**Apache Tiles Used in Application**

**In this Document we are going to discuss How can we Apache Tiles in application and what is need of Tiles.**

In Application every page has common part like header and footer and some where side menu also. Only few page are there which does not have common part like login page, registration page error page.

To cover these scenario we have two or three cases:

One is we can write code of common part(like header and footer) in every page of JSP but problem is that it increase size of code as well as code complexity and compiler will have to compile whole code every time for every page.

Second is we can make one JSP file for common part and include in every page in this case code complexity is decrease but size of file will be same, when compiler compile this at that time compiler first will include the code then will compiler so compiler is doing same work.

So instead of doing these things in Big Application we use tiles (Apache Tiles) tiles is nothing this is a templating language tiles is handled by Apache. It is used in Struts, tiles is best templating language so now spring also use this.

What tiles do? Tiles says make only one template or it can be more accordingly( like here we are creating home,jsp ,student.jsp, car.jsp, tyre.jsp so no need to create these jsp create only one template for all)

If in Big Application you have 100 pages, 50 pages has common part and 20 pages has bother common part and 10 has other common part and remaining 20 pages has no common part. Then in this scenario we will create three template instead of creating 80 pages of JSP (one for 50 pages second for 20 and third for 10 ) but we have to create 20 jsp page which has no common part.

Then we need to create one more JSP for dynamic content( body of page) suppose car.jsp tells about car, tyre.jsp tells about tyre, engine.jsp tells about engine.

Means Middle part is dynamic some tells about car information other tells about car manufacturing, car driving etc. middle part is dynamic make different-2 part for that.

Now only middle part will be compiled instead of whole jsp page. We can plugged in templates in dynamic page.

When we use simple JSP pages at that time we use InternalResourceViewResolver in ViewResolver there controller send view name viewResolver check prefix and suffix and the display that page. Like below:

@Bean

public ViewResolver viewResolver() {

InternalResourceViewResolver resolver = new InternalResourceViewResolver();

resolver.setPrefix("/WEB-INF/view/");

resolver.setSuffix(".jsp");

resolver.setExposeContextBeansAsAttributes(true);

return resolver;

}

but in case of Tiles we use TilesViewResolver in ViewResolver.

Here when controller send view name it goes to TilesViewResolver Because in case of Tiles we used TilesViewResolver instead of InternalResourceViewResolver. TilesViewResolver has special type configuration that is TilesConfigurer. TilesConfigurer has definition of tiles-def.xml(this has tiles definition name).

@Bean

**public** TilesConfigurer tilesConfigurer() {

TilesConfigurer tiles = **new** TilesConfigurer();

tiles.setDefinitions(**new** String[] {

"/WEB-INF/view/tiles/tiles-def.xml"

});

tiles.setCheckRefresh(**true**);

**return** tiles;

}

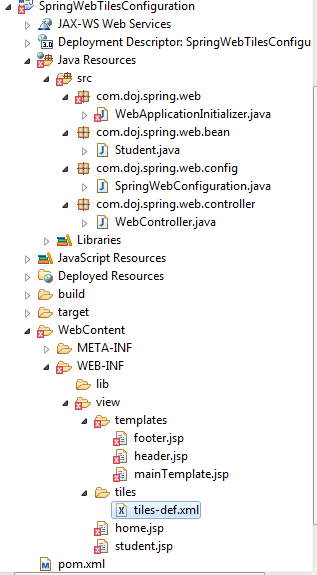
Whatever name controller return to TilesViewResolver, TilesViewResolver with help of TilesConfigurer definition name (like return home, student, that match to the definition name).

Student template extending mainTemplate( this has header and footer code). Student template has only student.jsp( this has only single form code). But student template display student.jsp form with header and footer because it extends mainTemplate.

If we want header and footer in home page then extends mainTemplate into this. Here main template is abstract so we have to provide body when home will extends this.

Header and footer, make one time and used every where body and titles are in dynamic state.

If we want to change into header and footer just change into template no need to change into every jsp file.



**Footer.jsp**

<h1>&copy; DOJ SOFTWARE CONSULTANTS</h1>

**Header.jsp**

<h1>DOJ STUDENT FORM</h1>

**mainTemplate.jsp**

<!DOCTYPE h1 PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<%@taglib uri=*"http://tiles.apache.org/tags-tiles"* prefix=*"tiles"* %>

<html>

<head>

<title><tiles:insertAttribute name=*"title"*/></title>

</head>

<body style="text-align: *center*;">

<div id=*"header"* style="height: *10%*;background-color: *gray*;"><tiles:insertAttribute name=*"header"*/></div>

<div id=*"body"* style="height: *70%*; background-color: *aqua*;padding-top: *40px*;"><tiles:insertAttribute name=*"body"*/></div>

<div id=*"footer"* style="height: *10%*;background-color: *gray*;"><tiles:insertAttribute name=*"footer"*/></div>

</body>

</html>

**tiles-def.xml**

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

<!DOCTYPE tiles-definitions PUBLIC

"-//Apache Software Foundation//DTD Tiles Configuration 2.0//EN"

"http://tiles.apache.org/dtds/tiles-config\_2\_0.dtd">

<tiles-definitions>

<definition name=*"mainTemplate"* template=*"/WEB-INF/view/templates/mainTemplate.jsp"*>

<put-attribute name=*"title"* value=*""*></put-attribute>

<put-attribute name=*"header"* value=*"/WEB-INF/view/templates/header.jsp"*></put-attribute>

<put-attribute name=*"body"* value=*""*></put-attribute>

<put-attribute name=*"footer"* value=*"/WEB-INF/view/templates/footer.jsp"*></put-attribute>

</definition>

<definition name=*"student"* extends=*"mainTemplate"*>

<put-attribute name=*"title"* value=*"STUDENT FORM"*></put-attribute>

<put-attribute name=*"body"* value=*"/WEB-INF/view/student.jsp"*></put-attribute>

</definition>

<definition name=*"home"* extends=*"mainTemplate"*>

<put-attribute name=*"body"* value=*"/WEB-INF/view/home.jsp"*></put-attribute>

<put-attribute name=*"title"* value=*"WELCOME DOJ CLASS"*></put-attribute>

</definition>

</tiles-definitions>

Home.jsp

<h1>${name} Welcome to DOJ Classes for Spring MVC!!!</h1>

**Student.jsp**

<form action=*"dojstudent"* method=*"post"*>

<table style="text-align: *center*; position: *absolute*;padding-left: *500px*;">

<tr>

<td>First Name : </td>

<td><input name=*"fname"* type=*"text"*></td>

</tr>

<tr>

<td>Last Name : </td>

<td><input name=*"lname"* type=*"text"*></td>

</tr>

<tr>

<td>Address : </td>

<td><input name=*"address"* type=*"text"*></td>

</tr>

<tr>

<td>Course : </td>

<td><input name=*"course"* type=*"text"*></td>

</tr>

<tr>

<td></td>

<td><input type=*"Submit"* name=*"Submit"* value=*"Submit"*></td>

</tr>

</table>

</form>

WebApplicationInitializer.java

package com.doj.spring.web;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

import com.doj.spring.web.config.SpringWebConfiguration;

//this file is equivalent to web.xml

public class WebApplicationInitializer extends AbstractAnnotationConfigDispatcherServletInitializer{

//Configuration for non web components like services, daos, repos, etc.

@Override

protected Class<?>[] getRootConfigClasses() {

return null;

}

//Specifying Spring MVC configuration class "SpringWebConfiguration.class" it equivalent to \*-servlet.xml file

@Override

protected Class<?>[] getServletConfigClasses() {

return new Class<?>[]{SpringWebConfiguration.class};

}

//Mapping dispatcher server to "/" i.e. Servlet Mapping in the web.xml

@Override

protected String[] getServletMappings() {

return new String[]{"/"};

}

}

Student.java

**package** com.doj.spring.web.bean;

**public** **class** Student {

String fname;

String lname;

String address;

String course;

**public** String getFname() {

**return** fname;

}

**public** **void** setFname(String fname) {

**this**.fname = fname;

}

**public** String getLname() {

**return** lname;

}

**public** **void** setLname(String lname) {

**this**.lname = lname;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** String getCourse() {

**return** course;

}

**public** **void** setCourse(String course) {

**this**.course = course;

}

}

SpringWebConfiguration.java

**package** com.doj.spring.web.config;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.web.servlet.ViewResolver;

**import** org.springframework.web.servlet.config.annotation.DefaultServletHandlerConfigurer;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

**import** org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

**import** org.springframework.web.servlet.view.tiles3.TilesConfigurer;

**import** org.springframework.web.servlet.view.tiles3.TilesViewResolver;

@Configuration

@EnableWebMvc

@ComponentScan("com.doj.spring.web.controller")

**public** **class** SpringWebConfiguration **extends** WebMvcConfigurerAdapter{

//Apache tiles

//Configuring and creating bean for view resolver

/\*@Bean

public ViewResolver viewResolver() {

InternalResourceViewResolver resolver = new InternalResourceViewResolver();

resolver.setPrefix("/WEB-INF/view/");

resolver.setSuffix(".jsp");

resolver.setExposeContextBeansAsAttributes(true);

return resolver;

}\*/

@Bean

**public** ViewResolver viewResolver() {

**return** **new** TilesViewResolver();

}

@Bean

**public** TilesConfigurer tilesConfigurer() {

TilesConfigurer tiles = **new** TilesConfigurer();

tiles.setDefinitions(**new** String[] {

"/WEB-INF/view/tiles/tiles-def.xml"

});

tiles.setCheckRefresh(**true**);

**return** tiles;

}

//Configure for default static content handling

@Override

**public** **void** configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {

configurer.enable();

}

}

WebController.java

**package** com.doj.spring.web.controller;

**import** java.util.HashMap;

**import** java.util.Map;

**import** javax.servlet.http.HttpServletRequest;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.ModelMap;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.servlet.ModelAndView;

**import** com.doj.spring.web.bean.Student;

@Controller

**public** **class** WebController {

//multiple mapping with one remote method

@RequestMapping(value={"/", "/index","/home","/welcome"})

**public** String home(){

**return** "home";

}

//With Model and Model Name to View Resolver

@RequestMapping("/indexc")

**public** ModelAndView welcome(){

Map<String, String> model = **new** HashMap<>();

model.put("name", "Sumit");

**return** **new** ModelAndView("home","model", model);

}

//We are using Spring ModelMap for return the model value

@RequestMapping("/hello")

**public** String index(ModelMap model){

model.put("name", "Sumit");

**return** "home";

}

//We are using Spring ModelMap and fetching request parameter here

@RequestMapping("/doj")

**public** String hello(ModelMap model, HttpServletRequest request){

String name = request.getParameter("name");

model.put("name", name);

**return** "home";

}

//We are using Spring ModelMap and Mapping the attribte with request param annotation with attributes whatever is ur requirement

@RequestMapping("/dojc")

**public** String doj(ModelMap model, @RequestParam(defaultValue = "DOJ Students", required=**true**, value="fname") String name,

@RequestParam(required = **false**, value="lname") String sname){

**if**(sname != **null**){

name = name +" "+ sname;

}

model.put("name", name);

**return** "home";

}

@RequestMapping("/doj-student-{fname}-{lname}")

**public** String dojStudent(ModelMap model, @PathVariable(value="fname") String name, @PathVariable(value="lname") String sname,

@RequestParam String address){

**if**(sname != **null**){

name = name +" "+ sname;

}

model.put("name", name);

**return** "home";

}

@RequestMapping(value="/doj-student", method=RequestMethod.GET)

**public** String getDojStudent(ModelMap model, Student student){

String name = **null**;

**if**(student.getFname() != **null**){

name = student.getFname();

}

**if**(student.getLname() != **null**){

name = name + " " +student.getLname();

}

**if**(student.getAddress() != **null**){

name = name+" "+student.getAddress();

}

**if**(student.getCourse() != **null**){

name = name+" "+student.getCourse();

}

model.put("name", name);

**return** "home";

}

@RequestMapping(value = "/dojstudent", method=RequestMethod.GET)

**public** String student(){

**return** "student";

}

@RequestMapping(value="/dojstudent", method=RequestMethod.POST)

**public** String getStudent(ModelMap model, Student student){

String name = **null**;

**if**(student.getFname() != **null**){

name = student.getFname();

}

**if**(student.getLname() != **null**){

name = name + " " +student.getLname();

}

**if**(student.getAddress() != **null**){

name = name+" "+student.getAddress();

}

**if**(student.getCourse() != **null**){

name = name+" "+student.getCourse();

}

model.put("name", name);

**return** "home";

}

}

Pom.xml

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.DOJSpring</groupId>

<artifactId>SpringWebTilesConfiguration</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>SpringWebTilesConfiguration</name>

<description>SpringWebTilesConfiguration is web application with configuring tiles templates</description>

<properties>

<spring.version>4.0.0.RELEASE</spring.version>

<aspect.version>1.6.11</aspect.version>

<tiles.version>3.0.5</tiles.version>

<jstl.version>1.2</jstl.version>

</properties>

<dependencies>

<!-- Spring Core dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring Core dependency -->

<!-- Spring AOP dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>${aspect.version}</version>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>${aspect.version}</version>

</dependency>

<!-- Spring AOP dependency -->

<!-- Spring MVC dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Spring MVC dependency -->

<!-- Tiles dependency -->

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-core</artifactId>

<version>${tiles.version}</version>

</dependency>

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-jsp</artifactId>

<version>${tiles.version}</version>

</dependency>

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-api</artifactId>

<version>${tiles.version}</version>

</dependency>

<!-- Tiles dependency -->

<!-- JSTL dependency -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<!-- JSTL dependency -->

</dependencies>

<build>

<sourceDirectory>src</sourceDirectory>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.1</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.3</version>

<configuration>

<warSourceDirectory>WebContent</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

</project>