

Configuring SiteMesh 3

SiteMesh supports two main approaches to configurations - **XML** or **Java**. It's up to you which you use. In fact, you can even use them both.

XML	Java
<ul style="list-style-type: none">• Simplest to get started with• Automatically reloads when config file changes• Does not require Java programming	<ul style="list-style-type: none">• Allows for greater customization of SiteMesh• Avoids yet another configuration file• Can be used from higher level languages such as JRuby, Groovy, Scala...

XML based configuration

The configuration file should live in `/WEB-INF/sitemesh3.xml` in your web-application.

Example

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

Java based configuration

To use the Java based configuration, subclass `org.sitemesh.config.ConfigurableSiteMeshFilter` and overload the **`applyCustomConfiguration(SiteMeshFilterBuilder builder)`** method. You shall be passed an object that you can use to configure SiteMesh. You then deploy this filter in to your web-application.

Example

```
public class MySiteMeshFilter extends ConfigurableSiteMeshFilter {
    @Override
    protected void applyCustomConfiguration(SiteMeshFilterBuilder builder) {
        builder.addDecoratorPath("/*", "/decorator.html")
                .addDecoratorPath("/admin/*", "/admin/decorator.html");
    }
}
```

Note: The `SiteMeshFilterBuilder` class supports a chainable API where each method returns an instance of itself. This is a convenience, but you don't have to use this style.

Note: If you also have an XML config file, SiteMesh will load this before calling **`applyCustomConfiguration()`**. This allows you to use XML for some configuration and Java for more advanced aspects.

Configuring Decorator Mappings

This is the most common configuration applied to SiteMesh - mapping which decorators are applied based on the paths.

Things you can do:

- Map a default decorator to all paths
- Map a decorator to a specific path
- Map multiple decorators to a path - each decorator is applied to the result of the previous
- Exclude a path from being decorated

XML

```

<sitemesh>
  <!-- Map default decorator. This shall be applied to all paths if no
other paths match. -->
  <mapping decorator="/default-decorator.html"/>

  <!-- Map decorators to path patterns. -->
  <mapping path="/admin/*" decorator="/another-decorator.html"/>
  <mapping path="/*.special.jsp" decorator="/special-decorator.html"/>

  <!-- Alternative convention. This is more verbose but allows multiple
decorators
to be applied to a single path. -->
  <mapping>
    <path>/articles/*</path>
    <decorator>/decorators/article.html</decorator>
    <decorator>/decorators/two-page-layout.html</decorator>
    <decorator>/decorators/common.html</decorator>
  </mapping>

  <!-- Exclude path from decoration. -->
  <mapping path="/javadoc/*" exclude="true"/>
  <mapping path="/brochures/*" exclude="true"/>

</sitemesh>

```

Java

```

public class MySiteMeshFilter extends ConfigurableSiteMeshFilter {

  @Override
  protected void applyCustomConfiguration(SiteMeshFilterBuilder builder) {
    // Map default decorator. This shall be applied to all paths if
no other paths match.
    builder.addDecoratorPath("/*", "/default-decorator.html")
    // Map decorators to path patterns.
    .addDecoratorPath("/admin/*", "/another-decorator.html")
    .addDecoratorPath("/*.special.jsp", "/special-decorator.html")
    // Map multiple decorators to the a single path.
    .addDecoratorPaths("/articles/*", "/decorators/article.html",

"/decoratos/two-page-layout.html",
                                                                    "/decorators/common.html")

    // Exclude path from decoration.
    .addExcludedPath("/javadoc/*")
    .addExcludedPath("/brochures/*");
  }
}

```

Advanced Configuration

For most users, the decorator mappings above should be enough. But if you want more options...

MIME Types

By default, SiteMesh will only intercept responses that set the **Content-Type** HTTP header to **text/html**.

This can be altered to allow SiteMesh to intercept responses for other types. This is only applicable for the SiteMesh Filter - it is ignored by the offline site builder.

XML

```
<sitemesh>
  <mime-type>text/html</mime-type>
  <mime-type>application/vnd.wap.xhtml+xml</mime-type>
  <mime-type>application/xhtml+xml</mime-type>
  ...
</sitemesh>
```

Java

```
public class MySiteMeshFilter extends ConfigurableSiteMeshFilter {

    @Override
    protected void applyCustomConfiguration(SiteMeshFilterBuilder builder) {
        builder.setMimeTypes("text/html", "application/xhtml+xml",
            "application/vnd.wap.xhtml+xml");
    }

}
```

Deploying Tag Rule Bundles

An advanced feature of SiteMesh is the ability to define custom rules that manipulate tags on a page. These are classes that implement **org.site.mesh.content.tagrules.TagRuleBundle**.

XML

```
<sitemesh>
  <content-processor>
    <tag-rule-bundle class="com.something.CssCompressingBundle" />
    <tag-rule-bundle class="com.something.LinkRewritingBundle"/>
  </content-processor>
  ...
</sitemesh>
```

Java

```
public class MySiteMeshFilter extends ConfigurableSiteMeshFilter {

    @Override
    protected void applyCustomConfiguration(SiteMeshFilterBuilder builder) {
        builder.addTagRuleBundles(new CssCompressingBundle(), new
LinkRewritingBundle());
    }

}
```