Tiles Framework

1

In this presentation, we are going to talk about Tiles framework one of the important additions to Struts 1.1.

Topics

- Evolution of Web page layout technologies
  - 4 different generations
- Tiles framework
  - Layout (template)
  - Screen definitions (definitions)
  - Tiles tag library
  - Internationalization
  - Multi-channels
  - Configuration

2

This is the list of topics we are going to talk about in this presentation.

Now before we start talking about the Tiles framework in detail, we will take a look at how Web page layout technology has evolved so far. The evolution can be categorized into 4 phases in which the Tiles framework is considered to be the 4th generation.

Then we will spend most of our time talking about the Tiles framework.

Evolution of Web Page Layout Technology (Before we talk about Tiles Framework)

3

Presentation Needs of Web Application

- Common look and feel among all pages
  - Common layout with common header, footer, menus, forms, copyright, promotions, etc.
- Easy maintenance
  - You don't want hard-code layout in each page
  - Single place to go for changing layout is desirable
- Separation of layout from content
  - Layout and contents should be able to change without affecting each other

4

Now let's talk about the presentation or display needs of a typical Web applications.

First, a web application needs to present common look and feel for all pages it displays. By “common look and feel”, we are talking about common layout structure with common header, footer, menu's, forms, copyright or promotional messages.

Second, these display pages need to be easily maintained. So the less hard-coded the layout, the easier to maintain the common look and feel. For example, it would be highly desirable to go to single place to change the common look and feel (instead of changing each and every page of a Web application).

Third, the layout and contents need to be separated. What this means is that layout and contents should be able to change without affecting the other.

Layout versus Content

5

So what is layout and what is content? This picture shows the home site of java.sun.com. Here the page has header, menu, and several different body contents. The layout defines where the header goes and where the menu goes and where the body content is displayed and so on.

Layout versus Contents

- Layout
  - How a page is logically structured
    - example: header is located on the top of the page
  - Deals with consistent Look and feel
  - Most pages in a single application share the same layout
    - header, footer, menu-bar, etc.
- Contents
  - What gets displayed
    - example: Login page has login content while logout page has logout content while both using the same layout

6

So just to repeat what I just said in the previous slide, a layout defines how a page is logically structured and it does not say anything about contents. Consequently the layout deals with consistent look and feel for all pages. So most pages in a single application shares the same layout.

The contents actually define actual data that is being displayed. For example, logon page has concrete contents regarding login process, for example, username and password fields, and logout page has some contents relevant to logging out process,

Evolution of Web Page Layout Technology (1)

- 1st generation: JSP pages with embedded HTML layout tags
  - No separation of layout from contents
- 2nd generation: JSP pages with JSP include directive's (static) or JSP include action's (dynamic)
  - Some separation of layout from contents
  - JSP pages are reusable
  - Layouts are not reusable, however, because every JSP page contains layout information – if you want to change layout, you have to change every JSP page

7

OK, we talked about differences between layout and contents. And we also learned the more separation there is between the two, the better.

Now let's take a look at the evolution of page layout technology. I categorized them into 4 generations.

The first generation technology is plain vanilla JSP page with HTML layout tags embedded in it. So here there is really no separation between the layout and contents.

In the 2nd generation technology, the JSP pages uses JSP include directives or include actions. Here there is some separation between layout and contents. The JSP pages that contain contents are reusable in several display pages. However, the layout page is not reusable because every display page contains layout information. This means, every time a layout needs to be changed, every JSP page needs to be changed.

Evolution of Web Page Layout Technology (2)

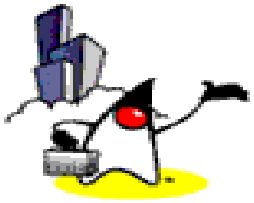
- 3rd generation: JSP templates (one defined by David Geary)
  - Separation of layout from contents
  - Both JSP pages and Layouts are reusable
- 4th generation: Tiles
  - Separation of layout from contents
  - Both JSP pages and Layouts are reusable
  - Superset of JSP templates with more features
  - Extends concept of JSP templates with "parameterized components" or "Tiles"

8

In the 3rd generation, the concept of JSP template was introduced. Here there is relatively clean separation between the layout and contents. And both JSP pages and layout pages can be reusable.

In the 4th generation, the concept of Tiles was introduced. As we will see later on, Tiles is basically a superset of template technology and in fact it extends the concept of JSP template with “parameterized components” or “tiles”.

So let's take a look at each of these technologies in a bit more detail and at the end we will spend quite a bit of our time talking about Tiles framework.



The 2nd Generation: Display Page with JSP Include Directives

9

We will start our discussion from the 2nd generation - displaying page with JSP include directives.

Example: Every display page has hard-coded layout

```
<html><head><title>Templates</title></head>
<body background='graphics/blueAndWhiteBackground.gif'>

<table width='610'>
<tr valign='top'><td><jsp:include page='sidebar.jsp'/></td>
<td><table>
<tr><td><jsp:include page='header.html'/></td></tr>
<tr><td><jsp:include page='chapter.jsp'/></td></tr>
<tr><td><jsp:include page='footer.jsp'/></td></tr>
</table>
</td>
</tr>
</table>
</body></html>
```

10

In this example display HTML page, content is included with <jsp:include> directives, which allow content to vary without modifying display page itself; however, because the layout is hard-coded, layout changes require modificatio to the display page.

If a Web application has many display pages, even simple layout changes such as removing the footer, for example, require modification of each and every page of the application.

Issues with the 2nd Gen.: JSP pages with JSP include directives

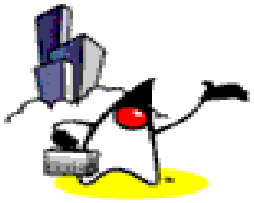
- The layout is still mixed with contents
  - Each display page explicitly specifies where *header.jsp* goes and where *footer.jsp* goes (hard-coded layout)
- Every display page has to have the same statements in the same order
  - If layout scheme needs to be changed, every display page has to be changed
- Reason for 3rd generation approach: Template

11

So this slide summarizes the issues of the 2nd generation layout technology.

First, the layout is still mixed with contents. This means, each display page explicitly specifies where header.jsp page goes and where footer.jsp goes. That is, the header page comes before footer.page.

This also means if layouts scheme needs to be changed, every display page has to be changed, which is an inconvenience. And this is the reason for the 3rd generation approach: template.



The 3rd Generation: Template

12

What is a Template?

- Template is a JSP page that uses JSP custom tag library to describe the layout of a page without specifying contents
- The template acts as a layout definition for what the pages of an application will look like (layout-wise), without specifying the content
- Content is inserted into the template page during runtime
- Several display pages use the same template
  - display page x = common template + contents x

13

So what is a template? A template is a JSP page that uses JSP template custom tag library to describe the layout of a page without specifying contents. So there is a separation of contents from the layout. And the contents is inserted into the template page during runtime.

Why Template?

- Separation of Layout from content
  - Content and layout can change without interfering each other
- A common template is shared by many display pages
  - A single place to change when layout change is required
- Template provides consistent look and feel without having to hard-code it in every page

14

So template enables the separation of layout from contents and they can change without interfering each other.

Because a common template is shared by many display pages, template provides a single place to go and change when layout change is required.

And of course, the template provides a consistent look and feel without having to hard-code it in every display page.

Example Display Page Using a Template

```
<%@ taglib URI='/WEB-INF/struts-template.tld' prefix='template' %>

<template:insert template='/defaultTemplate.jsp'>
  <template:put name='title' content='Java Passion' direct='true'/>
  <template:put name='header' content='/header.html'/>
  <template:put name='sidebar' content='/sidebar.jsp'/>
  <template:put name='content' content='/introduction.html'/>
  <template:put name='footer' content='/footer.html'/>
</template:insert>
```

15

So in this example, there is a template called “defaultTemplate.jsp”, which provides a layout in which contents can be inserted during runtime. Here a display page indicates that it is using defaultTemplate.jsp as a template and then actual contents, header.html, sidebar.jsp, introduction.html, footer.html are then inserted into the template during runtime.

Now even though this template approach definitely is in the right direction, it still lacks some flexibility as we will see in the following slides. And that is the reason why Tiles framework was devised.

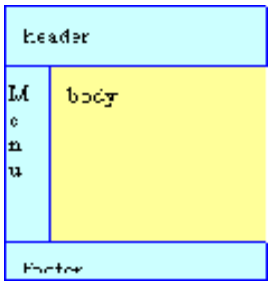
Tiles Framework

16

So Tiles framework is the 4th generation of layout technology.
And as we will discuss in the following slides, it is well-integrated with Struts.

What is Tiles Framework?

- Tiles framework allows building pages by assembling reusable Tiles
- A display page can be built by assembling a header, a footer, a menu and a body Tiles



17

The Tiles framework allows building pages by assembling reusable Tiles. As an example, the page in the next figure can be build by assembling a header, a footer, a menu and a body.

Tiles Framework

- Superset of Template
  - Tiles framework supports template functionality
    - Tiles framework use a term “Layout” for a template
  - Support parameter passing
- Extra features over Template
  - Screen definitions
  - Dynamic page building
  - Reuse of Tiles
  - Internationalization
  - Multi-channels

18

Tiles layout framework can be regarded as a superset of template technology. So Tiles framework support all the features of the template. In addition, it makes template technology more flexible by supporting parameter passing, which we will talk about later on. By the way, in the Tiles framework, the term “layout” is used to refer to “template”.

Tiles framework also provides additional features over basic template. For example, it supports what is called “screen definitions”. It also allows “dynamic page building”. The layout tiles can be reusable. And internationalization and multi-channel support are considered also extra features.

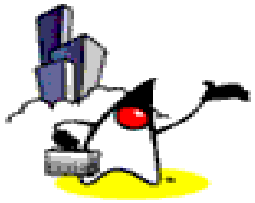
So let's spend some time talking about each of these features.

What is a Tile?

- Each Tile (header, menu, body, ...) is a JSP page and can itself be built by assembling other Tiles
- Using Tiles can be compared as using Java methods:
  - You need to define the Tiles (the method body), and then you can "call" this body anywhere you want, passing it some parameters
  - In Tiles, parameters are called "attributes" in order to avoid confusion with the request parameters

19

The Tiles framework allows building pages by assembling reusable Tiles. As an example, the page in the next figure can be build by assembling a header, a footer, a menu and a body.



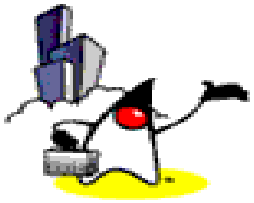
Tiles Framework: Pieces That Make up A Tiles Application (without using Definitions yet)

.

Pieces that make up Tiles App

- Layout page
  - Define a common layout
  - Layout specify attributes as filler's
  - ex: *classLayout.jsp*
- Display pages
  - Use a particular layouy page for layout
  - Specify parameter pages for actual contents
  - ex: *templateNoDef.jsp*
- Parameter pages
  - Provides actual contents
  - Shared among display pages
  - ex: *footer.jsp*

.



Tiles Framework:
Layout (Template)

22

First, let's talk about layout. Again, the functionality here is similar to what is provided in template based technology.

Layout (Page)

- Serves same purpose as Template
  - It is a JSP file
- Common layouts are defined once and reused across many different projects
  - Provides a common look and feel
    - Layout
    - Style
  - Define a menu layout and pass “lists of items and links” as parameters
  - Define a portal layout, use it by passing “list of Tiles (pages) to show” as parameters

23

So a layout serves the same purpose as a template. Developers typically define a common set of layouts and reuse them across many different projects. For example, you can define menu layouts and pass list of items and links. Or you can define a portal layout and use it as a layout by passing list of tiles to show.

You can reuse existing layouts or customize them or you can define a brand new layouts. In fact, Struts comes with a set of common layouts you can leverage with.

By the way, under Tiles framework, a layout itself is also considered as a tile. In the following slide, we will use terms like “layout tiles” or “non-layout” tiles to distinguish them when needed.

Layout

- You can reuse or customize existing layouts, or define your own ones
  - Tiles framework package comes with several pre-built layout files (We will see them in the following slide)
- Layout itself is also considered as a tile in Tiles framework
  - Called as “Layout tiles” (as opposed to “Non-layout tiles”)

24

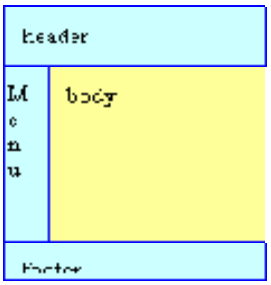
So a layout serves the same purpose as a template. Developers typically define a common set of layouts and reuse them across many different projects. For example, you can define menu layouts and pass list of items and links. Or you can define a portal layout and use it as a layout by passing list of tiles to show.

You can reuse existing layouts or customize them or you can define a brand new layouts. In fact, Struts comes with a set of common layouts you can leverage with.

By the way, under Tiles framework, a layout itself is also considered as a tile. In the following slide, we will use terms like “layout tiles” or “non-layout” tiles to distinguish them when needed.

Pre-built Layouts from Tiles Framework

- Classic layout
  - header, left menu, body, footer
- Menu layout
  - menu with links
- Vertical box layout
  - a list of tiles in a vertical column
- Columns layout
- Center layout
- Tabs layout



25

So Struts Tiles framework provides several pre-built layouts as mentioned above.

Classic layout renders a header, left menu, body, and footer.

Menu layout renders a menu with links.

Vertical box layout renders a list of tiles in a vertical column.

Columns layout renders a list of tiles in a multiple columns, each of which renders its tiles vertically stacked.

Center layout renders a header, left tile, right tile, body, and footer.

Tabs layout renders several tile in a tabs-like fashion.

Classic Layout (classicLayout.jsp)

```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<%-- Layout Tiles
  This layout create a html page with <header> and <body> tags. It render
   a header, left menu, body and footer tile.
  @param title String use in page title
  @param header Header tile (jsp url or definition name)
  @param menu Menu
  @param body Body
  @param footer Footer
--%>
<HTML>
  <HEAD>
<%-- <link rel=stylesheet
     href="<%=request.getContextPath()%>/layouts/stylesheet.css"
     type="text/css"> --%>
    <title><tiles:getAsString name="title"/></title>

  </HEAD>
```

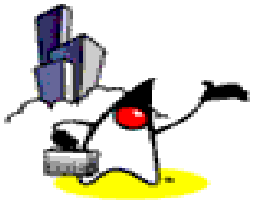

So this slide shows the classicLayout.jsp. Please note that taglib declaration of /WEB-INF/struts-tiles.tld.

Classic Layout (classicLayout.jsp)

```
<body bgcolor="#ffffff" text="#000000" link="#023264" alink="#023264"
    vlink="#023264">
<table border="0" width="100%" cellspacing="5">
<tr>
  <td colspan="2"><tiles:insert attribute="header" /></td>
</tr>
<tr>
  <td width="140" valign="top">
    <tiles:insert attribute='menu'/>
  </td>
  <td valign="top"  align="left">
    <tiles:insert attribute='body' />
  </td>
</tr>
<tr>
  <td colspan="2">
    <tiles:insert attribute="footer" />
  </td>
</tr>
</table>
</body>
</html>
```

27

This is the continuation of the classicLayout.jsp page. Here the several tags from struts <tiles> tag library is used. As we will talk about it later, the <tiles:insert> tag provides location where contents can be inserted as a parameter.



Tiles Framework: Display Page

.

Passing Parameter Pages to Layout

- Note that classicLayout.jsp (Layout page) does not know anything about content
  - this is the reason why this layout can be reused
- So content has to be supplied or passed as parameters to the layout page at runtime from your display page
  - Let's see an example of templateNoDef.jsp in the following slide
  - templateNoDef.jsp uses the classicLayout.jsp and passing contents (parameter pages) as parameters

29

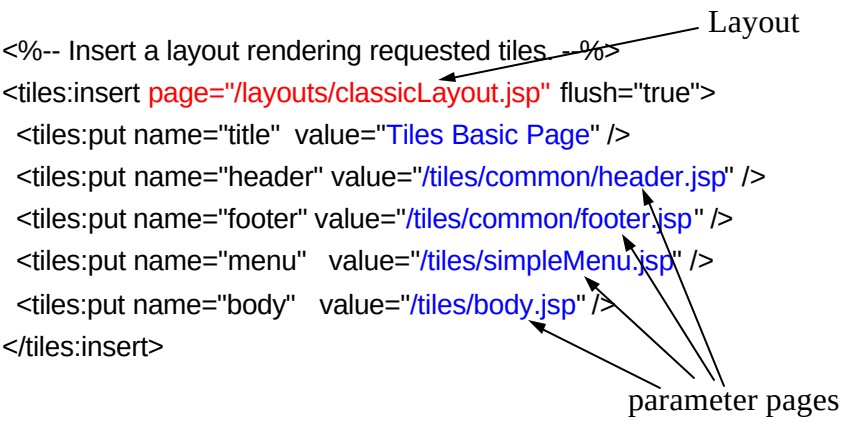
We already talked about passing contents as parameters to layout page a couple of times. So let's discuss it a bit more here.

As you have noted, the classicLayout.jsp does not know anything about content. And this is the reason why the same layout can be used by many display pages.

So contents has to be supplied or passed as parameters to the layout page at runtime. Now how do we do this? We will use templateNoDef.jsp from "struts-tiles-blank" sample code. Here I will call templateNoDef.jsp as a "display page".

templateNoDef.jsp

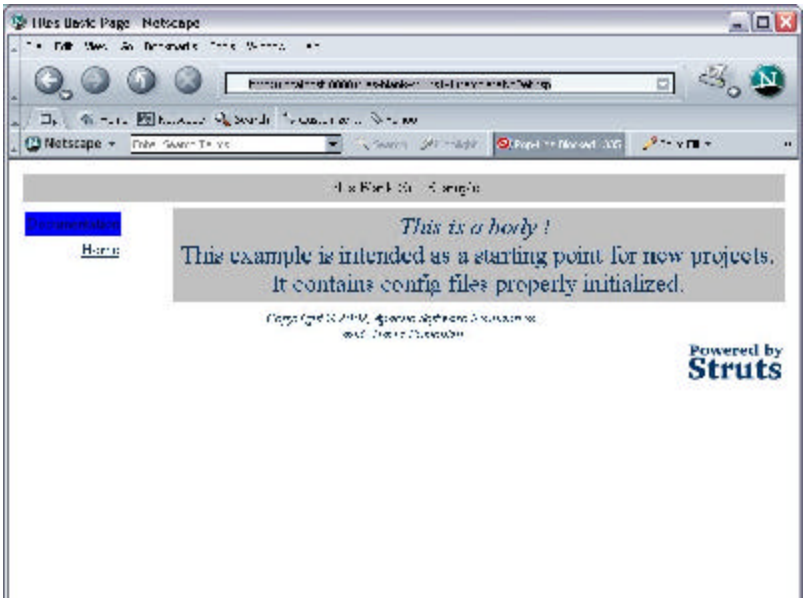
```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<%-- Insert a layout rendering requested tiles. --%>
<tiles:insert page="/layouts/classicLayout.jsp" flush="true">
  <tiles:put name="title"  value="Tiles Basic Page" />
  <tiles:put name="header" value="/tiles/common/header.jsp" />
  <tiles:put name="footer" value="/tiles/common/footer.jsp" />
  <tiles:put name="menu"   value="/tiles/simpleMenu.jsp" />
  <tiles:put name="body"   value="/tiles/body.jsp" />
</tiles:insert>
```



30

So here the layout page is declared with <tiles:insert page=”..”> element. And then actual content pages are then specified through <tiles:put> elements. Please note that the values of “name” attributes of <tiles:put> element have to match with the values of “attribute” attribute of <tiles:insert> element in the classicLayout.jsp layout page.

templateNoDef.jsp: Display Page

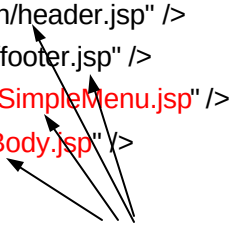


31

So when you access the display page, templateNoDef.jsp, what will happen underneath is that the contents are passed as parameters to the layout page and then the final result will be displayed and this slide shows the result.

myOwnDisplayPage.jsp

```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<%-- Insert a layout rendering requested tiles. --%>
<tiles:insert page="/layouts/classicLayout.jsp" flush="true">
  <tiles:put name="title"  value="This Is My Own Page" />
  <tiles:put name="header" value="/tiles/common/header.jsp" />
  <tiles:put name="footer" value="/tiles/common/footer.jsp" />
  <tiles:put name="menu"   value="/tiles/myOwnSimpleMenu.jsp" />
  <tiles:put name="body"   value="/tiles/myOwnBody.jsp" />
</tiles:insert>
```

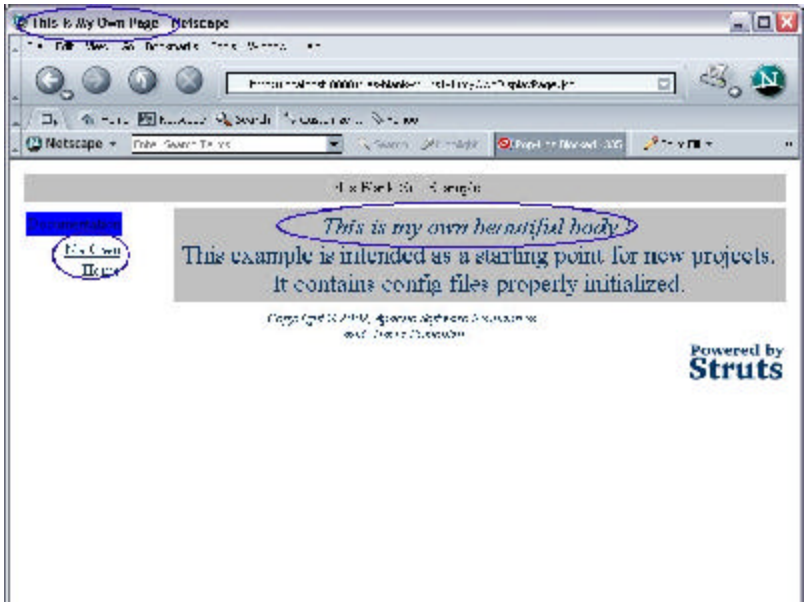


Parameter page

Now suppose you want to add another display page called myOwnDisplayPage.jsp using the same layout, classicLayout.jsp as following:

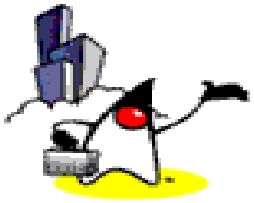
* Give a new title "This Is My Own Page"
* You want to use the same header (/tiles/common/header.jsp) and footer(/tiles/common/footer.jsp) as templateNoDef.jsp page.
* You want to use different body and menu

myOwnDisplayPage.jsp



33

So when you access the display page, templateNoDef.jsp, what will happen underneath is that the contents are passed as parameters to the layout page and then the final result will be displayed and this slide shows the result.



Tiles Framework:
Parameter Pages

.

Parameter Pages

- Provide actual contents
- Defines fragments of a JSP page
  - Should not have <head> or <body>, which are to be defined in the header
  - They are not to be accessed directly

35

We already talked about passing contents as parameters to layout page a couple of times. So let's discuss it a bit more here.

As you have noted, the classicLayout.jsp does not know anything about content. And this is the reason why the same layout can be used by many display pages.

So contents has to be supplied or passed as parameters to the layout page at runtime. Now how do we do this? We will use templateNoDef.jsp from "struts-tiles-blank" sample code. Here I will call templateNoDef.jsp as a "display page".

body.jsp (from tiles-blank-struts1-1 example)

```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<table bgcolor="#C0C0C0" cellspacing="2" cellpadding="2" border="0"
   width="100%" align="center">
<tr>
   <td align="center" >
      <font color="#023264" size="+2">
         <em>              This is a body  !            </em>
         <br>

         This example is intended as a starting point for new projects.
         It contains config files properly initialized.
      </font>
  </td>
</tr>
</table>
```

36

We already talked about passing contents as parameters to layout page a couple of times. So let's discuss it a bit more here.

As you have noted, the classicLayout.jsp does not know anything about content. And this is the reason why the same layout can be used by many display pages.

So contents has to be supplied or passed as parameters to the layout page at runtime. Now how do we do this? We will use templateNoDef.jsp from "struts-tiles-blank" sample code. Here I will call templateNoDef.jsp as a "display page".

simpleMenu.jsp (from tiles-blank-struts1-1 example)

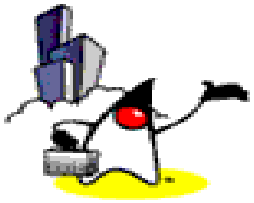
```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<table cellspacing="2" cellpadding="2" border="0" width="15%">
<tr>
   <td  bgcolor="#0000FF"> Documentation</td>
</tr>
<tr>
   <td align="right"> <a
   href="<%=request.getContextPath()%>/index.jsp"
   >Home</a></td>
</tr>
</table>
```

37

We already talked about passing contents as parameters to layout page a couple of times. So let's discuss it a bit more here.

As you have noted, the classicLayout.jsp does not know anything about content. And this is the reason why the same layout can be used by many display pages.

So contents has to be supplied or passed as parameters to the layout page at runtime. Now how do we do this? We will use templateNoDef.jsp from “struts-tiles-blank” sample code. Here I will call templateNoDef.jsp as a “display page”.



Tiles Framework:
Screen Definitions

38

Now let's talk about the concept of screen definitions under Tiles framework.

Why Definitions?

- Without definitions (in scheme shown previously), in each of display pages, there is redundant code that specifies commonly used contents
  - In the following two slides, there are two display pages (templateNoDef.jsp and templateNoDef2.jsp)
  - There are redundant code (header, footer) between the two
  - With large number of display pages, replacing, for example, a header with a different one could be a chore since you have to change them all

39

Why definitions? Without definitions, in each of non-layout tiles, there is redundant code that specifies commonly used contents. And I will show what I mean in the following two slides.

In the following two slides, you will see two non-layout tiles called “templateNoDef.jsp” and 'templateNoDef2.jsp”. And as you will, there are some redundant code between the two. When there are large number of non-layout tiles, having this redundant code could be a maintenance problem. For example, if you want to remove a footer somehow, you have to change all these non-layout tiles.

templateNoDef.jsp as display page #1

```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<%-- Insert a layout rendering requested tiles. --%>
<tiles:insert page="/layouts/classicLayout.jsp" flush="true">
  <tiles:put name="title"  value="Tiles Basic Page" />
  <tiles:put name="header" value="/tiles/common/header.jsp" />
  <tiles:put name="footer" value="/tiles/common/footer.jsp" />
  <tiles:put name="menu"   value="/tiles/simpleMenu.jsp" />
  <tiles:put name="body"   value="/tiles/body.jsp" />
</tiles:insert>
```


So this slide shows templateNoDef.jsp tile as non-layout tile #1. Here you see there are file content pieces, title, header, footer, menu, and body. Now please note the header and footer. And let's assume all the non-layout tiles have the same header and footer as we will see in the following slide.

templateNoDef2.jsp as display page #2

```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<%-- Insert a layout rendering requested tiles. --%>
<tiles:insert page="/layouts/classicLayout.jsp" flush="true">
  <tiles:put name="title"  value="Tiles Basic Page 2" />
  <tiles:put name="header" value="/tiles/common/header.jsp" />
  <tiles:put name="footer" value="/tiles/common/footer.jsp" />
  <tiles:put name="menu"   value="/tiles/simpleMenu2.jsp" />
  <tiles:put name="body"   value="/tiles/body2.jsp" />
</tiles:insert>
```

41

So this is tempateNoDef2.jsp as non-layout tile #2. Here the footer and header remains the same.

So idea of definition is to capture this common contents into a single definition file.

What is a Screen Definition?

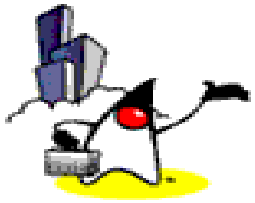
- Advanced form of layout management
  - Definitions provide an inheritance mechanism
- Definitions can take place :
  - in a centralized xml file (tiles-def.xml)
  - directly in jsp page
- You can use a defintion (i.e. mycompany.com.mytilesdefinition) instead of a JSP page (my.page.jsp)
  - This will provide contents in fully “layed out” format

.

Definitions Can be Inherited

- A definition can extend another one, overload some attributes, add new attributes
  - This allows the declaration of a "master" definition declaring the common layout, header, menu and footer
  - All other definitions extend this master layout thereby making it possible to change the entire site look & feel simply by changing the master definition.

.



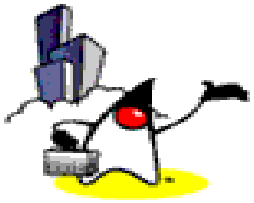
Tiles Framework: Pieces That Make up A Tiles-based Page (using Definitions)

.

Pieces that make up Tiles Page

- Definitions
- Layouts
- Display pages
- Parameter pages

45



Tiles Framework:
Tiles Definitions File

.

Where to Create Definitions?

- In a JSP page or an XML configuration file
- If you create an XML configuration file
  - Tiles Definition File
  - Usually as /WEB-INF/tiles-def.xml
    - specified in Plugin configuration
  - Contains all the definitions for the entire application

47

Tiles Definition File (WEB-INF/tiles-def.xml) From tiles-blank-struts1-1 Sample Code

```
<tiles-definitions>

 <!-- Main definition as a root for other defintions. -->
 <definition name="site.mainLayout" path="/layouts/classicLayout.jsp">
   <put name="title"  value="Tiles Blank Site" />
   <put name="header" value="/tiles/common/header.jsp" />
   <put name="menu"   value="site.menu.bar" />
   <put name="footer" value="/tiles/common/footer.jsp" />
   <put name="body"   value="/tiles/body.jsp" />
 </definition>

 <!-- This definition inherits from the main definition.
 It overload the page title, and the body used.
 Use the same mechanism to define new pages sharing common
 properties (here header, menu, footer, layout) -->
 <definition name="site.index.page" extends="site.mainLayout" >
   <put name="title"  value="Tiles Blank Site Index" />
   <put name="body"   value="/tiles/body.jsp" />
 </definition>
```

48

This is the Tiles Definition File (tiles-def.xml file) of the “struts-tiles-blank” application in our sample code. As you see in the previous slide, the “site.index.page” definition is used in the index.html page of the application. Now let's talk about a few things you need to understand in this Tile Definition File.

First off the definition names, "site.index.page" and “site.mainLayout” are unique named views within the application.

Second, the "put" elements stick attributes into the definition that the index.html page is going to use. In this case, the “title” of the page, and the “body” pages to use.

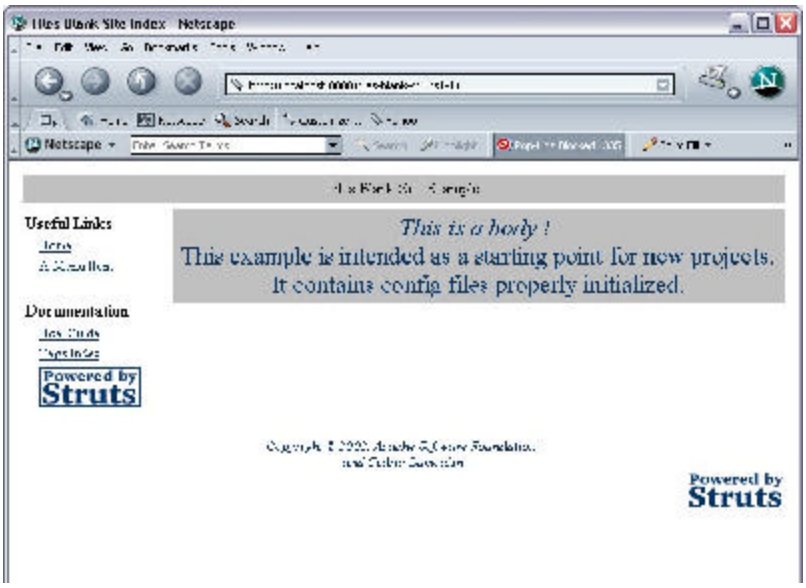
Third, “site.index.page” definition extends “site.mainLayout” definition, which means it inherits that definition's attributes. (Similar to class inheritance). Extensions are one of the ways Tiles reduce duplication between pages.

How Tiles Definition is Used: index.html of struts-tiles-blank Sample

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

  <%-- Insert a definition described in tiles configuration file
    Change the definition name to insert another definition.
    It is possible to overload some definition attribute by adding
    some put tags with appropriate name.
  --%>
<tiles:insert definition="site.index.page" flush="true" />
```


49

How Tiles Definition is Used: index.html of struts-tiles-blank Sample code



50

Tiles Definition File (WEB-INF/tiles-def.xml) From tiles-blank-struts1-1 Sample Code

```
<!-- Menu bar definition
    This definition describe a "bar" of menu stacked verticaly.
    Each menu is describe elsewhere.
    Add new entry in the list to add new menu. -->

<definition name="site.menu.bar" path="/layouts/vboxLayout.jsp" >
  <putList name="list" >
    <add value="site.menu.links" />
    <add value="site.menu.documentation" />
  </putList>
</definition>
  ...
```


.

<tiles:putList> tag

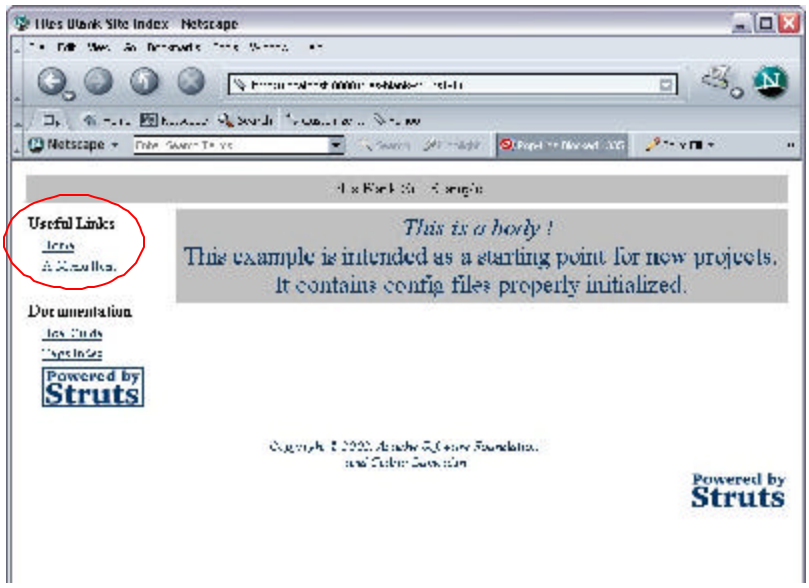
- Declare a list that will be pass as attribute to tile
- List elements are added using the tag 'add'
- Can only be used inside 'insert' or 'definition' tag

.

Tiles Definition File (WEB-INF/tiles-def.xml) From tiles-blank-struts1-1 Sample Code

```
<!-- Menu description
   A menu has a title and a set of entries rendered as links.
   Add new entry to add new links in menu.-->

<definition name="site.menu.links" path="/layouts/menuNoStruts.jsp" >
 <put name="title" value="Useful Links" />
   <putList name="items" >
    <item value="Home"
             link="/index.jsp"
             classtype="org.apache.struts.tiles.beans.SimpleMenuItem" />
    <item value="A Menu Item"
             link="/templateNoDef.jsp"
             classtype="org.apache.struts.tiles.beans.SimpleMenuItem" />
   </putList>
</definition>
 ...
```

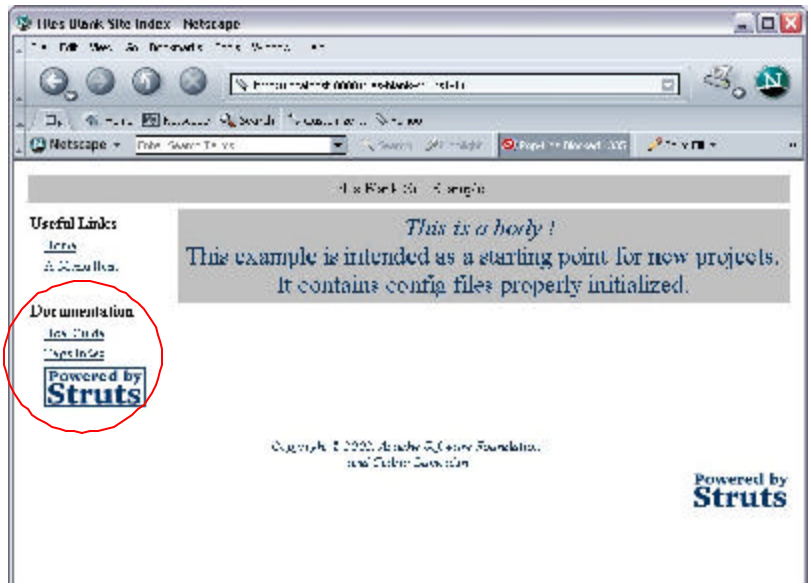
How Tiles Definition is Used: index.html of struts-tiles-blank Sample code



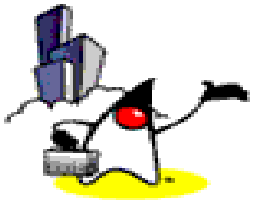
Tiles Definition File (WEB-INF/tiles-def.xml) From tiles-blank-struts1-1 Sample Code

```
<!-- Another Menu description
    A menu has a title and a set of entries rendered as links.
    Add new entry to add new links in menu.-->

<definition name="site.menu.documentation" path="/layouts/menuNoStruts.jsp" >
  <put name="title" value="Documentation" />
    <putList name="items" >
      <item value="User Guide"
                link="/index.jsp"
                classtype="org.apache.struts.tiles.beans.SimpleMenuItem" />
      <item value="Tags Index"
                link="/index.jsp"
                classtype="org.apache.struts.tiles.beans.SimpleMenuItem" />
      <item value="Struts Home"
                icon="/images/struts-power.gif"
                link="http://www.apache.org"
                classtype="org.apache.struts.tiles.beans.SimpleMenuItem" />
    </putList>
</definition>
```

How Tiles Definition is Used: index.html of struts-tiles-blank Sample code



56



Tiles Framework:
Best Practice Guidelines

OK, we already saw how Tiles tags such as <tiles:insert> and <tiles:put> are used. Now let's talk about them in a bit more detail.

Tiles Best Practice Guidelines

- How to structure the Development tree?
- Where to use Tiles framework?
- Steps for building Tiles based application

.

How to structure Development Tree?

- See development tree structure of the tiles-documentation sample application
- Layouts
  - Have all layouts under its own directory, i.e. ./web/layouts
- Tiles
  - Have all tiles (parameter pages) under its own directory - i.e. ./web/tiles
  - Under tiles directory, have the commonly used tiles under ./web/tiles/common directory

.

How to structure Development Tree?

- Definitions
  - Have definition names to follow hierarchical name space
  - Example

```
<!-- Main definition as a root for other definitions -->
 <definition name="site.mainLayout" path="/layouts/classicLayout.jsp">
   <put name="title"  value="Tiles Blank Site" />
   ...
 </definition>

 <!-- This definition inherits from the main definition.
 <definition name="site.mainLayout.nextChild" extends="site.mainLayout" >
   <put name="title"  value="Tiles Blank Site Index" />
   <put name="body"   value="/tiles/body.jsp" />
 </definition>
```

How to structure Development Tree?

- Definitions
  - Have multiple Definition files if there are many files
  - Example in struts-config.xml

```
<plug-in
 className="org.apache.struts.tiles.TilesPlugin" >
  <set-property property="definitions-config"
            value="/WEB-INF/tiles-defs-parent.xml,
                    /WEB-INF/tiles-defs-child.xml" />
  <set-property property="moduleAware" value="true"
 />
</plug-in>
```


.

Where to Use Tiles Framework?

- Wherever you place your normal JSP page can be replaced with Tiles definition
- Tiles definition provides fully laid out page contents

.

62

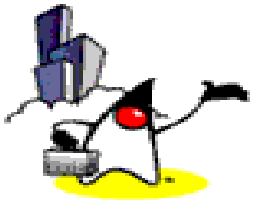
Where to use Tiles Framework?

- Instead of including JSP page
  - <jsp:include page=”my.jsp” />
- Use Tiles definition
  - <%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>
  - <tiles:insert definition="my.definition" flush="true" />

Suggested Steps for building Tiles-based Application (Review)

- Create layouts
  - See if existing layouts meet the need
  - If not, see if new layouts can be created using the existing ones
- Create Definitions in /WEB-INF/tiles-def.xml
  - See if existing definitions meet the need
- Create Tiles (parameter pages)
  - See if existing tiles meet the need
- Create Display pages

.



Tiles Framework:
Tiles Tag Library

OK, we already saw how Tiles tags such as <tiles:insert> and <tiles:put> are used. Now let's talk about them in a bit more detail.

Tags in Tiles Tag Library

- insert
- put
- definition
- getAsString
- get
- add
- importAttribute
- useAttribute
- putList
- initComponentDefinitions

66

This slide lists the tags that are supported in Tiles tag library. Among these, you will use insert, put, and definition tags the most. So let's talk about these two here.

<tiles:insert> tag

- In a Layout tile
  - Prescribes where the content will go using attribute attribute
  - example: <tiles:insert attribute='menu'/>
- In a non-layout tile
  - Retrieves a layout using page attriute
  - example: <tiles:insert page="/layouts/classicLayout.jsp" flush="true">

67

Insert tag is responsible for inserting contents into a page. Now depending on where it is used, it functions a bit differently.

In a layout tile, it prescribes where the content will go using “attribute” value. If it is used in a non-layout tile, it is used to retrieve a layout and allow content to be passed to the layout using put tags.

So let's see how these are used again in the following slides.

classicLayout.jsp as Layout tile

```
<body bgcolor="#ffffff" text="#000000" link="#023264" alink="#023264"
    vlink="#023264">
<table border="0" width="100%" cellspacing="5">
<tr>
  <td colspan="2"><tiles:insert attribute="header" /></td>
</tr>
<tr>
  <td width="140" valign="top">
    <tiles:insert attribute='menu'/>
  </td>
  <td valign="top"  align="left">
    <tiles:insert attribute='body' />
  </td>
</tr>
<tr>
  <td colspan="2">
    <tiles:insert attribute="footer" />
  </td>
</tr>
</table>
</body>
</html>
```


Here <tiles:insert> tag is used in a layout tile called classicLayout.jsp file. Here <tiles:insert> tag is used to prescribe where the content will go using “attribute” value. So this layout tile specifies where a content that is associated with “header” will go and so on.

templateNoDef.jsp as non-Layout tile

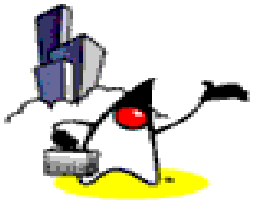
```
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>

<%-- Insert a layout rendering requested tiles. --%>
<tiles:insert page="/layouts/classicLayout.jsp" flush="true">
  <tiles:put name="title"  value="Tiles Basic Page" />
  <tiles:put name="header" value="/tiles/common/header.jsp" />
  <tiles:put name="footer" value="/tiles/common/footer.jsp" />
  <tiles:put name="menu"   value="/tiles/simpleMenu.jsp" />
  <tiles:put name="body"   value="/tiles/body.jsp" />
</tiles:insert>
```


If it is used in a non-layout tile, it is used to retrieve a layout and allow contents to be passed to the layout using <tiles:put>tags.

<tiles:put> tag

- Define an attribute to pass to tile/component/template
- Can only be used inside 'insert' or 'definition' tag
- Value (or content) is specified using attribute 'value' (or 'content'), or using the tag body

.



Tiles Framework:
Tiles & Struts

.

Using Tiles Framework with Struts

- Use the Tiles plug-in to enable Tiles definitions
- This plug-in creates the definition factory and passes it a configuration object populated with parameters
- Parameters can be specified in the web.xml file or as plug-in parameters
- The plug-in first reads parameters from web.xml, and then overloads them with the ones found in the plug-in
- All parameters are optional and can be omitted

72

Definition of a Tiles Plug-in in the struts-config.xml

```
<!-- ========== Tiles plug-in setting settings =================== -->
  <!-- Here we specified the tiles plug-in.
       This plug-in register appropriate Request Processor -->
  <!--  <controller
     processorClass="org.apache.struts.tiles.TilesRequestProcessor"  />  -->

  <!-- === Associated Messages Ressource settings ============ -->
  <!-- Not used by tiles or this website, but needed due to a bug in actual Struts
     version -->
  <message-resources parameter="org.apache.struts.webapp.tiles.dev1-
     1.ApplicationResources" null="false" />


  <plug-in className="org.apache.struts.tiles.TilesPlugin" >
    <set-property property="definitions-config"
                value="/WEB-INF/tiles-defs.xml" />
    <set-property property="moduleAware" value="true" />
  </plug-in>
```

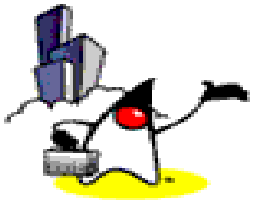

73

How a Definition can be used as Forward in Struts

- Tile definition can be used as Struts forward (instead of URLs)
- In struts-config.xml file

```
<global-forwards>
  <forward name="success"
           path="site.mainLayout" />
</global-forwards>
```


74



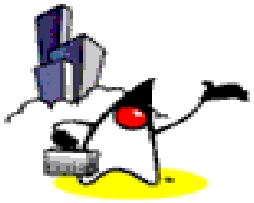
Tiles Framework: Internationalization

75

Internationalization

- It is possible to load different Tiles according to the user's Locale
- A mechanism similar to Java properties files is used for definition files: you can have one definition file per Locale, the appropriate definition is loaded according to the current Locale
  - tiles-definitions-en.xml
  - tiles-definitions-de.xml

76



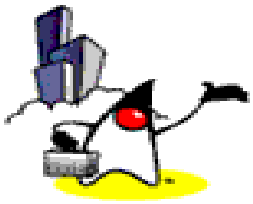
Tiles Framework:
Multi-channels

77

Multi-Channels

- It is possible to load different Tiles according to a key stored e.g. in session context
- The key could hold e.g. user privileges, browser type, ...
- A mechanism similar to Java properties files is used for definition files: you can have one definition file per key, the appropriate definition is loaded according to the key

78



Tiles Framework: How to configure Tiles framework with Struts

79

Things to Configure

- Configure WEB-INF/web.xml file

```
<taglib>
   <taglib-uri>/WEB-INF/struts-tiles.tld</taglib-uri>
   <taglib-location>/WEB-INF/struts-tiles.tld</taglib-location>
</taglib>
```

- Configure Tiles Plugin in struts-config.xml file

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
   <set-property property="definitions-config" value="/WEB-
   INF/tiles-defs.xml"/>
      <set-property property="moduleAware" value="true"/>
      <set-property property="definitions-parser-validate"
   value="true"/>
</plug-in>
```

Passion!

81

Inserting a Tile

- Inserting the body, or calling it, is done with the tag <tiles:insert ...> anywhere in a JSP page
- Insertion can also be done by specifying a definition name as the path of a Struts forward or as input, forward or include attributes of a Struts action
  - This is Struts/Tiles integration and the level of integration is possible with other frameworks

The Tiles framework allows building pages by assembling reusable Tiles. As an example, the page in the next figure can be build by assembling a header, a footer, a menu and a body.

Example: Inserting a JSP Page

- This example inserts the specified page in place of the tag

  ```
  <tiles:insert page="/layouts/commonLayout.jsp" flush="true" />
  ```

- The page attribute is any valid URL pointing to a resource inside the current site

.

Example: Inserting a Tile passing some attributes

- This example inserts the specified page, passing it the attributes.

```
<tiles:insert page="/layouts/classicLayout.jsp" flush=&quot;true">
  <tiles:put name="title"  value="Page Title" />
  <tiles:put name="header" value="/common/header.jsp" />
  <tiles:put name="footer" value="/common/footer.jsp" />
  <tiles:put name="menu"   value="/common/menu.jsp" />
  <tiles:put name="body"   value="/tiles/mainBody.jsp" />
</tiles:insert>
```

- Attributes are stored in a Tiles context which is passed to the inserted page and can then be accessed by their names

.

Example: Inserting a Tile by an attribute

- This inserts the Tiles referenced by the attribute "menu" value.

```
<tiles:insert attribute='menu' />
```

- The specified attribute value is first retrieved from current Tile's context, and then the value is used as a page target to insert.

.

Dynamic Page Building

- Tiles are gathered dynamically during page reload – Dynamic page building
- It is possible to change any attributes: layout, list of Tiles in portal, list of menu items, ...

Reuse of Tiles

- If well defined, a Tile can be reused in different locations
- Dynamic attributes are used to parameterize Tiles
- It is possible to define libraries of reusable Tiles
- Build a page by assembling predefined components, give them appropriate parameters

How Definitions are Used

- Insertion of a Tiles body can be associated to a logical name in what Tiles calls a "definition"
- A definition contains a logical name, a page used as body and some attribute values.
- The definition declaration doesn't insert the associated Tiles body. It just associates it with the name.
- A definition name can be used anywhere insertion of a Tiles body can occur.
- The associated Tiles body is then inserted with associated attributes.

.