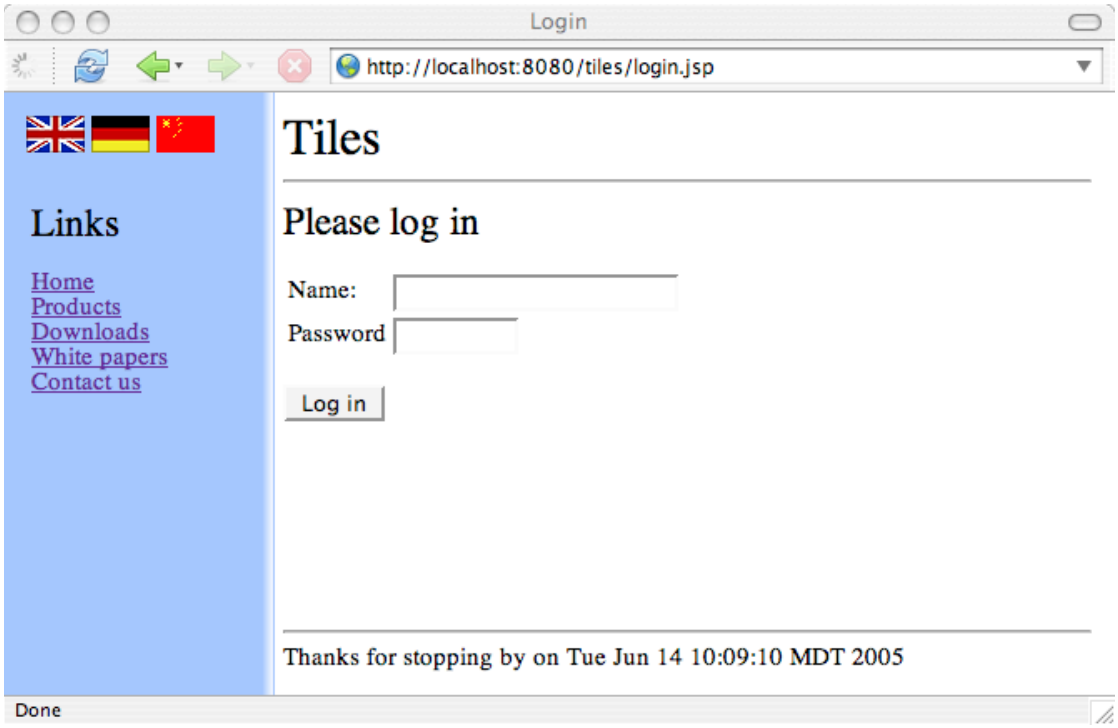
Tiles

An Introduction

This document discusses the application in the /examples/simple directory of the Tiles distribution.

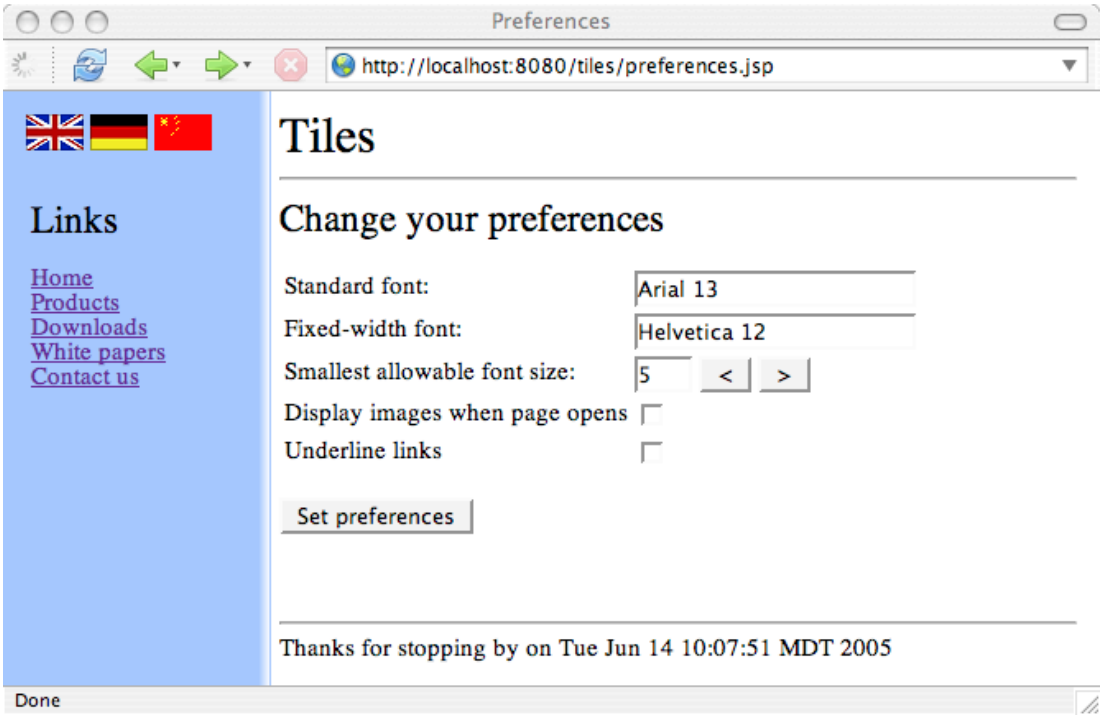
If you've used JSP, you may have used jsp:include or c:import to include shared content. That's powerful because you can automatically update all pages that include shared content by modifying the content alone.

Fundamentally, Tiles lets you reuse *layout* in addition to content. Here's an example:



The Login Page

This application has two pages: a login page and a preferences page. Here's the latter:



The Preferences Page

Like most web applications, this application has multiple pages that share content and layout. We'll use Tiles to share layout. First, we declare the Tiles servlet in web.xml:

```
<servlet>
   <servlet-name>Tiles Servlet</servlet-name>
   <servlet-class>
      org.apache.tiles.servlets.TilesServlet
   </servlet-class>
   <init-param>
      <param-name>definitions-config</param-name>
      <param-value>/WEB-INF/tiles.xml</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
</servlet>
```

The servlet container will start the Tiles servlet when it loads the application. That servlet reads the Tiles configuration file, in this case /WEB-INF/tiles.xml. That configuration file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE tiles-definitions PUBLIC
  "-//Apache Software Foundation//DTD Tiles Configuration//EN"
  "http://jakarta.apache.org/struts/dtds/tiles-config.dtd">
```

```
<tiles-definitions>
   <definition name="login-tile"
                   path="/header-footer-sidebar-layout.jsp">
      <put name="header"  value="/header.jsp"/>
      <put name="sidebar" value="/sidebar.jsp"/>
      <put name="content" value="/loginContent.jsp"/>
      <put name="footer"  value="/footer.jsp"/>
  </definition>

  <definition name="preferences-tile"
                   path="/header-footer-sidebar-layout.jsp">
      <put name="header"  value="/header.jsp"/>
      <put name="sidebar" value="/sidebar.jsp"/>
      <put name="content" value="/preferencesContent.jsp"/>
      <put name="footer"  value="/footer.jsp"/>
   </definition>

</tiles-definitions>
```

The preceeding configuration file defines two tiles: login-tile and preferences-tile. Notice that the tiles share a layout, /header-footer-sidebar-layout.jsp, specified by the path attribute. That layout looks like this:

```
<%@ taglib uri="http://jakarta.apache.org/tiles"
        prefix="tiles" %>

<%-- One table lays out all content for this attribute --%>
<table width='100%' height='100%'>
   <tr>
      <%-- Sidebar section --%>
      <td width='150' valign='top' align='left'>
         <tiles:insert attribute='sidebar'/>
      </td>

      <%-- Main content section --%>
      <td height='100%' width='*'>
         <table width='100%' height='100%'>
            <tr>
               <%-- Header section --%>
               <td valign='top' height='15%'>
                  <tiles:insert attribute='header'/>
               </td>
            <tr>

            <tr>
               <%-- Content section --%>
```

```
                <td valign='top' height='*'>
                   <tiles:insert attribute='content'/>
                </td>
             </tr>

             <tr>
                <%-- Footer section --%>
                <td valign='bottom' height='15%'>
                   <tiles:insert attribute='footer'/>
                </td>
             </tr>
          </table>
       </td>
    </tr>
</table>
```

The layout uses the tiles:insert tag to insert content defined in the tiles. This is very much like using jsp:include or c:import. But with Tiles, we specify content *indirectly*, via tile attributes, instead of *directly* via filenames, as is the the case for jsp:include or c:import. For our efforts, *our layout is reusable by any tile that defines sidebar, header, content, and footer attributes*. That's how Tiles makes layout reusable.

The only thing left is to display our tiles. Each tile has a JSP page that displays the tile; for example, here's login.jsp:

```
<%@ taglib uri="http://jakarta.apache.org/tiles" prefix="tiles" %>

<html>
   <head>
      <title>Login</title>
   </head>

   <body background="graphics/blueAndWhiteBackground.gif">
                <tiles:insert definition="login-tile"/>
   </body>
</html>
```

The preceeding JSP page inserts the login-tile into the page. There is a similar JSP page for the preferences tile.

NOTE: If you use a framework such as Struts, Shale, or MyFaces with Tiles, JSP pages that insert tiles, such as the preceeding JSP page are not necessary.

Bells and Whistles

Fundamentally, Tiles lets you encapsulate and reuse layout, but Tiles also has a considerable bag of tricks besides. Here are some features:

- Nested tiles
- An inheritance mechanism
- Role-dependent tiles
- Tile controllers

You can nest tiles inside of one another, by specifying a tile, instead of a JSP page, for a tile attribute's value. That comes in handy in composing a hierarchy of tiles. Tiles also comes with an inheritance mechanism that lets you define tiles that extend existing tile definitions; role-dependent tiles that are only displayed when the user is in the specified role; and tile controllers, which are Java objects that implement a callback that Tiles invokes just before it displays the controller's associated tile. In a controller, you can modify the current tile's attributes, which means that you can define tiles dynamically at runtime, instead of at compile time in XML, as illustrated in this document.

To illustrate the inheritance mechanism, here's a refactored version of our application's configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE tiles-definitions PUBLIC
  "-//Apache Software Foundation//DTD Tiles Configuration//EN"
  "http://jakarta.apache.org/struts/dtds/tiles-config.dtd">

<tiles-definitions>
   <definition name="header-footer-sidebar"
                   path="/header-footer-sidebar-layout.jsp">
      <put name="header"  value="/header.jsp"/>
      <put name="sidebar" value="/sidebar.jsp"/>
      <put name="content"/>
      <put name="footer"  value="/footer.jsp"/>
   </definition>

   <definition name="login-tile"
            extends="header-footer-sidebar">
      <put name="content" value="/loginContent.jsp"/>
   </definition>

   <definition name="preferences-tile"
            extends="header-footer-sidebar">
      <put name="content" value="/preferencesContent.jsp"/>
   </definition>
</tiles-definitions>
```

Now we have a tile--header-footer-sidebar--that servers as a base tile, if you will, for the login tile and the preferences tile, both of which extend header-footer-sidebar. The

header-footer-sidebar tile's content attribute is analagous to an abstract method in Java: tiles that extend header-footer-sidebar are obliged to provide a value for the content attribute.

Other Resources

Here are some other resources on Tiles:

- http://www.lilf.fr/~dumoulin/tiles/tilesAdvancedFeatures.pdf
- http://javaworld.com/javaworld/jw-09-2002/jw-0913-designpatterns_p.html
- http://www-106.ibm.com/developerworks/java/library/j-strutstiles.html?loc=j
- http://librenix.com/?inode=3792
- http://www.onjava.com/pub/a/onjava/excerpt/progjakstruts_14/index1.html