

Introduction à Sitemesh ou le layouting sans douleur

par Loïc Mathieu ([Accueil](#))

Date de publication : 17/09/2007

Dernière mise à jour : 17/09/2007

Cet article est un petit tutoriel introductif à Sitemesh, un framework Java/J2EE web permettant de facilement gérer le layout d'une application web. Dans cet article, je vais commencer par introduire les autres possibilités de gestion de layout (inclusion de JSP, Tiles), puis je présenterais Sitemesh et en donnerais un petit exemple pour finir sur ses fonctionnalités avancées.

- I - Introduction
- II - Quelques mots de Tiles
- III - L'approche de Sitemesh
- IV - Exemple
- V - Fonctionnalités avancées
 - V-A - Les DecoratorMapper
 - V-B - Le tag <content>
 - V-C - Accès avancés aux données de la page décorée (accès à l'objet Page)
 - V-D - Accéder à la requête d'une page décoré depuis un décorateur
- VI - Liens

I - Introduction

La gestion des layouts dans les applications web a toujours été pour moi un épine dans le pied jusqu'à ce que je découvre Sitemesh.

Une API de layouting, doit permettre d'éviter de copier/coller du code inutile pour décrire dans chaque page le header, la navigation, le footer, ... C'est ce que permet Sitemesh.

Avant Sitemesh, deux solutions :

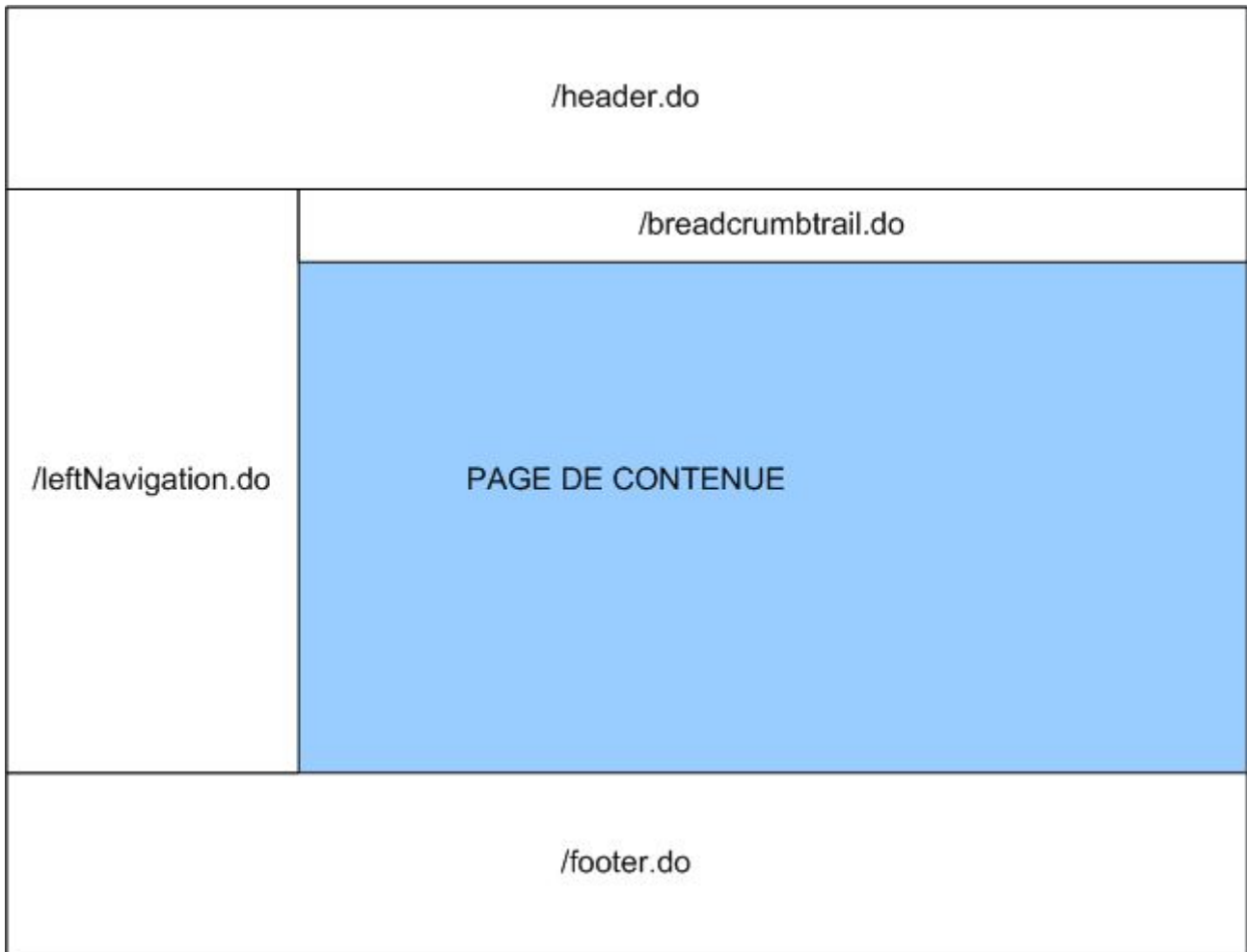
- Inclusion de JSP
- Tiles
- Cocoon

L'inclusion de JSP est basique et très très limitée car ne peut réellement tirer profit des framework MVC avec elle. Pour moi ce n'est pas une solution donc je ne vais pas en parler ici.

Le cas de Cocoon est beaucoup plus compliqué. Je ne connais pas trop ce framework mais il permet par un system de pipelines d'effectuer des transformations (XSL par exemple) sur une requête. Il a été créé dans un but de gestion de publication mais on pourrait aussi l'utiliser pour gérer un layout grâce à une transformation XSL de l'HTML généré. Ne connaissant pas trop ce framework, je n'en dirais pas plus, si cela vous intéresse allez donc jeter un coup d'oeil à la documentation officiel ([lien en fin de page](#)).

Sitemesh se base sur le pattern décorateur pour en gérer le layout des applications web. Mais qu'est-ce donc que cela me direz-vous? (ceux qui connaissent peuvent passer au paragraphe suivant) Le pattern Decorateur permet de modifier dynamiquement les comportements de certains objets en leur ajoutant de nouvelles fonctionnalités. Le pattern Decorateur fonctionne sur le principe des LEGO(R): on crée de nouveaux comportements en assemblant des modules, qui constituent les "briques" de notre application. On peut facilement chaîner les décorateurs pour pouvoir ajouter des fonctionnalités complexes grâce à des enchaînement de modules. Les décorateurs permettent aussi de coder des chaînes de traitement, l'idée essentielle est de pouvoir définir un traitement complexe comme un assemblage de traitements simples, offrant ainsi souplesse et modularité. Pour aller plus loin, l'article complet des design pattern de pcaboche est [ici](#) avec une bonne explication du design pattern decorateur (dont je me suis inspiré).

J'ai utilisé Sitemesh pour gérer le layout du site institutionnel d'une société de télécom. Voici un petit schéma du layout du site:



Exemple d'un layout de site utilisant Sitemesh

II - Quelques mots de Tiles

Très longtemps j'ai utilisé Tiles qui a de nombreux avantages :

- Intégration à Struts et Spring
- Peut être utilisé comme framework seul (ce n'est pas un MVC complet mais ça peut rendre pas mal de service pour une couche web basique)
- Configuration XML permettant l'héritage
- Possibilité de splitter le fichier de configuration en plusieurs parties
- Orientation composant avec un contrôleur par composant
- Bonne performance (un collègue à fait un test avec 1000 tiles imbriqués générés ... et ça n'a pris que quelques centaines de millisecondes à être affiché)

Mais qui, hélas, comprend aussi de nombreux inconvénients :

- Fichier de configuration peu structuré vite complexe et brouillon
- Pas de possibilité de partage de layouts entre différents sites
- Pas de gestion spécifique des headers HTML (voir les possibilités de Sitemesh dans ce domaine)
- Très vite, beaucoup d'héritage entre composants rendent le fichier de configuration très peu lisible
- Très verbeux en configuration
- Le principe de composant amène très vite à créer énormément de composants, donc énormément de JSP et le temps de première compilation d'une page en devient très long
- La gestion des erreurs au niveau d'un composant Tiles n'est pas aisée (impossible de faire une redirection web depuis un controller Tiles car à ce stade, la requête est déjà commitée)
- Utilisable uniquement avec des JSP

III - L'approche de Sitemesh

Sitemesh lui, a une approche fort différente: Sitemesh utilise le design pattern decorator pour ajouter des briques à votre page. Concrètement, vous ne développez que l'intérieur des pages, puis vous définissez dans Sitemesh un layout à utiliser dans lequel vous donnez les différents décorateurs qui vont aller ajouter des parties d'HTML (donc vont aller décorer) vos pages. La page interne que vous allez développer va être calculé, puis en fin de requête le filtre de Sitemesh va décorer votre page selon ce que vous avez configuré. Votre page va être parsé par Sitemesh et va pouvoir ensuite être accessible aux décorateurs. Bien sûr, Sitemesh permet la composition, vous pouvez décorer avec une page qui est elle-même décorée par une autre ... Mais attention aux effets de bord et aux risques de décoration 'en boucle'. (Page A, décoré par B, B est lui même décoré par C, C est lui même décoré par A ...)

La principale différence de Sitemesh se situe dans le fait que vos pages peuvent être créées avec n'importe quelle technologie. Par exemple, vous définissez une page /toto.do en Spring MVC (ou en Struts ou n'importe quel framework que vous aimez), vous utilisez ensuite le filtre Servlet de Sitemesh et vous lui dites d'intercepter les requêtes en *.do. Sitemesh, lorsque vous appelez /toto.do, va alors intercepter la requête et la décorer.

Dans Sitemesh, les décorateurs sont eux même des pages complètes (donc, /footer.do, /header.do, /navigation.do ou /toto.do peuvent être vues indépendamment, en HTML et peuvent toutes être des contrôleurs Spring MVC), ce qui permet de les développer et les tester indépendamment. Et en plus, Sitemesh peut décorer une page d'un site, par une page d'un autre site! Pour cela il suffit de définir la page du décorateur comme une page externe (donc commençant par http://), attention, cela peut engendrer des problèmes si vous utilisez de la réécriture d'URL car Sitemesh utilise un HttpClient pour accéder à ces pages (ouvre une session sur le serveur distant, demande la page, puis fait la décoration). Le fait que chaque page, décorée ou "décorante", ait un HTML complet peut ouvrir des horizons et des possibilités multiples, par exemple définir en XHTML une navigation qui pourra être réutilisable indépendamment depuis un site web en spring MVC via une décoration de Sitemesh ou depuis un composant flex ou ajax qui utiliserait la même navigation que vous ... vos composants de pages deviennent indépendants les uns des autres! Sitemesh n'opérant qu'en fin de requête, n'importe quelle technologie web peut être utilisée du côté JAVA. Sitemesh lui-même, pour les pages décorées, permet d'utiliser les technologies JSP, Velocity ou FreeMarker. Par contre, pour les décorateurs, aucune limite, on peut très bien, en plus des technologies JAVA, décorer ses pages par des HTML statiques, des pages PHP, des scripts CGI, ...

Dernière fonctionnalité évoquée la gestion des header. Comme les pages décorées et les décorateurs sont des pages HTML complètes, elles contiennent toutes un titre, des CSS,... Sitemesh permet donc, au niveau de la décoration, de spécifier des header par défaut puis de les réécrire par les valeurs des header de l'objet décoré. En fait, Sitemesh parse les différentes pages HTML et nous donne accès aux éléments head et body de la page pour pouvoir aller puiser dedans les informations qui nous intéressent.

En plus, le passage de paramètre par les header n'est pas limité aux header de la page finale. On peut très bien définir un paramètre dans les header de la page de contenu qui sera ensuite utilisé dans le body du décorateur. Voici un exemple :

Page décoré

```
<html>
  <meta name="author" content="test@example.com">
  <head>
    <title>Simple Document</title>
  </head>
  <body>
    Hello World! <br />
  </body>
</html>
```

Décorateur

```
<%@ taglib uri="sitemesh-decorator" prefix="decorator" %>
<decorator:usePage id="myPage" />
<html>
<head>
<title>My Site - <decorator:title default="Welcome!" /></title>
<decorator:head />
</head>
<body>
<h1><decorator:title default="Welcome!" /></h1>
<a href="mailto:<decorator:getProperty property="meta.author" default="staff@example.com" />">
<decorator:getProperty property="meta.author" default="staff@example.com" />
</a>
<decorator:body />
</body>
</html>
```

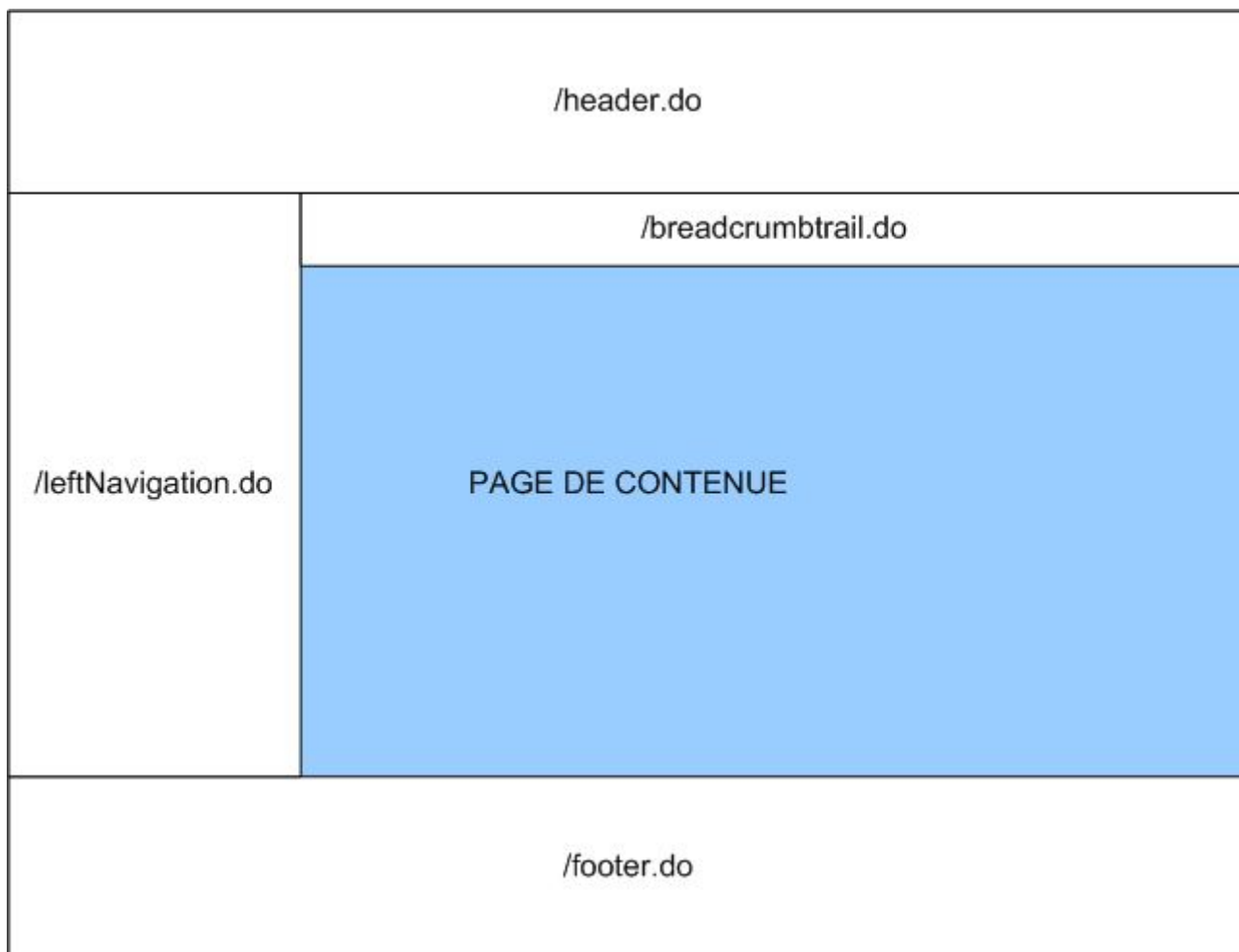
Bien sûr, plein d'autres fonctionnalités pour ce framework très simple d'utilisation et de principe mais très performant, allez voir sur le site. Je précise quand même que j'utilise Tiles depuis 3 ans et que je viens de découvrir Sitemesh que j'utilise pour un nouveau projet et j'en suis très content alors pourquoi pas vous (ça sonne comme une pub à la télé :=).

Petit récapitulatif Sitemesh:

- Pattern decorator qui permet la séparation entre le code et le layout, l'intégration du layout se faisant via un filtre en fin de requête
- Possibilité d'avoir les éléments du layout dans un site externe (donc de les partager entre plusieurs applications)
- Tout les éléments (éléments de layout et pages internes) sont des HTML complets et donc développable et testable indépendamment
- Agnostique par rapport à la technologie web utilisée (Java - Struts, Spring, ... -, PHP, CGI, ...)
- Agnostique par rapport à la technologie utilisé pour les pages décorées (JSP, Velocity, Freemarker)
- Gestion intégrée des différents header des pages et des décorateurs
- Nombreuses fonctionnalités avancés (voir plus bas)

IV - Exemple

Rappelons tout d'abord, le schéma du site:



Exemple d'un layout de site utilisant Sitemesh

Définition du filtre de servlet

```
<filter>
  <filter-name>sitemesh</filter-name>
  <filter-class>com.opensymphony.module.sitemesh.filter.PageFilter</filter-class>
</filter>
```

Définition du fichier de configuration de Sitemesh

```
<?xml version="1.0" encoding="utf-8"?>
<decorators defaultdir="/view/decorator">
  <decorator name="main" page="main.jsp">
    <pattern>*/</pattern>
    <excludes>
      <pattern>/ErrorPage</pattern>
    </excludes>
  </decorator>
  <decorator name="empty" page="empty.jsp"/>
</decorators>
```

Définition du fichier de configuration de Sitemesh

```
</decorators>
```

Ici, on définit la JSP `main.jsp` comme description de layout et on lui dit d'appliquer ce layout à toutes les pages à l'exclusion de la page `/ErrorPage`. Le decorator `empty` servant à pouvoir définir des décorateur qui ne décorent pas!

La page de layout

```
< ?xml version="1.0" encoding="utf-8"?>
< %@ taglib uri="http://www.opensymphony.com/sitemesh/decorator" prefix="decorator" %>
< %@ taglib uri="http://www.opensymphony.com/sitemesh/page" prefix="page" %>
< !DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
  <head>
    <title><decorator:title default="DEFAULT TITLE"/></decorator></title>
    <decorator:head/>
    <div><page:applyDecorator name="empty" page="/decorator/header"/></page></div>
    <div><decorator:body /></div>
    <div><page:applyDecorator name="empty" page="/decorator/footer"/></page></div>
  </head>
</html>
```

Ici, on définit donc la pages HTML globale, on lui précise des éléments du header global qui pourront être redéfinis dans l'élément décoré puis on as ensuite la pages HTML de layout elle même.

- **<decorator:title/>** : renvoie le titre de la page décoré
- **<decorator:head/>** : renvoie le header de la page décoré
- **<decorator:body/>** : renvoie le body de la page décoré
- **<decorator name="empty" page="/decorator/footer"/>** : applique le layout donné dans l'attribut 'name' sur la page donnée dans l'attribut 'page' et met le résultat ici. Grâce à cette ligne on décore la page interne par des morceaux d'autres pages, et on remarque que l'on peut chaîner les décorateurs facilement.

V - Fonctionnalités avancées

V-A - Les DecoratorMapper

En plus du mappage standard par fichier de configuration pour définir quel décorateur utiliser pour quel page. Sitemesh permet d'utiliser d'autre stratégie de mapping. Elles sont bien sûres à définir au sein du fichier de configuration de Sitemesh (sitemesh.xml).

En voici un exemple de configuration :

Configuration d'un PageDecoratorMapper

```
<mapper class="com.opensymphony.module.sitemesh.mapper.PageDecoratorMapper">
  <param name="property.1" value="meta.decorator" />
</mapper>
```

Les différents mapper de Sitemesh

- PrintableDecoratorMapper: permet de définir un décorateur spécifique pour la version print d'une page
- PageDecoratorMapper: permet de spécifier le nom du décorateur en attribut meta de la page
- ParameterDecoratorMapper: se base du des paramètre de requête pour savoir quel décorateur utiliser
- FrameSetDecoratorMapper: mapper spécifique à l'utilisation des frame
- CookieDecoratorMapper: se base sur la valeur d'une cookie pour choisir quel décorateur utiliser
- RobotDecoratorMapper: permet de définir un décorateur spécifique pour les robots

V-B - Le tag <content>

Le tag content permet de passer des données d'une page JSP à un décorateur. Dans la page, il faut définir un tag content dans lequel on met les données, puis dans la page du décorateur, on utilise le tag decorator:getProperty pour récupérer les données. Exemple ci dessous :

Définition du tag content dans la page décorée

```
<content tag="pageName">
  Login Page
</content>
```

Récupération des données dans le décorateur

```
<decorator:getProperty property="page.pageName" />
```

On peut remarquer que Sitemesh utilise un scope pour accéder aux données de la page: meta.property (comme vu précédemment), permet d'accéder aux propriété des meta tags, et page.property permet d'accéder aux données définie dans la page même.

V-C - Accès avancés aux données de la page décorée (accès à l'objet Page)

Il y a trois manières avancées d'accéder aux données d'une page décorée:

1. En utilisant les tags JSP

```
<%@ taglib uri="http://www.opensymphony.com/sitemesh/decorator" prefix="decorator" %>
<decorator:usePage id="thePage" />
<% String author = thePage.getProperty("meta.author"); %>
```

2. Par Velocity

```
$page.getProperty("meta.author")
```

3. Accès en pure JAVA

```
import com.opensymphony.module.sitemesh.Page;
import com.opensymphony.module.sitemesh.RequestConstants;
...
Page thePage = request.getAttribute(RequestConstants.PAGE);
String author = thePage.getProperty("meta.author");
```

Je n'ai hélas pas réussi à faire fonctionner ce dernier sous WebSphere 5.1, l'objet Page obtenue est null. Mais normalement, cela est sensé fonctionner

V-D - Accéder à la requête d'une page décoré depuis un décorateur

L'objet Page contient la requête, en y accédant, on a donc accès à l'objet requête de la page décoré. Sitemesh faisant l'assemblage de la page décoré et des éléments de décorations, dans un décorateur, la requête n'est pas celle de la page décoré. L'accès à la requête se fait en JAVA.

Accéder à la requête décorée

```
import com.opensymphony.module.sitemesh.Page;
import com.opensymphony.module.sitemesh.RequestConstants;
...
Page thePage = request.getAttribute(RequestConstants.PAGE);
HttpServletRequest decoratedRequest = thePage.getRequest();
```

VI - Liens

- Site officiel de Sitemesh :  <http://www.opensymphony.com/sitemesh/>
- Un article très complet, surtout sur les DecoratorMapper :
 <http://www.onjava.com/pub/a/onjava/2004/09/22/sitemesh.html>
- La FAQ officielle :  <http://www.opensymphony.com/sitemesh/faq.html>
- L'article initial sur mon blog : <http://loicmathieu.free.fr/wordpress/index.php?p=40>
- Article sur le pattern decorator :
http://pcaboche.developpez.com/article/design-patterns/programmation-modulaire/?page=page_2#L1.2

