Getting Started with SiteMesh 3

Introduction

This tutorial is a quick introduction to using SiteMesh3 in a web-application. It covers:

- A high level overview of how SiteMesh3 works
- Installation and configuration
- Building and applying a simple decorator

It is recommend you read the high level SiteMesh 3 Overview before this tutorial.

SiteMesh in web applications

In a web application, SiteMesh acts as a Servler Filter. It allows requests to be handled by the Servlet engine as normal, but the resulting HTML (referred to as the **content**) will be intercepted before being returned to the browser.

The intercepted content has certain properties extracted (typically the contents of the <title>, <head> and <body> tags and is then passed on to a second request that should return the common look and feel for the site (referred to as the **decorator**). The decorator contains placeholders for where the properties extracted from the content should be inserted.

Under the hood, a key component of the SiteMesh architecture is the **content processor**. This is an efficient engine for transforming and extracting content from HTML content. For most users, it's fine to use it as it comes, but it is also possible to define your own transformation and extraction rules.

SiteMesh does not care what technologies are used to generate the content or the decorator. They may be static files, Servlet, JSPs, other filters, MVC frameworks, etc. So long as it's served by the Servlet engine, SiteMesh can work with it.

Dependencies

Running SiteMesh3 requires at least:

- JDK 1.5
- A Servlet 2.5 compliant container
- The SiteMesh runtime library

The SiteMesh library should be downloaded and placed in **/WEB-INF/lib/**.

Setup

Insert the SiteMesh Filter in **/WEB-INF/web.xml**:

```
<web-app>

  ...

  <filter>
    <filter-name>sitemesh</filter-name>

<filter-class>org.sitemesh.config.ConfigurableSiteMeshFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>sitemesh</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

</web-app>
```

Deploy the web-application to your Servlet container. From this point onwards, this tutorial assumes the web-app is running at http://myserver/.

Creating a decorator

The **decorator** contains the common layout and style that should be applied to the pages in the web application. It is a template that contains place holders for the content's <title>,<head> and <body> elements.

At the bare minimum, it should contain:

```
<html>
  <head>
    <title><sitemesh:write property='title'/></title>
    <sitemesh:write property='head'/>
  </head>
  <body>
    <sitemesh:write property='body'/>
  </body>
</html>
```

The **<sitemesh:write property='...'/>** tag will be rewritten by SiteMesh to include properties extracted from the content. There are more properties that can be extracted from the content and it's possible to define your own rules - that will be covered in another tutorial.

The bare minimum decorator isn't very useful. Let's add some style and a bit of common layout.

Create the file **/decorator.html** in your web-app, containing:

```
<html>
  <head>
    <title>SiteMesh example: <sitemesh:write property='title'/></title>
    <style type='text/css'>
      /* Some CSS */
     body { font-family: arial, sans-serif; background-color: #ffffcc; }
     h1, h2, h3, h4 { text-align: center; background-color: #ccffcc;
                      border-top: 1px solid #66ff66; }
     .mainBody { padding: 10px; border: 1px solid #555555; }
     .disclaimer { text-align: center; border-top: 1px solid #cccccc;
                   margin-top: 40px; color: #666666; font-size: smaller; }
    </style>
    <sitemesh:write property='head'/>
  </head>
  <body>

    <h1 class='title'>SiteMesh example site: <sitemesh:write
property='title'/></h1>

    <div class='mainBody'>
      <sitemesh:write property='body'/>
    </div>

    <div class='disclaimer'>Site disclaimer. This is an example.</div>

  </body>
</html>
```

In this example, the decorator is a static .html file, but if you want the decorator to be more dynamic, technologies such as JSP, FreeMarker, etc can be used. SiteMesh doesn't care - it just needs a path that can be served content by the Servlet engine.

Configuration

SiteMesh needs to be configured to know about this decorator and what it should do with it.

The configuration file should be created at **/WEB-INF/sitemesh3.xml**:

```
<sitemesh>
  <mapping path="/*" decorator="/decorator.html"/>
</sitemesh>
```

This tells SiteMesh that requests matching the path /* (i.e. all requests) should be decorated with /decorator.html that we just created.

If you don't like the idea of having to use XML to configure SiteMesh, don't worry - there are alternative mechanisms including directly in WEB-INF/web.xml, programatically through a Java API, through Spring, by naming convention, or any custom way you may choose to plug in. These are explained further in another article.

Creating some content

Now to create some content. This is defined in plain HTML content. Create **/hello.html**:

```
<html>
  <head>
    <title>Hello World</title>
    <meta name='description' content='A simple page'>
  </head>
  <body>
    <p>Hello <strong>world</strong>!</p>
  </body>
</html>
```

Like the decorator, the content may be static files or dynamically generated by the Servlet engine (e.g. JSP).

The result

Pointing your browser to http://myserver/hello.html will serve the content you just created, with the decorator applied. The resulting merged HTML will look like this:

```
<html>
  <head>
    <title>SiteMesh example: Hello World</title>
    <style type='text/css'>
      /* Some CSS */
      body { font-family: arial, sans-serif; background-color: #ffffcc; }
      h1, h2, h3, h4 { text-align: center; background-color: #ccffcc;
                       border-top: 1px solid #66ff66; }
      .mainBody { padding: 10px; border: 1px solid #555555; }
      .disclaimer { text-align: center; border-top: 1px solid #cccccc;
                    margin-top: 40px; color: #666666; font-size: smaller; }
    </style>
    <meta name='description' content='A simple page'>
  </head>
  <body>

    <h1 class='title'>SiteMesh example site: Hello World</h1>

    <div class='mainBody'>
      <p>Hello <strong>world</strong>!</p>
    </div>

    <div class='disclaimer'>Site disclaimer. This is an example.</div>

  </body>
</html>
```

As you can see, the <title>, <head> and <body> have been extracted from the content and inserted into the decorator template.

Summary

A quick recap:

- SiteMesh is installed by dropping the library jar in /WEB-INF/lib and creating a filter (with mapping) in /WEB-INF/web.xml
- It can be configured by creating a /WEB-INF/sitemesh3.xml file, or through other configuration methods
- The filter intercepts requests to **Content**, runs it through the **Content Processor** and merges with a **Decorator**
- The **Content** is defined by an HTML page, that contains the vanilla HTML content of the site
- The **Decorator** is also defined by an HTML page, that contains the look and feel of the site, and placeholder <sitemesh:write> tags to indicate where the **Content** should be merged in
- The **Content Processor** contains the rules for extracting and transforming the content - it has some simple default rules and can be customized