一、**SiteMesh**项目简介   
    OS(OpenSymphony)的**SiteMesh**是一个用来在JSP中实现页面布局和装饰（layout and decoration）的框架组件，能够帮助网站开发人员较容易实现页面中动态内容和静态装饰外观的分离。  
    **Sitemesh**是 由一个基于Web页面布局、装饰以及与现存Web应用整合的框架。它能帮助我们在由大量页面构成的项目中创建一致的页面布局和外观，如一致的导航条，一致 的banner，一致的版权，等等。它不仅仅能处理动态的内容，如jsp，php，asp等产生的内容，它也能处理静态的内容，如htm的内容，使得它的 内容也符合你的页面结构的要求。甚至于它能将HTML文件象include那样将该文件作为一个面板的形式嵌入到别的文件中去。所有的这些，都是GOF的 Decorator模式的最生动的实现。尽管它是由java语言来实现的，但它能与其他Web应用很好地集成。  
       官方：[http://www.opensymphony.com/**sitemesh**/](http://www.opensymphony.com/sitemesh/)  
       下载地址：[http://www.opensymphony.com/**sitemesh**/download.action](http://www.opensymphony.com/sitemesh/download.action) 目前的最新版本是Version 2.3；  
二、为什么要使用**SiteMesh**?  
    我们的团队开发J2EE应用的时候，经常会碰到一个比较头疼的问题：由于Web页面是由不同的人所开发，所以开发出来的界面通常是千奇百怪，通常让项目管 理人员苦笑不得。而实际上，任何一个项目都会要求界面的统一风格和美观，既然风格统一，那就说明UI层肯定有很多可以抽出来共用的静态或动态部分；如何整 合这些通用的静态或动态UI呢？Apache Tiles框架站了出来很好的解决了这一问题，再加上他与struts的完美集成，导致大小项目都把他作为UI层的首选框架，但是:Tiles确实有着它 很多的不足之处,下文我会介绍,本文想说的是，除了Apache Tiles框架,其实我们还有更好的解  
决方案，那就是:**SiteMesh**；  
本文  
   介绍了一个基于Web页面的布局、装饰以及应用整合的框架**Sitemesh**，它能帮助你为你的应用创建一致的外观，很好的取代Apache Tiles;  
三、**SiteMesh** VS Apache Tiles  
   用过struts的朋友应该对Apache Tiles的不会陌生，我曾经有一篇文章介绍过struts中tiles框架的组合与继承，现在怎么看怎么觉得复杂;  
    从使用角度来看，Tiles似乎是**Sitemesh**标签<page:applyDecorator>的一个翻版。其实**sitemesh**最强的一个特性是**sitemesh**将 decorator模式用在过滤器上。任何需要被装饰的页面都不知道它要被谁装饰，所以它就可以用来装璜来自php、asp、CGI等产生的页面了。你可 以定义若干个装饰器，根据参数动态地选择装饰器，产生动态的外观以满足你的需求。它也有一套功能强大的属性体系，它能帮助你构建功能强大而灵活的装饰器。 相比较而言，在这方面Tiles就逊色许多。  
    个人觉得在团队开发里面，Apache Tiles框架会导致所有人不仅仅要了解并且清楚Apache Tiles的存在，并且要特别熟悉每一个Tiles layout模板的作用，否则就可能出现用错模板的情况；除此之外，每个人涉及到的所有WEB页面都需要去配置文件里面逐个配置，不仅麻烦出错的几率还 高；而以上所有的不足都是**SiteMesh**所不存在的；  
四、**SiteMesh**的基本原理  
    一个请求到服务器后，如果该请求需要**sitemesh**装饰，服务器先解释被请求的资源，然后根据配置文件获得用于该请求的装饰器，最后用装饰器装饰被请求资源，将结果一同返回给客户端浏览器。  
五、如何使用**SiteMesh**  
   这里以struts2+spring2+hibernate3构架的系统为例  
     1、下载**SiteMesh**   
          下载地址：[http://www.opensymphony.com/**sitemesh**/download.action](http://www.opensymphony.com/sitemesh/download.action) 目前的最新版本是Version 2.3；  
     2、在工程中引入**SiteMesh**的必要jar包，和struts2-**sitemesh**-plugin-2.0.8.jar；  
     3、修改你的web.xml,在里面加入**sitemesh**的过滤器，示例代码如下：  
<!-- **sitemesh**配置 -->  
    <filter>  
        <filter-name>**sitemesh**</filter-name>  
        <filter-class>  
             com.opensymphony.module.**sitemesh**.filter.PageFilter  
        </filter-class>  
    </filter>  
    <filter-mapping>  
        <filter-name>**sitemesh**</filter-name>  
        <url-pattern>/\*</url-pattern>  
    </filter-mapping>  
    注意过滤器的位置：应该在struts2的org.apache.struts2.dispatcher.FilterDispatcher过滤器之前 org.apache.struts2.dispatcher.ActionContextCleanUp过滤器之后，否则会有问题；  
     4、在下载的**SiteMesh**包中找到**sitemesh**.xml，(\**sitemesh**-2.3\src\example-webapp\WEB-INF目录下就有)将其拷贝到/WEB-INF目录下；  
     5、在**sitemesh**.xml文件中有一个property结点(如下)，该结点指定了decorators.xml在工程中的位置，让**sitemesh**.xml能找到他;按照此路径新建decorators.xml文件，当然这个路径你可以任意改变，只要property结点的value值与其匹配就行；  
<property name="decorators-file" value="/WEB-INF/**sitemesh**/decorators.xml"/>  
     6、在WebRoot目录下新建decorators目录，并在该目录下新建一个模板jsp，根据具体项目风格编辑该模板，如下示例：我的模板：main.jsp  
<%@ page language="java" pageEncoding="UTF-8"%>  
<[%@taglib](mailto:%25@taglib) prefix="decorator"  
     uri="[http://www.opensymphony.com/**sitemesh**/decorator"%](http://www.opensymphony.com/sitemesh/decorator%22%25)>  
<[%@taglib](mailto:%25@taglib) prefix="page" uri="[http://www.opensymphony.com/**sitemesh**/page"%](http://www.opensymphony.com/sitemesh/page%22%25)>  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1 Transitional//EN"   
"<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">  
<%  
     response.setHeader("Pragma", "no-cache");  
     response.setHeader("Cache-Control", "no-cache");  
     response.setDateHeader("Expires", 0);  
%>  
<html>  
    <head>  
        <title><decorator:title default="kangxm test" />  
        </title>  
        <!-- 页面Head由引用模板的子页面来替换 -->  
        <decorator:head />  
    </head>  
    <body id="page-home">  
        <div id="page-total">  
            <div id="page-header">  
                <table width="100%" border="0" cellspacing="0" cellpadding="0">  
                    <tr>  
                        <td>  
                            <div class="topFunc">  
                                 我的账户  
                                 |  
                                 退出  
                            </div>  
                        </td>  
                    </tr>  
                </table>  
            </div>  
        </div>  
        <!-- end header -->  
        <!--   Menu Tag begin -->  
        <div id="page-menu" style="margin-top: 8px; margin-bottom: 8px;">  
            <div>  
                 这里放菜单  
            </div>  
        </div>  
        <!--   Menu Tag end -->  
        <div id="page-content" class="clearfix">  
            <center>  
                <table width="100%" border="0" cellpadding="0" cellspacing="0">  
                    <tr>  
                        <td>  
                            <decorator:body /><!-- 这里的内容由引用模板的子页面来替换 -->  
                        </td>  
                    </tr>  
                </table>  
            </center>  
        </div>  
        <!-- end content -->  
        <div id="page-footer" class="clearfix">  
             这里放页面底部  
            <!-- end footer -->  
        </div>  
        <!-- end page -->  
    </body>  
</html>  
这就是个简单的模板，页面的头和脚都由模板里的静态HTML决定了，主页面区域用的是<decorator:body />标签；  
也就是说凡是能进入过滤器的请求生成的页面都会默认加上模板上的头和脚，然后页面自身的内容将自动放到<decorator:body />标签所在位置；  
<decorator:title default="Welcome to test **sitemesh**!" />：读取被装饰页面的标题，并给出了默认标题。  
<decorator:head />：读取被装饰页面的<head>中的内容；  
<decorator:body />：读取被装饰页面的<body>中的内容；  
    7、说到这里大家就要想了，那如果某个特殊的需求请求路径在过滤器的范围内，但又不想使用模板怎么办？你总不能这么不讲道理吧！大家放心吧，**SiteMesh**早就考虑到这一点了，上面第5步说道的decorators.xml这个时候就起到作用了！下面是我的decorators.xml：  
<?xml version="1.0" encoding="ISO-8859-1"?>  
<decorators defaultdir="/decorators">  
    <!-- Any urls that are excluded will never be decorated by **Sitemesh** -->  
    <excludes>  
        <pattern>/index.jsp\*</pattern>  
          <pattern>/login/\*</pattern>  
    </excludes>  
    <decorator name="main" page="main.jsp">  
        <pattern>/\*</pattern>  
    </decorator>  
</decorators>  
decorators.xml有两个主要的结点：  
       decorator结点指定了模板的位置和文件名，通过pattern来指定哪些路径引用哪个模板  
       excludes结点则指定了哪些路径的请求不使用任何模板  
如上面代码，/index.jsp和凡是以/login/开头的请求路径一律不使用模板；  
另外还有一点要注意的是：decorators结点的defaultdir属性指定了模板文件存放的目录;  
六、实战感受  
    刚刚做完一个用到**sitemesh**的项目，跟以前用tiles框架相比，最大的感受就是简单，系统设计阶段  
就把模板文件和**sitemesh**框架搭好了！哪些页面使用框架哪些不使用，全部都通过UI Demo很快就定义出来了；  
在接下来的开发中所有成员几乎感受不到**sitemesh**的存在，各自仅仅关心自己的模块功能实现；  
七、总结  
    使用**sitemesh**给我们带来的是不仅仅是页面结构问题，它的出现让我们有更多的时间去关注底层业务  
逻辑，而不是整个页面的风格和结构。它让我们摆脱了大量用include方式复用页面尴尬局面，也避免了tiles  
框架在团队开发中的复杂度，它还提供了很大的灵活性以及给我们提供了整合异构Web系统页面的一种方案。