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.service;
3. **import** java.util.ArrayList;
4. **import** java.util.Date;
5. **import** java.util.List;
7. **import** com.ajun.template.bean.User;
8. /\*\*
9. \* @author ajun
10. \* @http://blog.csdn.net/ajun\_studio
11. \*\*/
12. **public** **class** UserService {
14. **private** **static** List<User> users = **new** ArrayList<User>();
16. **static**{
17. **for**(**int** i=0;i<10;i++){
18. User u = **new** User(i,"ajun"+i,i+10,**new** Date());
19. users.add(u);
20. }
21. }
23. **public** **static** List<User> getUsers(){
24. **return** users;
25. }
27. **public** **static** **void** delete(**int** index){
28. **for**(**int** i=0 ;i<users.size();i++){
29. User u = users.get(i);
30. **if**(u.getId()==index){
31. users.remove(u);
32. //users.remove(index);
33. }
34. }
35. }
36. }

上面主要是模板你的一些业务和dao层得操作，因此没有涉及数据库的操作，主要是为实验。

生成html对外调用的方法，会用到FreeMarkertUtil这个类 这个类得代码上面已经给出。

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.client;
3. **import** java.io.Writer;
4. **import** java.util.HashMap;
5. **import** java.util.List;
6. **import** java.util.Map;
8. **import** com.ajun.template.bean.Footer;
9. **import** com.ajun.template.bean.Header;
10. **import** com.ajun.template.bean.User;
11. **import** com.ajun.template.service.FooterService;
12. **import** com.ajun.template.service.HeaderService;
13. **import** com.ajun.template.service.UserService;
14. **import** com.ajun.template.utils.FreeMarkertUtil;
16. /\*\*
17. \* @author ajun
18. \* @http://blog.csdn.net/ajun\_studio
19. \*\*/
20. **public** **class** ProcessClient {
22. **private** **static** Map<String,Object> root = **new** HashMap<String,Object>();
24. /\*\*
25. \* 调用FreeMarkertUtil.java
26. \* FreeMarkertUtil.processTemplate("body.ftl", root, out);
27. \* 来生成html文件
28. \* @param out
29. \*/
30. **public** **static** **void** processBody(Writer out){
31. Header h = HeaderService.getHeader();
32. root.put("h", h);
33. Footer f = FooterService.gerFooter();
34. root.put("f", f);
35. List<User> users = UserService.getUsers();
36. root.put("users", users);
37. FreeMarkertUtil.processTemplate("body.ftl", root, out);
38. }
40. }

此时我会提供一个servlet在客户端进行第一次请求的时候 我会调用这个ProcessClient来生成html页面，之后每次访问就可以直接访问html，来做到真正的静态化了

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.servlet;
3. **import** java.io.File;
4. **import** java.io.FileOutputStream;
5. **import** java.io.IOException;
6. **import** java.io.OutputStreamWriter;
7. **import** java.io.Writer;
9. **import** javax.servlet.ServletConfig;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.http.HttpServlet;
12. **import** javax.servlet.http.HttpServletRequest;
13. **import** javax.servlet.http.HttpServletResponse;
15. **import** com.ajun.template.client.ProcessClient;
16. **import** com.ajun.template.utils.DirectoryFilter;
17. **import** com.ajun.template.utils.FreeMarkertUtil;
19. /\*\*
20. \* @author ajun
21. \* @http://blog.csdn.net/ajun\_studio
22. \*\*/
23. **public** **class** Index **extends** HttpServlet {
25. **private** **static** **final** **long** serialVersionUID = 7474850489594438527L;
27. **public** Index() {
28. **super**();
29. }

32. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
33. **throws** ServletException, IOException {
35. **this**.doPost(request, response);
36. }

39. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)
40. **throws** ServletException, IOException {
41. //html生成之后存放的路径
42. String dirPath = request.getSession().getServletContext().getRealPath("/templates/html");
43. File path = **new** File(dirPath);
44. //生成的文件的名字
45. String indexFileName = "index.html";
46. /\*\*
47. \* 判断是否已经存在该html文件，存在了就直接访问html ，不存在生成html文件
48. \*/
49. String[] indexfileList = path.list(**new** DirectoryFilter(indexFileName));
50. **if**(indexfileList.length<=0){
51. Writer out = **new** OutputStreamWriter(**new** FileOutputStream(dirPath+"/"+indexFileName),"UTF-8");
52. //生成html文件
53. ProcessClient.processBody(out);
54. request.getRequestDispatcher("/templates/html/index.html").forward(request, response);
55. }**else**{
56. request.getRequestDispatcher("/templates/html/"+indexfileList[0]).forward(request, response);
57. }

60. }


64. /\*\*
65. \* 初始化模板配置，供以后获得模板，在init里加载也主要是为保证Configuration实例唯一
66. \*/
67. **public** **void** init(ServletConfig config) **throws** ServletException {
68. String templateDir = config.getInitParameter("templateDir");
69. FreeMarkertUtil.initConfig(config.getServletContext(), templateDir);
70. }

73. }

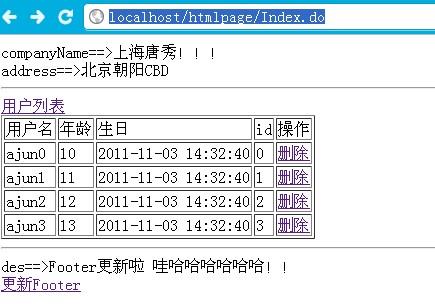
web.xml配置

**[html]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **<servlet>**
2. **<description>**This is the description of my J2EE component**</description>**
3. **<display-name>**This is the display name of my J2EE component**</display-name>**
4. **<servlet-name>**Index**</servlet-name>**
5. **<servlet-class>**com.ajun.template.servlet.Index**</servlet-class>**
6. **<init-param>**
7. **<param-name>**templateDir**</param-name>**模板存放位置，是基于app的根目录的
8. **<param-value>**/templates**</param-value>**
9. **</init-param>**
10. **<load-on-startup>**3**</load-on-startup>**为了启动的时候初始化模板配置
11. **</servlet>**
13. **<servlet-mapping>**
14. **<servlet-name>**Index**</servlet-name>**
15. **<url-pattern>**/Index.do**</url-pattern>**
16. **</servlet-mapping>**

部署到tomcat上，输入：<http://localhost/htmlpage/Index.do>

页面效果：



页面是做好了，但是内容变化了 ，更新怎么办呢，我这里是当列表内容变化之后 ，我是删除原来的html ，利用模板然后重新生成的符合新结果的html页面

当我删除一条的时候，我会重新生成html页面，但是由于浏览器缓存的原因，即是你删除了，重新生成了新html页面，可是浏览器还是保存原来的页面，不刷新两次是不行的，

这里我采用的没更新的时候，都会给这个html改个名字，让浏览器去加载最新的，就可以了

具体的删除操作如下：

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.servlet;
3. **import** java.io.File;
4. **import** java.io.FileOutputStream;
5. **import** java.io.IOException;
6. **import** java.io.OutputStreamWriter;
7. **import** java.io.Writer;
8. **import** java.util.UUID;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.http.HttpServlet;
12. **import** javax.servlet.http.HttpServletRequest;
13. **import** javax.servlet.http.HttpServletResponse;
15. **import** com.ajun.template.client.ProcessClient;
16. **import** com.ajun.template.service.UserService;
17. **import** com.ajun.template.utils.DirectoryFilter;
18. /\*\*
19. \* @author ajun
20. \* @http://blog.csdn.net/ajun\_studio
21. \*\*/
22. **public** **class** DelUser **extends** HttpServlet {

25. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response)
26. **throws** ServletException, IOException {
27. **this**.doPost(request, response);
28. }
30. //删除用户
31. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)
32. **throws** ServletException, IOException {
34. String id = request.getParameter("id");
35. UserService.delete(Integer.valueOf(id));
37. //生成html的位置
38. String dirPath = request.getSession().getServletContext().getRealPath("/templates/html");
39. //文件名字
40. String indexFileName = "index.html";
42. //删除原来的文件
43. delOldHtml(dirPath,indexFileName);
45. //防止浏览器缓存，用于重新生成新的html
46. UUID uuid = UUID.randomUUID();
47. Writer out = **new** OutputStreamWriter(**new** FileOutputStream(dirPath+"/"+uuid+indexFileName),"UTF-8");
48. ProcessClient.processBody(out);
49. response.sendRedirect("templates/html/"+uuid+"index.html");
50. }
52. /\*\*
53. \* 删除原来的html文件
54. \* @param htmlDir
55. \* @param htmlName
56. \*/
57. **private** **void** delOldHtml(String htmlDir,String htmlName){
58. File path = **new** File(htmlDir);
59. String[] indexfileList = path.list(**new** DirectoryFilter(htmlName));
60. **if**(indexfileList.length>=0){
61. **for**(String f:indexfileList){
62. File delf = **new** File(htmlDir+"/"+f);
63. delf.delete();
64. }
65. }
66. }
68. }

通过以上操作，每次更新html，就可以不解决浏览器缓存的问题了。

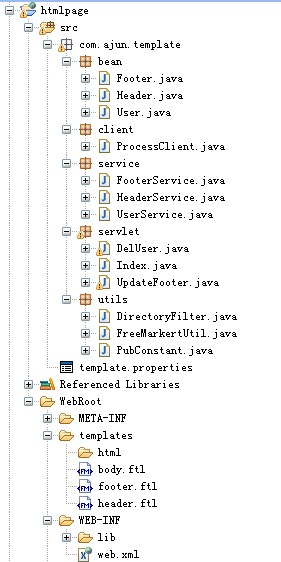
还有一个工具类需要介绍，就是判断是否已经生成了特定的html文件的java类

**[java]** [view plaincopyprint?](http://blog.csdn.net/ajun_studio/article/details/6932185/)

1. **package** com.ajun.template.utils;
3. **import** java.io.File;
4. **import** java.io.FilenameFilter;
5. /\*\*
6. \* @author ajun
7. \* @http://blog.csdn.net/ajun\_studio
8. \*\*/
9. **public** **class** DirectoryFilter **implements** FilenameFilter {
11. String myString;
12. **public** DirectoryFilter(String myString)
13. {
14. **this**.myString = myString;
15. }
17. **public** **boolean** accept(File dir,String name)
18. {   //FilenameFilter.accept(File dir, String name)
19. // 测试指定文件是否应该包含在某一文件列表中。
20. String f= **new** File(name).getName();
21. **if**(f.contains(myString) || f.equals(myString)){
22. **return** **true**;
23. }
24. **return** **false**;
25. }
27. }

到这里整个静态化就完成了，静态化更新机制，是根据你自己项目的业务进行定制的，可以定时生成html文件，也可以需要手动生成。

项目结构图如下：



记住：网站不是所有的页面都是需要静态化的，主要是一些实时性不是很高的数据页面进行静态化（来提高访问速度），其他都是通过伪静态来实现的，就是重写utl。  
  
页面静态化不是提高网站性能的唯一途径，还可以利用一些缓存产品来实现。

常用FreeMarker资源

官网主页：<http://www.freemarker.org/>

Eclipse插件JbossTool:<http://www.jboss.org/tools/download/>

中文文档:<https://sourceforge.net/projects/freemarker/files/chinese-manual/FreeMarker_Manual_zh_CN.pdf/download>